



ユーザーガイド

# AWS Identity and Access Management



# AWS Identity and Access Management: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、お客様に混乱を招く可能性が高い方法、または Amazon の評判もしくは信用を損なう方法で、Amazon が所有しない製品またはサービスと関連付けて使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

# Table of Contents

IAM とは .....	1
IAM の紹介ビデオ .....	2
IAM の機能 .....	2
IAM へのアクセス .....	4
IAM をいつ使用しますか .....	5
さまざまな職務を遂行しているとき .....	5
AWS リソースへのアクセスが許可されている場合 .....	5
IAM ユーザーとしてサインインした場合 .....	6
IAM ロールを引き受けるとき .....	7
ポリシーと許可を作成するとき .....	8
IAM の仕組み .....	9
規約 .....	11
Principal .....	13
リクエスト .....	13
認証 .....	13
認可 .....	14
アクションまたはオペレーション .....	15
リソース .....	15
AWS のユーザー .....	15
初回アクセスのみ: ルートユーザーの認証情報 .....	16
IAM ユーザーと IAM Identity Center 内のユーザー .....	16
既存のユーザーのフェデレーション .....	17
アクセスコントロール方法 .....	18
IAM のアクセス許可とポリシー .....	22
ポリシーとアカウント .....	22
ポリシーとユーザー .....	22
ポリシーとグループ .....	23
フェデレーテッドユーザーとロール .....	24
アイデンティティベースのポリシーとリソースベースのポリシー .....	24
ABAC とは .....	25
ABAC と従来の RBAC モデルの比較 .....	26
IAM の外部でのセキュリティ機能 .....	27
一般的なタスクへのクイックリンク .....	29
IAM コンソール検索 .....	32

IAM コンソール検索の使用 .....	33
IAM コンソール検索結果のアイコン .....	33
サンプルの検索語句 .....	34
AWS CloudFormation リソース .....	35
IAM および AWS CloudFormation テンプレート .....	35
AWS CloudFormation の詳細はこちら .....	36
AWS CloudShell を使用する .....	36
AWS CloudShell の IAM アクセス許可の取得 .....	36
AWS CloudShell を使用して IAM とやりとりする .....	37
AWS SDK の操作 .....	39
準備作業 .....	41
AWS アカウントへのサインアップ .....	42
管理アクセス権を持つユーザーを作成する .....	42
最小特権アクセス許可に備える .....	43
IAM 管理メソッド .....	44
AWS コンソール .....	45
AWS コマンドラインインターフェイス (CLI) と Software Development Kit (SDK) .....	46
AWS アカウント ID とそのエイリアス .....	48
AWS アカウント ID を表示する .....	49
アカウントのエイリアスについて .....	50
AWS アカウント のエイリアスの作成、削除および一覧表示 .....	51
開始 .....	55
前提条件 .....	55
最初の IAM ユーザーを作成する .....	55
最初のロールを作成します .....	57
最初の IAM ポリシーの作成 .....	60
プログラマ的なアクセス .....	61
セキュリティのベストプラクティスとユースケース .....	63
セキュリティベストプラクティス .....	63
人間のユーザーが一時的な認証情報を使用して AWS にアクセスする場合に ID プロバイ ダーとのフェデレーションを使用することを必須とする .....	64
ワークロードが AWS にアクセスする場合に IAM ロールで一時的な資格情報を使用するこ とを必須とする .....	65
多要素認証 (MFA) を必須とする .....	65
長期的な認証情報を必要とするユースケースのためにアクセスキーを必要な時に更新す る .....	66



ルートユーザーの認証情報を保護するためのベストプラクティスに沿う .....	67
最小特権アクセス許可を適用する .....	67
AWS 管理ポリシーの使用を開始し、最小特権のアクセス許可に移行する .....	67
IAM Access Analyzer を使用して、アクセスアクティビティに基づいて最小特権ポリシーを生成する .....	67
未使用のユーザー、ロール、アクセス許可、ポリシー、および認証情報を定期的に確認して削除する .....	68
IAM ポリシーで条件を指定して、アクセスをさらに制限する .....	68
IAM Access Analyzer を使用して、リソースへのパブリックアクセスおよびクロスアカウントアクセスを確認する .....	68
IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保する .....	69
複数のアカウントにまたがるアクセス許可のガードレールを確立する .....	69
アクセス許可の境界を使用して、アカウント内のアクセス許可の管理を委任する .....	69
ルートユーザーのベストプラクティス .....	70
ルートのユーザーの認証情報を保護して不正使用を防止する .....	71
アクセスを保護するために、強度の高いルートユーザーパスワードを設定します .....	71
多要素認証 (MFA) でルートユーザーのサインインを保護する .....	72
ルートユーザーのアクセスキーは作成しないでください .....	72
可能な場合、ルートユーザーのサインインには複数人による承認を求る .....	72
ルートユーザーの認証情報にグループ E メールアドレスを使用する .....	73
アカウント回復メカニズムへのアクセスを制限する .....	73
Organizations アカウントのルートユーザー認証情報を保護する .....	74
アクセスおよび使用状況をモニタリングする .....	74
ビジネスユースケース .....	76
Example Corp の初期設定 .....	76
Amazon EC2 での IAM ユースケース .....	77
Amazon S3 での IAM のユースケース .....	79
チュートリアル .....	81
請求コンソールへのアクセス権の付与 .....	81
前提条件 .....	83
ステップ 1: テスト用の AWS アカウントで請求データへの IAM アクセスを有効にする .....	83
ステップ 2: テスト用のユーザーとグループを作成する .....	84
ステップ 3: AWS Billing コンソールへのアクセス権を付与するロールを作成する .....	86
ステップ 4: コンソールへのアクセスをテストする .....	87
まとめ .....	89

関連リソース .....	89
AWS アカウント 間のロールを使用したアクセスの委任 .....	90
前提条件 .....	91
本番稼働用アカウントでロールを作成する .....	92
ロールにアクセス許可を付与する .....	96
ロールを切り換えてアクセスをテストする .....	98
関連リソース .....	103
まとめ .....	104
カスタマー管理ポリシーを作成する .....	104
前提条件 .....	105
ステップ 1: ポリシーを作成する .....	105
ステップ 2: ポリシーのアタッチ .....	106
ステップ 3: ユーザーアクセスのテスト .....	107
関連リソース .....	107
まとめ .....	107
属性ベースのアクセスコントロール (ABAC) を使用する .....	107
チュートリアル概要 .....	108
前提条件 .....	110
ステップ 1: テストユーザーを作成する .....	110
ステップ 2: ABAC ポリシーを作成する .....	112
ステップ 3: ロールを作成する .....	116
ステップ 4: シークレットの作成をテストする .....	118
ステップ 5: シークレットの表示をテストする .....	121
ステップ 6: スケーラビリティをテストする .....	123
ステップ 7: シークレットの更新と削除をテストする .....	125
[概要] .....	126
関連リソース .....	127
ABAC での SAML セッションタグの使用 .....	127
ユーザーが認証情報と MFA 設定を管理できるようにする .....	132
前提条件 .....	133
ステップ 1: MFA サインインを強制するポリシーを作成する .....	134
ステップ 2: テストユーザーグループにポリシーをアタッチする .....	135
ステップ 3: ユーザーアクセスをテストする .....	136
関連リソース .....	138
ID .....	139
AWS アカウント ルートユーザー .....	140

IAM ユーザー .....	140
IAM ユーザーグループ .....	141
IAM ロール .....	141
IAM での一時認証情報 .....	142
IAM Identity Center ユーザーを使用する場合とは? .....	143
IAM ユーザーの作成が適している場合 (ロールではなく) .....	143
IAM ロールの作成が適している場合 (ユーザーではなく) .....	144
AWS アカウントのルートユーザー と IAM ユーザーを比較する .....	145
AWS アカウントのルートユーザー .....	146
AWS アカウントのルートユーザー の MFA を有効にする (コンソール) .....	147
パスワードを変更する .....	156
紛失または忘れたルートユーザーのパスワードのリセット .....	158
ルートユーザーのアクセスキーの作成 .....	159
ルートユーザーのアクセスキーの削除 .....	162
ルートユーザーが必要なタスク .....	163
ルートユーザーの問題のトラブルシューティング .....	165
関連情報 .....	166
Users .....	167
AWS が IAM ユーザーを識別する方法 .....	167
IAM ユーザーと認証情報 .....	167
IAM ユーザーおよびアクセス許可 .....	169
IAM ユーザーとアカウント .....	170
IAM ユーザーとサービスアカウント .....	170
ユーザーの追加 .....	170
コンソール へのユーザーアクセスのコントロール .....	178
IAM ユーザーが AWS にサインインする方法 .....	180
ユーザーの管理 .....	184
ユーザーのアクセス許可の変更 .....	191
パスワードの管理 .....	198
アクセスキー .....	218
紛失したパスワードまたはアクセスキーの取得 .....	235
多要素認証 (MFA) .....	236
使用していない認証情報の検索 .....	313
認証情報レポートの取得 .....	317
CodeCommit での IAM の使用 .....	323
Amazon Keyspaces での IAM の使用 .....	327

サーバー証明書の管理 .....	328
ユーザーグループ .....	335
ユーザーグループを作成する .....	337
ユーザーグループの管理 .....	339
ロール .....	346
用語と概念 .....	348
一般的なシナリオ .....	352
サービスにリンクされたロール .....	371
ロールの作成 .....	385
ロールの使用 .....	425
ロールの管理 .....	599
ID プロバイダーとフェデレーション .....	624
IAM Identity Center とのフェデレーション .....	625
IAM とのフェデレーション .....	626
Amazon Cognito アイデンティティプールとのフェデレーション .....	626
一般的なシナリオ .....	627
OIDC フェデレーション .....	632
SAML 2.0 フェデレーション .....	651
一時的な認証情報 .....	683
AWS STS と AWS リージョン .....	683
一時的な認証情報の一般的なシナリオ .....	684
一時的なセキュリティ認証情報のリクエスト .....	686
AWS リソースを使用した一時的な認証情報の使用 .....	704
一時的なセキュリティ認証情報のアクセス権限を制御する .....	708
AWS リージョン での AWS STS の管理 .....	741
ベアラートークンを使用する .....	752
一時認証情報を使用するサンプルアプリケーション .....	753
カスタム ID ブロカーに対する AWS コンソールへのアクセスの許可 .....	754
一時的な認証情報のための追加リソース .....	769
IAM リソースのタグ付け .....	770
AWS タグ命名規則を選択する .....	770
IAM および AWS STS でのタグ付けの規則 .....	772
IAM ユーザーのタグ付け .....	775
IAM ロールのタグ付け .....	778
カスタマー管理ポリシーのタグ付け .....	781
IAM ID プロバイダーのタグ付け .....	784

インスタンスプロファイルのタグ付け .....	791
サーバー証明書のタグ付け .....	793
仮想 MFA デバイスのタグ付け .....	796
セッションタグ .....	798
CloudTrail によるイベントのログ記録 .....	812
CloudTrail での IAM および AWS STS 情報 .....	813
IAM および AWS STS API リクエストのログ記録 .....	813
他の AWS のサービスへの API リクエストのログ記録 .....	814
ユーザーサインインイベントのログ記録 .....	814
一時的な認証情報のサインインイベントのログ記録 .....	815
CloudTrail ログの IAM API イベントの例 .....	817
例 AWS STS CloudTrail ログの API イベント .....	818
CloudTrail ログのサインインイベントの例 .....	828
IAM ロールの信頼ポリシーの動作 .....	831
アクセス管理 .....	832
アクセス管理リソース .....	833
ポリシーとアクセス許可 .....	834
ポリシータイプ .....	834
ポリシーとルートユーザー .....	840
JSON ポリシー概要 .....	840
最小限の特権を認める。 .....	845
管理ポリシーとインラインポリシー .....	846
データ境界 .....	856
アクセス許可の境界 .....	861
アイデンティティとリソースの比較 .....	875
ポリシーを使用したアクセス制御 .....	879
タグを使用して IAM ユーザーおよびロールへのアクセスを制御します .....	892
タグにより AWS リソースへのアクセスを制御します .....	894
クロスアカウントのリソースへのアクセス .....	899
転送アクセスセッション .....	906
ポリシーの例 .....	909
IAM ポリシーを管理する .....	988
IAM ポリシーの作成 .....	989
ポリシーの検証 .....	999
ポリシーの生成 .....	1000
IAM ポリシーをテストする .....	1001

ID アクセス許可の追加または削除 .....	1017
IAM ポリシーのバージョニング .....	1029
IAM ポリシーの編集 .....	1034
IAM ポリシーを削除する .....	1040
アクセス情報を使用したアクセス許可の調整 .....	1044
ポリシーについて .....	1588
ポリシー概要 (サービスの一覧) .....	1589
サービス概要 (アクションのリスト) .....	1603
アクション概要 (リソースのリスト) .....	1609
ポリシー概要の例 .....	1613
必要な許可 .....	1623
IAM ID を管理するためのアクセス許可 .....	1623
AWS Management Consoleで作業するための許可 .....	1625
AWS アカウント全体にわたるアクセス権限の付与 .....	1626
あるサービスから他のサービスへのアクセス権限 .....	1626
必須アクション .....	1627
IAM のポリシーの例 .....	1628
コードの例 .....	1632
IAM .....	1637
アクション .....	1652
シナリオ .....	2222
AWS STS .....	2577
アクション .....	2578
シナリオ .....	2606
セキュリティ .....	2625
AWS セキュリティ認証情報 .....	2626
セキュリティに関する考慮事項 .....	2627
フェデレーティッド ID .....	2628
多要素認証 (MFA) .....	2628
プログラマ的なアクセス .....	2629
長期的なアクセスキーに対する代替方法 .....	2631
AWS にAWS認証情報を使用してアクセスする .....	2633
AWS セキュリティ監査のガイドライン .....	2633
セキュリティ監査を実行するタイミング .....	2634
監査のガイドライン .....	2634
AWS アカウントの認証情報の確認 .....	2635

IAM ユーザーの確認 .....	2635
IAM グループの確認 .....	2636
IAM ロールの確認 .....	2636
SAML および OpenID Connect (OIDC) 用 IAM プロバイダの確認 .....	2636
モバイルアプリの確認 .....	2636
IAM ポリシーを確認するためのヒント .....	2637
データ保護 .....	2639
IAM および AWS STSでのデータの暗号化 .....	2640
IAM および AWS STS のキーの管理 .....	2640
IAM および AWS STS のネットワーク間トラフィックプライバシー .....	2640
ログ記録とモニタリング .....	2641
コンプライアンス検証 .....	2642
耐障害性 .....	2643
IAM レジリエンスのためのベストプラクティス .....	2645
インフラストラクチャセキュリティ .....	2646
設定と脆弱性の分析 .....	2647
AWS マネージドポリシー .....	2647
IAMReadOnlyAccess .....	2648
IAMUserChangePassword .....	2648
IAMAccessAnalyzerFullAccess .....	2649
IAMAccessAnalyzerReadOnlyAccess .....	2650
AccessAnalyzerServiceRolePolicy .....	2651
.....	2654
ポリシーの更新 .....	2654
IAM Access Analyzer .....	2659
外部エンティティと共有されているリソースを識別する .....	2659
IAM ユーザーおよびロールに付与された未使用のアクセスを特定する .....	2661
AWS ベストプラクティスに照らしてポリシーを検証する .....	2662
指定したセキュリティ標準に照らしてポリシーを検証する .....	2662
ポリシーの生成 .....	2663
IAM Access Analyzer の価格設定 .....	2663
外部アクセスと未使用のアクセスに関する検出結果 .....	2663
IAM Access Analyzer の結果 .....	2665
IAM Access Analyzer の結果の開始方法 .....	2667
検出結果ダッシュボード .....	2673
結果を使用する .....	2677

調査結果を確認する .....	2678
検出結果のフィルタリング .....	2682
結果のアーカイブ .....	2687
結果の解決 .....	2687
サポートされているリソースタイプ .....	2691
設定 .....	2698
アーカイブルール .....	2701
EventBridge によるモニタリング .....	2703
セキュリティハブの統合 .....	2712
CloudTrail によるログ記録 .....	2720
IAM Access Analyzer フィルターキーにアクセスする .....	2723
サービスにリンクされたロールの使用 .....	2733
アクセスのプレビュー .....	2736
Amazon S3 コンソールでのアクセスのプレビュー .....	2737
IAM Access Analyzer API を使用したアクセスのプレビュー .....	2738
ポリシーを検証するためのチェック .....	2742
IAM Access Analyzer ポリシーの検証 .....	2742
カスタムポリシーチェック .....	2848
IAM Access Analyzer ポリシーの生成 .....	2852
ポリシー生成の仕組み .....	2853
サービスレベルとアクションレベルの情報 .....	2853
主要事項 .....	2854
必要なアクセス許可 .....	2855
CloudTrail アクティビティに基づくポリシーの生成 (コンソール) .....	2858
別のアカウントの AWS CloudTrail データを使用してポリシーを生成する .....	2862
CloudTrail アクティビティに基づくポリシーの生成 (AWS CLI) .....	2865
CloudTrail アクティビティに基づいたポリシーの生成 (AWS API) .....	2866
IAM Access Analyzer のポリシー生成サービス .....	2866
IAM Access Analyzer のクォータ .....	2877
IAM のトラブルシューティング .....	2879
一般的な問題 .....	2879
自分の AWS アカウントにサインインできません .....	2880
アクセスキーを紛失した .....	2880
ポリシーの変数が機能していない .....	2880
行った変更がすぐに表示されないことがある .....	2881
iam:DeleteVirtualMFADevice を実行する権限がありません .....	2881



IAM ユーザーを安全に作成するにはどうすればよいですか？ .....	2882
その他のリソース .....	2883
アクセス拒否エラーメッセージ .....	2883
AWS サービスに要求を送信すると "アクセス拒否" が発生する .....	2884
一時的な認証情報を使用して要求を送信すると "アクセス拒否" が発生する .....	2886
アクセス拒否の例 .....	2887
IAM ポリシー .....	2893
ビジュアルエディタを使用したトラブルシューティング .....	2894
ポリシー概要を使用したトラブルシューティング .....	2899
ポリシー管理のトラブルシューティング .....	2908
JSON ポリシードキュメントのトラブルシューティング .....	2909
FIDO セキュリティキー .....	2915
FIDO セキュリティキーを有効にできない .....	2915
自分の FIDO セキュリティキーを使用してサインインできない .....	2916
FIDO セキュリティキーの紛失または破損した場合 .....	2917
その他の問題 .....	2917
IAM ロール .....	2917
ロールを引き受けることができない .....	2918
AWS アカウントに新しいロールが表示される .....	2920
自分の AWS アカウント でロールを編集または削除できない .....	2920
次のことを実行する権限がない: iam:PassRole .....	2921
12 時間のセッションに使用するロールを引き受けることができない (AWS CLI、AWS API) .....	2921
IAM コンソールでロールを切り替えようとするエラーが発生する .....	2922
ロールには、アクションの実行を許可するポリシーがあるが、「アクセスが拒否されました」というメッセージが表示される .....	2922
サービスがロールのデフォルトのポリシーバージョンを作成しなかった .....	2923
コンソールにサービスロールのユースケースがない .....	2924
IAM および Amazon EC2 .....	2926
インスタンスの起動時に、Amazon EC2 コンソールの [IAM ロール] リストにあるべきロールが見当たらない .....	2926
インスタンスの認証情報が間違っただけのロールのものになっている .....	2927
AddRoleToInstanceProfile を呼び出そうとすると、AccessDenied エラーが発生する。 .....	2927
Amazon EC2: ロールを使用してインスタンスを起動しようとする、AccessDenied エラーが発生する .....	2927

EC2 インスタンスにある一時的なセキュリティ認証情報にアクセスできない .....	2928
IAM サブツリーの info ドキュメントのエラーは何を意味しますか? .....	2929
IAM および Amazon S3 .....	2930
Amazon S3 バケットへの匿名アクセスを付与する方法 .....	2930
AWS アカウントのルートユーザーとしてサインインしているが、Amazon S3 バケットに アクセスできない。 .....	2931
SAML 2.0 フェデレーション .....	2931
無効な SAML レスポンス .....	2932
RoleSessionName が必要です .....	2932
AssumeRoleWithSAML に対して承認されていません .....	2933
RoleSessionName 文字が無効です .....	2933
無効なソース ID 文字 .....	2934
レスポンスの署名が無効です .....	2934
ロールを継承できませんでした .....	2934
メタデータを解析できませんでした .....	2934
指定されたプロバイダーが存在しません。 .....	2935
DurationSeconds が MaxSessionDuration を超過しています .....	2935
レスポンスに必要なオーディエンスが含まれていません。 .....	2936
ブラウザで SAML レスポンスを表示する .....	2936
リファレンス .....	2940
Amazon リソースネーム (ARN) .....	2940
ARN 形式 .....	2940
リソースの ARN 形式を検索する .....	2942
ARN のパス .....	2942
IAM ID .....	2943
フレンドリ名とパス .....	2943
IAM ARN .....	2944
一意の識別子 .....	2951
IAM と AWS STSクォータ .....	2954
IAM 名前の要件 .....	2954
IAM オブジェクトクォータ .....	2955
IAM Access Analyzer のクォータ .....	2957
IAM Roles Anywhere クォータ .....	2957
IAM 文字制限および STS 文字制限 .....	2957
インターフェイス VPC エンドポイント .....	2962
可用性 .....	2963

AWS STS の VPC エンドポイントを作成する .....	2964
IAM と連携するサービス .....	2965
IAM と連携するサービス .....	2966
詳細情報 .....	3039
AWS API リクエストの署名 .....	3043
リクエストに署名するタイミング .....	3045
リクエストに署名する理由 .....	3045
Signature Version 4 のリクエスト要素 .....	3046
認証方法 .....	3048
署名付きリクエストを作成する .....	3052
リクエスト署名の例 .....	3064
トラブルシューティング .....	3066
ポリシーリファレンス .....	3070
JSON 要素リファレンス .....	3071
ポリシーの評価論理 .....	3142
ポリシーの文法 .....	3166
AWSジョブ機能の 管理ポリシー .....	3174
グローバル条件キー .....	3191
IAM 条件キー .....	3253
アクション、リソース、および条件キー .....	3283
リソース .....	3284
ID .....	3284
認証情報 ( パスワード、アクセスキー、MFA デバイス ) .....	3284
アクセス許可とポリシー .....	3285
フェデレーションと委任 .....	3285
IAM と他の AWS 製品 .....	3286
Amazon EC2 で IAM を使用する .....	3286
Amazon S3 での IAM の使用 .....	3286
Amazon RDS で IAM を使用する .....	3286
Amazon DynamoDB で IAM を使用する .....	3287
セキュリティに関する一般的な慣行 .....	3287
一般的なリソース .....	3287
HTTP クエリリクエストを行う .....	3289
エンドポイント .....	3289
HTTPS の必要性 .....	3290
IAM API リクエストのサインアップ .....	3290

---

ドキュメント履歴 .....	3292
----------------	------

# IAM とは

 [Follow us on Twitter](#)

---

AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスを安全に管理するためのウェブサービスです。IAM を使用すると、ユーザーがアクセスできる AWS のリソースを制御するアクセス許可を集中管理できます。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。

AWS アカウント を作成する場合、このアカウントのすべての AWS のサービス とリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウント のルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

## コンテンツ

- [IAM の紹介ビデオ](#)
- [IAM の機能](#)
- [IAM へのアクセス](#)
- [IAM をいつ使用しますか？](#)
- [IAM の仕組み](#)
- [AWS ID 管理の概要: ユーザー](#)
- [アクセス管理の概要: アクセス許可とポリシー](#)
- [AWS の ABAC とは](#)
- [IAM の外部でのセキュリティ機能](#)
- [一般的なタスクへのクイックリンク](#)
- [IAM コンソール検索](#)
- [AWS CloudFormation での AWS Identity and Access Management リソースの作成](#)
- [AWS CloudShell を使用して AWS Identity and Access Management と連携する](#)

- [AWS SDK での IAM の使用](#)

## IAM の紹介ビデオ

AWS トレーニングと認定では、IAM の概要について説明する 10 分の動画を提供しています。

### [AWS Identity and Access Management への概論](#)

## IAM の機能

IAM には、以下の機能があります。

### AWS アカウント への共有アクセス

パスワードやアクセスキーを共有しなくても、お客様の AWS アカウントのリソースを管理および使用するためのアクセス許可を他の人に付与できます。

### 詳細なアクセス権限

リソースごとに、ユーザーごとに、さまざまなアクセス権限を付与できます。例えば、一部のユーザーは、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、Amazon Redshift、他の AWS のサービスへの完全なアクセスを許可する場合があります。他のユーザーには、一部の S3 バケットへの読み取り専用アクセス、一部の EC2 インスタンスのみへの管理アクセス、または請求情報のみへのアクセスを許可できます。

### Amazon EC2 で動作するアプリケーションから AWS リソースへの安全なアクセス

IAM の機能を使用して、EC2 インスタンスで実行されているアプリケーションの認証情報を安全に提供できます。これらの認証情報は、他の AWS リソースにアクセスするためにアプリケーションのアクセス許可を提供します。例として、S3 バケットや DynamoDB テーブルなどがあります。

### 多要素認証 (MFA)

アカウントおよび個々のユーザーに 2 エlement 認証を追加することで、セキュリティを強化できます。MFA を使用すると、ユーザーはお客様のアカウントで使用しているパスワードまたはアクセスキーの入力だけでなく、特別に設定されたデバイスからのコードの入力も必要になります。既に他のサービスで FIDO セキュリティキーを使用していて、AWS がサポートする設定がある場合、MFA セキュリティに WebAuthn を使用できます。詳細については、「[パスキーとセキュリティキーを使用するためのサポートされる設定](#)」を参照してください。

## ID フェデレーション

他の場所 (社内ネットワーク、インターネット ID プロバイダーなど) でパスワードを既に持つユーザーに、お客様の AWS アカウント への一時的なアクセスを許可できます。

### 保証のための ID 情報

[AWS CloudTrail](#) を使用している場合は、お客様のアカウントのリソースをリクエストしたユーザーに関する情報がログレコードに保存されます。その情報は IAM ID に基づきます。

## PCI DSS コンプライアンス

IAM は、マーチャントまたはサービスプロバイダーによるクレジットカードデータの処理、ストレージ、および伝送をサポートしており、Payment Card Industry (PCI) Data Security Standard (DSS) に準拠していることが確認されています。PCI DSS の詳細 (AWS PCI Compliance Package のコピーをリクエストする方法など) については、「[PCI DSS レベル 1](#)」を参照してください。

## 多くの AWS サービスとの統合

IAM と連携する AWS サービスのリストについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

## 結果整合性

IAM は、他の多くの AWS サービスと同様に、[最終的に一貫性](#)があります。IAM は、世界中の Amazon のデータセンター内の複数のサーバーにデータを複製することにより、高可用性を実現します。何らかのデータの変更リクエストが正常に受け付けられると、当該変更はコミットされ、安全に保管されます。ただし、変更は IAM 全体で複製される必要があり、これには多少時間がかかることがあります。このような変更には、ユーザー、グループ、ロール、またはポリシーの作成や更新が含まれます。アプリケーションの重要で高可用性のコードパスには、このような IAM の変更を含めないことをお勧めします。代わりに、実行頻度が低い別の初期化またはセットアップルーチンに IAM の変更を加えます。また、本番稼働ワークフローが依存する前に、変更が伝達済みであることを確認します。詳細については、「[行った変更がすぐに表示されないことがある](#)」を参照してください。

## 使用料無料

AWS Identity and Access Management (IAM) および AWS Security Token Service (AWS STS) は追加料金なしで提供される AWS アカウントの機能です。IAM ユーザーまたは AWS STS の一時的なセキュリティクレデンシャルを使用して他の AWS のサービスにアクセスした場合にのみ課金されます。他の AWS 製品の料金設定については、[Amazon Web Services 料金設定ページ](#)を参照してください。

# IAM へのアクセス

次のいずれかの方法で AWS Identity and Access Management を使用できます。

## AWS Management Console

コンソールは IAM および AWS リソースを管理するためのブラウザベースのインターフェイスです。コンソールから IAM にアクセスする方法の詳細については、AWS サインイン ユーザーガイドの「[AWS へサインイン方法](#)」を参照してください。

## AWS コマンドラインツール

AWS コマンドラインツールを使用して、システムのコマンドラインでコマンドを発行することで、IAM および AWS タスクを実行できます。コマンドラインを使用すると、コンソールよりも高速で便利になります。コマンドラインツールは、AWS のタスクを実行するスクリプトを作成する場合にも便利です。

AWS には、[AWS Command Line Interface](#) AWS CLI と [AWS Tools for Windows PowerShell](#) という 2 セットのコマンドラインツールが用意されています。AWS CLI のインストールおよび使用の方法については、[AWS Command Line Interface ユーザーガイド](#) を参照してください。Tools for Windows PowerShell のインストールおよび使用の方法については、[AWS Tools for Windows PowerShell ユーザーガイド](#) を参照してください。

コンソールにサインインすると、ブラウザから AWS CloudShell を使用して CLI または SDK コマンドを実行できます。AWS リソースにアクセスするためのアクセス許可は、コンソールへのサインインに使用した認証情報に基づいています。経験によっては、CLI の方がより効率的な AWS アカウント の管理方法である場合があります。詳細については、「[AWS CloudShell を使用して AWS Identity and Access Management と連携する](#)」を参照してください。

## AWS SDK

AWS には、さまざまなプログラミング言語およびプラットフォーム (Java、Python、Ruby、.NET、iOS、Android など) のライブラリとサンプルコードで構成された SDK (ソフトウェア開発キット) が用意されています。SDK は、IAM や AWS へのプログラムによるアクセス権限を作成する際に役立ちます。例えば、SDK は要求への暗号を使用した署名、エラーの管理、要求の自動的な再試行などのタスクを処理します。AWS SDK のダウンロードやインストールなどの詳細については、「[アマゾン ウェブ サービスのツール](#)」ページを参照してください。



## IAM Query API

サービスに HTTPS リクエストを直接発行できる IAM Query API を使用して、プログラムにより IAM と AWS にアクセスできます。Query API を使用する場合は、認証情報を使用してリクエストにデジタル署名するコードを含める必要があります。詳細については、「[HTTP クエリリクエストを使用した IAM API の呼び出し](#) および [IAM API リファレンス](#)」を参照してください。

## IAM をいつ使用しますか？

### さまざまな職務を遂行しているとき

AWS Identity and Access Management は、AWS 内部の ID に基づくアクセス制御の基盤を提供するコアインフラストラクチャサービスです。AWS アカウントにアクセスするたびに IAM を使用します。

IAM の用途は、AWS で行う作業によって異なります。

- サービスユーザー – ジョブを実行するために AWS のサービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。より高度な機能を使用して仕事をすると、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。
- サービス管理者 – 社内で AWS リソースを担当している場合は、通常、IAM へのフルアクセスがあります。サービスのユーザーがどの IAM 機能やリソースにアクセスするかを決めるのは、管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。
- IAM 管理者 – IAM 管理者であれば、IAM ID を管理し、IAM へのアクセスを管理するポリシーを記述できます。

## AWS リソースへのアクセスが許可されている場合

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。

ID ソースから提供された認証情報を使用して、フェデレーテッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Center フェデレーテッドアイデンティティの例としては、(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の

認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWS にアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、「AWS サインイン User Guide」の「[How to sign in to your AWS アカウント](#)」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「[AWS API リクエストの署名](#)」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では、アカウントのセキュリティ強化のために多要素認証 (MFA) の使用をお勧めしています。詳細については、「AWS IAM Identity Center User Guide」の「[Multi-factor authentication](#)」および「IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

## IAM ユーザーとしてサインインした場合

[IAM ユーザー](#)は、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「[長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を参照してください。

[IAM グループ](#)は、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「[IAM ユーザーの作成が適している場合 \(ロールではなく\)](#)」を参照してください。

## IAM ロールを引き受けるとき

[IAM ロール](#)は、特定の権限を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。[ロールを切り替える](#) ことによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「[IAM ロールを使用する](#)」を参照してください。

一時的な認証情報を持った IAM ロールは、以下の状況で役立ちます。

- フェデレーティッドユーザーアクセス - フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「[サードパーティーアイデンティティプロバイダー向けロールの作成](#)」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[権限セット](#)」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできません。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「[IAM ロールとリソースベースのポリシーとの相違点](#)」を参照してください。
- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
- 転送アクセスセッション (FAS、Forward Access Session) - IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、あるアクションを実行することで、別のサービスの別のアクションが開始される

ことがあります。FAS は、AWS のサービス呼び出すプリンシパルのアクセス許可を使用し、リクエスト元の AWS のサービスと組み合わせて、ダウストリームサービスにリクエストを行います。FAS リクエストは、完了するために他の AWS のサービスまたはリソースとのやり取りを必要とするリクエストをサービスが受信した場合にのみ作成されます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスに権限を委任するロールの作成](#)」を参照してください。
- サービスリンクロール - サービスリンクロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウントに表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する](#)」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「[IAM ロールの作成が適している場合 \(ユーザーではなく\)](#)」を参照してください。

## ポリシーと許可を作成するとき

ポリシーを作成することで、ユーザーにアクセス許可を付与します。ポリシーは、ユーザーが実行できるアクションと、それらのアクションが影響を与えることができるリソースを登録したドキュメントです。明示的に許可されていないアクションやリソースはすべて、デフォルトで拒否されます。ポリシーは、プリンシパル (ユーザー、ユーザーグループ、ユーザーが引き受けるロール、およびリソース) に対して作成およびアタッチできます。

これらのポリシーは IAM ロールで使用されます。

- 信頼ポリシー — どの**プリンシパル**がどのような条件でロールを引き受けることができるかを定義します。信頼ポリシーは、IAM ロール用の特定のタイプのリソースベースのポリシーです。ロールが持つことが可能な信頼ポリシーは 1 つのみです。
- ID ベースのポリシー (インラインおよび管理) — これらのポリシーは、ロールのユーザーが実行できる (または実行が拒否される) 権限と、どのリソースに対して実行できるかを定義します。

[IAM アイデンティティベースのポリシーの例](#) を使用して、IAM ID のアクセス許可を定義するのに役立ちます。必要なポリシーを見つけたら、[View this policy (このポリシーを表示)] を選択してそのポリシーの JSON を表示します。JSON のポリシードキュメントをテンプレートとして使用して、独自のポリシーを作成できます。

#### Note

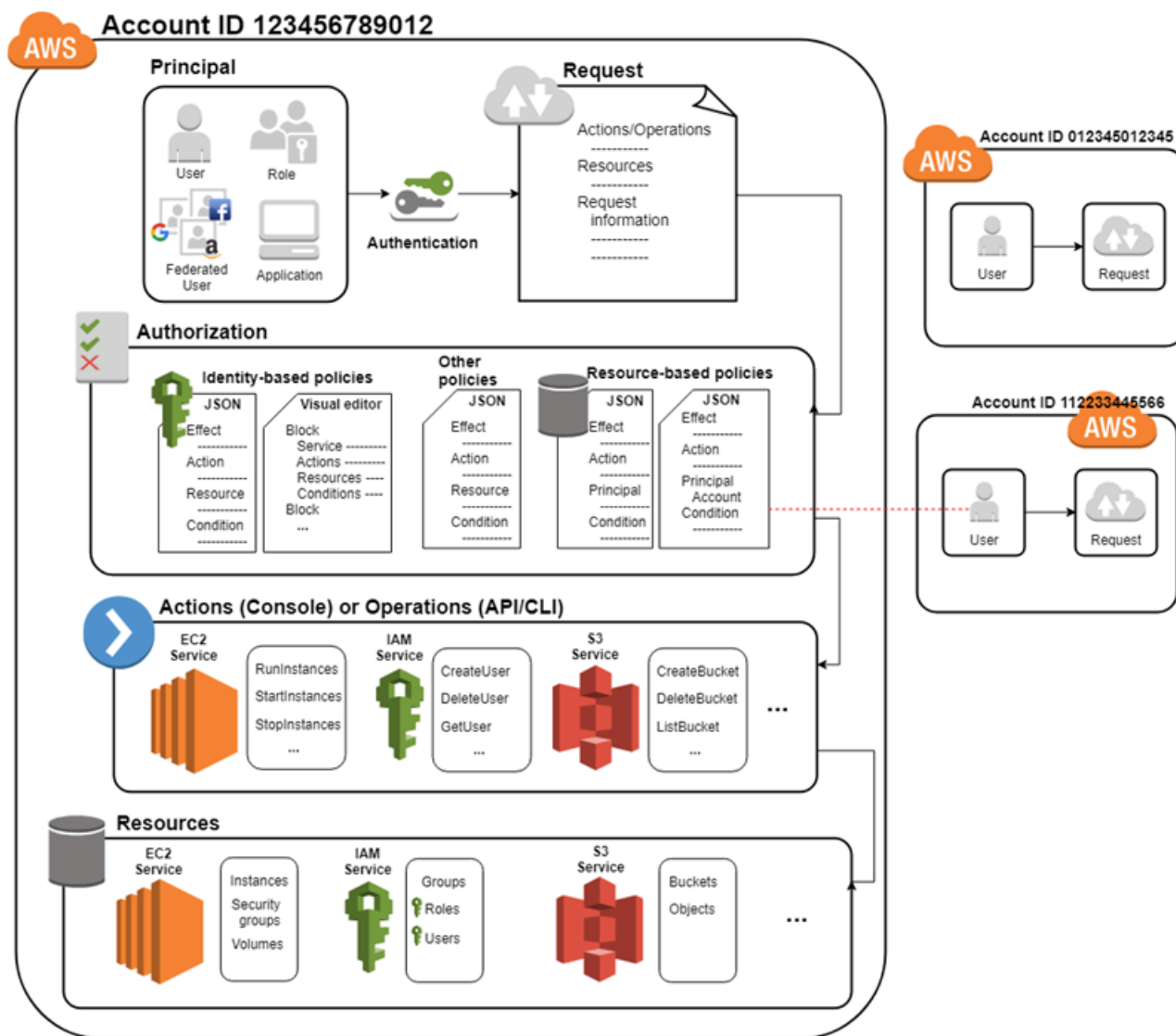
IAM Identity Center を使用してユーザーを管理している場合は、プリンシパルにアクセス許可ポリシーをアタッチするのではなく IAM ID Center で許可セットを割り当てます。アクセス許可セットをグループまたは AWS IAM アイデンティティセンターのユーザーに割り当てると、IAM Identity Center は、各アカウントに対応する IAM ロールを作成し、アクセス許可セットで指定されたポリシーをそれらのロールにアタッチします。IAM Identity Center がロールを管理し、定義した正規ユーザーがロールを引き受けることを可能にします。アクセス許可セットを変更すると、IAM Identity Center は、対応する IAM ポリシーとロールがそれに応じて更新されることを保証します。

IAM アイデンティティセンターの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM アイデンティティセンターとは) を参照してください。

## IAM の仕組み

IAM は、AWS アカウントの認証と認可を制御するために必要なインフラストラクチャを提供します。IAM のインフラストラクチャーは、次の図で示されています。





まず、人間のユーザーまたはアプリケーションがサインイン認証情報を使用して AWS と認証します。認証は、サインイン認証情報を AWS アカウント が信頼するプリンシパル (IAM ユーザー、フェデレーションユーザー、IAM ロール、またはアプリケーション) と照合することによって行われます。

次に、プリンシパルにリソースへのアクセスを許可するリクエストが行われます。アクセスは、承認リクエストに応じて許可されます。例えば、コンソールに初めてサインインしてコンソールのホームページを開いたときは、特定のサービスにアクセスしてはなりません。サービスを選択すると、承認リクエストがそのサービスに送信され、ユーザーの ID が認証されたユーザーのリストに含まれているかどうか、付与されるアクセスレベルを制御するためにどのようなポリシーが適用されているか、そして、その他の有効なポリシーがないかが確認されます。承認リクエストは、AWS アカウント 内または信頼できる別の AWS アカウント プリンシパルが行うことができます。

承認されると、プリンシパルはユーザー内の AWS アカウント リソースに対してアクションを実行したり、操作を実行したりできます。例えば、プリンシパルは新しい Amazon Elastic Compute Cloud インスタンスを起動したり、IAM グループメンバーシップを変更したり、Amazon Simple Storage Service バケットを削除したりできます。

## 基本概念

- [規約](#)
- [Principal](#)
- [リクエスト](#)
- [認証](#)
- [認可](#)
- [アクションまたはオペレーション](#)
- [リソース](#)

## 規約

これらの IAM 用語は、AWS と連携する際によく使用されます。

### IAM リソース

IAM リソースは IAM に保存されます。IAM から追加、編集、削除できます。

- user
- グループ
- ロール
- ポリシー
- ID プロバイダーオブジェクト

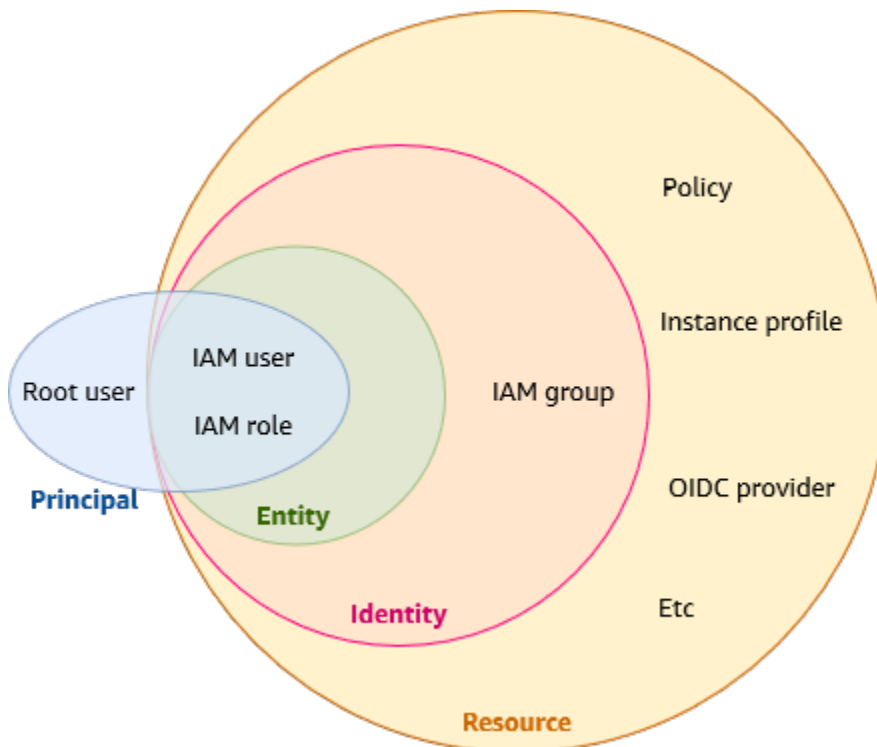
### IAM エンティティ

AWS が認証に使用する IAM リソース。エンティティはリソースベースのポリシーでプリンシパルとして指定できます。

- user
- ロール

## IAM アイデンティティ

アクションの実行とリソースへのアクセスを行うためにポリシーで承認できる IAM リソース。IAM アイデンティティには、ユーザー、グループ、およびロールがあります。



### プリンシパル

AWS アカウントのルートユーザー、IAM ユーザー、または IAM ロールを使用してサインインし、AWS へのリクエストを行う人またはアプリケーション。プリンシパルには、フェデレーションユーザーと引き受けたロールが含まれます。

### 人間のユーザー

人間 ID と呼ばれ、人、管理者、デベロッパー、オペレーター、およびアプリケーションのコンシューマーを指します。

### ワークロード

アプリケーションやバックエンドプロセスなど、ビジネス価値を提供するリソースやコードの集合体のことです。アプリケーション、運用ツール、およびコンポーネントが含まれることがあります。



## Principal

プリンシパルとは、AWS リソースに対するアクションまたはオペレーションをリクエストできる人間の ID またはワークロードです。認証後、プリンシパルには、プリンシパルのタイプに応じて、AWS にリクエストを行うための永続的または一時的な認証情報を付与できます。IAM ユーザーとルートユーザーには永続的な認証情報が付与され、ロールには一時的な認証情報が付与されます。[ベストプラクティス](#)として、人間のユーザーとワークロードに一時的な認証情報を使用して AWS のリソースにアクセスさせることをお勧めします。

## リクエスト

プリンシパルが AWS Management Console、AWS API、または AWS CLI を使用しようとする、プリンシパルは AWS にリクエストを送信します。リクエストには、以下の情報が含まれます。

- アクションまたはオペレーション – プリンシパルが実行するアクションまたはオペレーション。AWS Management Console ではアクション、AWS CLI や AWS API ではオペレーションです。
- リソース – アクションまたはオペレーションを実行する対象の AWS リソースオブジェクト。
- プリンシパル – エンティティ (ユーザーまたはロール) を使用してリクエストを送信するユーザーまたはアプリケーション。プリンシパルに関する情報には、プリンシパルがサインインに使用したエンティティに関連付けられたポリシーが含まれます。
- 環境データ – IP アドレス、ユーザーエージェント、SSL 有効化ステータス、または時刻に関する情報。
- リソースデータ – リクエストされているリソースに関連するデータ。これには、DynamoDB テーブル名、Amazon EC2 インスタンスのタグなどの情報が含まれる場合があります。

AWS は、リクエスト情報をリクエストコンテキスト内に収集し、リクエストの評価と承認に使用します。

## 認証

プリンシパルとして、AWS に認証情報を使用してリクエストを送信するには、認証されている必要があります (AWS にサインイン)。Amazon S3 や AWS STS などの一部のサービスは、匿名ユーザーからの少数のリクエストを許可します。ただし、これは例外的です。

コンソールからルートユーザーとして認証するには、E メールアドレスとパスワードでサインインする必要があります。フェデレーションユーザーとして、ID プロバイダーによって認証され、IAM

ロールを引き継ぐことで AWS リソースへのアクセス権が付与されます。IAM ユーザーとして、アカウント ID またはエイリアスを入力してから、ユーザー名とパスワードを入力します。API または AWS CLI からワークロードを認証するには、ロールを割り当てて一時的な認証情報を使用することも、アクセスキーとシークレットキーを指定して長期的な認証情報を使用することもできます。また、追加のセキュリティ情報を提供する必要があります。ベストプラクティスとして、AWS では、アカウントのセキュリティを強化するために多要素認証 (MFA) と一時的な認証情報を使用することをお勧めしています。AWS が認証できる IAM エンティティの詳細については、「[IAM ユーザー](#)」および「[IAM ロール](#)」を参照してください。

## 認可

リクエスト完了を認可される (許可される) 必要があります。認可の際に AWS はリクエストコンテキストの値に基づいて、リクエストに適用されるポリシーを確認します。次に、ポリシーを使用してリクエストの許可または拒否を決定します。通常、ポリシーは [JSON ドキュメント](#) として AWS に保存され、プリンシパルエンティティのアクセス許可を指定します。[複数のポリシータイプ](#) がリクエストの承認/却下に影響する場合があります。ユーザーに許可したアクセス許可でユーザー自身のアカウント内の AWS リソースにアクセスする場合は、アイデンティティベースのポリシーのみが必要です。リソースベースのポリシーは、一般的に [クロスアカウントアクセス](#) を許可するために使用します。他のポリシータイプはアドバンスド機能であり、慎重に使用する必要があります。

AWS は、リクエストのコンテキストに該当する各ポリシーをチェックします。1 つのアクセス許可ポリシーに拒否されたアクションが含まれている場合、AWS はリクエスト全体を拒否し、評価を停止します。このプロセスは明示的な拒否と呼ばれています。リクエストはデフォルトで拒否されるため、AWS では、リクエストのすべての部分が該当するアクセス許可ポリシーによって許可された場合に限り、リクエストを承認します。単一アカウント内のリクエストの評価ロジックは、以下の一般的なルールに基づきます。

- デフォルトでは、すべてのリクエストが拒否されます。(通常、AWS アカウントのルートユーザー認証情報を使用したアカウントのリソースに対するリクエストは常に許可されます)。
- アクセス許可ポリシー (アイデンティティベースまたはリソースベース) に明示的な許可が含まれている場合、このデフォルト設定は上書きされます。
- Organizations SCP、IAM アクセス許可の境界、またはセッションポリシーがある場合、その許可は上書きされます。上記のポリシータイプが 1 つ以上存在する場合は、そのすべてにおいてリクエストが許可されている必要があります。それ以外の場合、リクエストは暗黙的に拒否されます。
- ポリシー内の明示的な拒否は、すべての許可に優先します。

すべてのタイプのポリシーの評価方法については、「[ポリシーの評価論理](#)」を参照してください。別のアカウントでリクエストを作成する必要がある場合は、他のアカウントのポリシーでリソースへのアクセスを許可する必要があります。また、リクエストを作成するために使用する IAM エンティティには、リクエストを許可するアイデンティティベースのポリシーが必要です。

## アクションまたはオペレーション

リクエストが認証および認可されると、AWS はリクエストのアクションまたはオペレーションを承認します。オペレーションはサービスによって定義されます。オペレーションは、リソースの表示、作成、編集、削除など、リソースに対して実行できる処理です。例えば、IAM は、ユーザーリソースに対して、以下のアクションを含む約 40 のアクションをサポートしています。

- CreateUser
- DeleteUser
- GetUser
- UpdateUser

オペレーションを実行することをプリンシパルに許可するには、プリンシパルまたは対象のリソースに適用されるポリシーに、必要なアクションを含める必要があります。各サービスでサポートされているアクション、リソースタイプ、および条件キーのリストについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## リソース

AWS で承認されたリクエスト内のオペレーションは、アカウント内の関連リソースに対して実行できます。リソースは、サービス内に存在するオブジェクトです。例として、Amazon EC2 インスタンス、IAM ユーザー、Amazon S3 バケットなどがあります。このサービスは、各リソースで実行できる一連のアクションを定義します。関連していないアクションをリソースに対して実行するリクエストを作成すると、そのリクエストは拒否されます。例えば、IAM ロールの削除をリクエストしても IAM グループリソースを指定すると、そのリクエストは失敗します。アクションの影響を受けるリソースを示す AWS のサービス別の表については、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## AWS ID 管理の概要: ユーザー

AWS アカウント への特定のユーザーにアクセス権を付与し、そのユーザーに AWS アカウント 内のリソースにアクセスするための特定の権限を与えることができます。IAM と AWS IAM Identity

Center の両方を使用して、新しいユーザーを作成したり、既存のユーザーを AWS にフェデレートしたりできます。この 2 つの主な違いは、IAM ユーザーには AWS リソースへの長期認証情報が付与されるのに対し、IAM Identity Center のユーザーにはユーザーが AWS にサインインするたびに確立される一時的な認証情報が付与されることです。[ベストプラクティス](#)として、一時的な認証情報の使用により AWS にアクセスするには、人間のユーザーに対して IAM ユーザーではなく ID プロバイダーとのフェデレーションの使用を必須とします。IAM ユーザーは主に、IAM ロールを使用できないワークロードに AWS のサービスに対して API または CLI を使用したプログラムによるリクエストを可能にする場合に使用します。

## トピック

- [初回アクセスのみ: ルートユーザーの認証情報](#)
- [IAM ユーザーと IAM Identity Center 内のユーザー](#)
- [既存のユーザーのフェデレーション](#)
- [アクセスコントロール方法](#)

## 初回アクセスのみ: ルートユーザーの認証情報

AWS アカウント を作成する場合、このアカウントのすべての AWS のサービス とリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。ルートユーザーに付与されるアクセス許可を制限できるのは、組織内のサービスコントロールポリシー (SCP) のみです。

## IAM ユーザーと IAM Identity Center 内のユーザー

IAM ユーザーは個別のアカウントではなく、アカウント内のユーザーです。各ユーザーには、AWS Management Console にアクセスするための、独自のパスワードを割り当てることができます。また各ユーザーには、お客様のアカウントのリソースをプログラムによりリクエストできるように、個別のアクセスキーを作成できます。

IAM ユーザーには、AWS リソースへの長期的な認証情報が付与されます。ベストプラクティスとして、ヒューマンユーザー用の長期認証情報を使用して IAM ユーザーを作成しないでください。代わりに、人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用が必要です。

**Note**

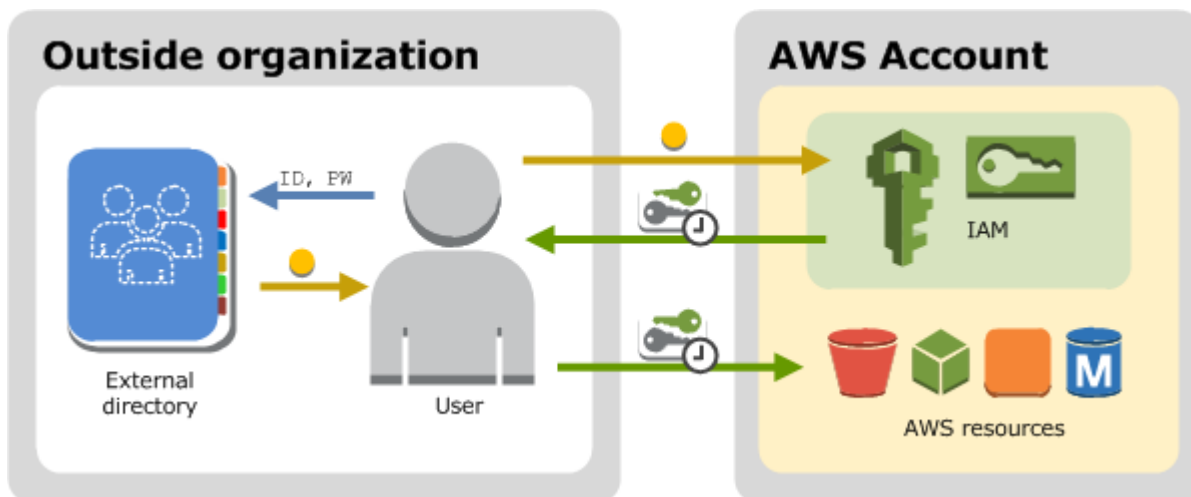
プログラムによるアクセスや長期的な認証情報を持つ IAM ユーザーが必要なシナリオでは、必要な時にアクセスキーを更新することをお勧めします。詳細については、「[アクセスキーの更新](#)」を参照してください。

対照的に、AWS IAM アイデンティティセンターのユーザーには、AWS リソースに短期間の認証が付与されます。一元的なアクセス管理を行うには、[AWS IAM Identity Center \(IAM Identity Center\)](#) を使用して、ご自分のアカウントへのアクセスと、それらのアカウント内でのアクセス許可を管理することをお勧めします。IAM Identity Center は、デフォルトの ID ソースとして Identity Center ディレクトリで自動的に設定され、ユーザーやグループを作成し、AWS リソースへのアクセスレベルを割り当てることができます。詳細については、[AWS IAM Identity Center ユーザーガイド](#)の AWS IAM Identity Center とは を参照してください。

## 既存のユーザーのフェデレーション

組織内のユーザーが、企業ネットワークにサインインするなどして認証される方法を既に持っている場合、個別の IAM ユーザーまたは IAM Identity Center でそれらのユーザーを作成する必要はありません。代わりに、IAM または AWS IAM Identity Center のいずれかを使用して、これらのユーザー ID を AWS にフェデレートすることができます。

以下の図では、ユーザーが一時的に AWS セキュリティ認証情報を取得する方法を示しています。この情報により、ユーザーは AWS アカウントのリソースにアクセスできるようになります。



フェデレーションは以下の場合に特に便利です。

- ユーザーがすでに社内ディレクトリに所属している。

社内ディレクトリが Security Assertion Markup Language 2.0 (SAML 2.0) と互換性がある場合は、ユーザーに AWS Management Console へのシングルサインオン (SSO) アクセスを許可するように、社内ディレクトリを設定できます。詳細については、「[一時的な認証情報の一般的なシナリオ](#)」を参照してください。

社内ディレクトリが SAML 2.0 と互換性がない場合は、ユーザーに AWS Management Console へのシングルサインオン (SSO) アクセスを許可するために、ID ブローカーアプリケーションを作成できます。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

社内ディレクトリが Microsoft Active Directory である場合は、AWS IAM Identity Center を使用して、Active Directory の自己管理型ディレクトリや「[AWS Directory Service](#)」のディレクトリを接続し、社内ディレクトリと AWS アカウント との間で信頼関係を確立できます。

Okta や Microsoft Entra などの外部 ID プロバイダー (IdP) を使用してユーザーを管理している場合は、AWS IAM Identity Center を使用して IdP と AWS アカウント 間の信頼を確立できます。詳細については、AWS IAM Identity Center ユーザーガイドの「[外部の ID プロバイダーに接続](#)」を参照してください。

- ユーザーがすでにインターネットの ID を所有している。

作成するモバイルアプリケーションやウェブベースのアプリケーションで、Login with Amazon、Facebook、Google などの OpenID Connect (OIDC) 互換 ID プロバイダーのようなインターネット ID プロバイダーから、ユーザーがログインして認証されるようにする場合、アプリケーションからフェデレーションを使用して AWS へのアクセスが可能になります。詳細については、「[OIDC フェデレーション](#)」を参照してください。

#### Tip

インターネット ID プロバイダーによる ID フェデレーションを使用するには、[Amazon Cognito](#) を使用することをお勧めします。

## アクセスコントロール方法

ここでは、AWS リソースへのアクセスをコントロールする方法について説明します。



ユーザーアクセスのタイプ	なぜ、使用するのか？	どこで、詳細情報を取得できるか？
<p>IAM Identity Center を使用した、従業員など人間のユーザーによる、AWS リソースに対するシングルサインオンでのアクセス</p>	<p>IAM Identity Center はユーザーと AWS アカウントへのアクセス、クラウドアプリケーションの管理を一元化する場所を提供します。</p> <p>IAM Identity Center 内に ID ストアを設定することも、既存の ID プロバイダー (IdP) とのフェデレーションを設定することもできます。セキュリティ上のベストプラクティスとしては、人間のユーザーに付与する AWS リソースに対する認証情報は、必要に応じて限定されたものであることが推奨されます。</p> <p>これにより、ユーザーのサインインが簡素化されるとともに、リソースへのアクセスの管理を単一のシステムから実行できるようになります。IAM Identity Center では、追加のアカウントセキュリティとして多要素認証 (MFA) もサポートしています。</p>	<p>IAM Identity Center の設定の詳細については、「AWS IAM Identity Centerユーザーガイド」の「<a href="#">Getting Started</a>」(使用開始)を参照してください。</p> <p>IAM Identity Center での MFA の使用方法については、「AWS IAM Identity Center ユーザーガイド」の「<a href="#">Multi-factor authentication</a>」(多要素認証)を参照してください。</p>
<p>IAM ID プロバイダーを使用した、従業員など人間のユーザーによる、AWS</p>	<p>IAMは、OpenID Connect (OIDC) または SAML 2.0 (Security Assertion Markup Language 2.0) と互換性のある IdP をサポートします。IAM ID プロバイダーを作成し</p>	<p>IAM ID プロバイダーおよびフェデレーションの詳細については、「<a href="#">ID プロバイダーとフェデレーション</a>」を参照してください。</p>

ユーザーアクセスのタイプ	なぜ、使用するのか？	どこで、詳細情報を取得できるか？
サービスに対するフェデレーションされたアクセス	<p>たら、フェデレーションユーザーに動的割り当てが可能な、1つ以上の IAM ロールを作成する必要があります。</p>	
AWS アカウント間でのクロスアカウントアクセス	<p>特定の AWS リソースへのアクセスを、他の AWS アカウントのユーザーと共有したい場合があります。</p> <p>クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の AWS サービスでは、リソースにポリシーを直接アタッチすることができます (ロールをプロキシとして使用する代わりに)。これらはリソースベースのポリシーと呼ばれます。</p>	<p>IAM ロールの詳細については、「<a href="#">IAM ロール</a>」を参照してください。</p> <p>サービスにリンクされたロールの詳細については、「<a href="#">サービスリンクロールの使用</a>」を参照してください。</p> <p>サービスにリンクされたロールの使用をサービスでサポートするには、「<a href="#">IAM と連携する AWS のサービス</a>」を参照してください。[サービスにリンクされたロール] 列が「はい」になっているサービスを見つけます。サービスにリンクされたロールのドキュメントをサービスで表示するには、その列の「はい」に関連するリンクを選択します。</p>



ユーザーアクセスのタイプ	なぜ、使用するのか？	どこで、詳細情報を取得できるか？
AWS アカウントの専有 IAM ユーザー向けの長期的な認証情報	<p>特定のユースケースでは、AWS の IAM ユーザーに長期的な認証情報が必要となることがあります。これらの IAM ユーザーを AWS アカウントに作成するために IAM を使用し、また、そのユーザーのアクセス許可を IAM により管理できます。ユースケースには次のようなものがあります。</p> <ul style="list-style-type: none"> <li>• IAM ロールを使用できないワークロード</li> <li>• アクセスキーを介したプログラムによるアクセスを必要とするサードパーティーの AWS クライアント</li> <li>• AWS CodeCommit または Amazon Keyspaces サービス固有の認証情報</li> <li>• アカウントで AWS IAM Identity Center を利用できず、他に ID プロバイダーがない場合</li> </ul> <p><u>ベストプラクティス</u>として、<u>プログラムによるアクセスと長期的な認証情報</u>を持つ IAM ユーザーが必要なシナリオでは、必要な時にアクセスキーを更新することをお勧めします。詳細については、</p>	<p>どこで、詳細情報を取得できるか？</p> <p>IAM ユーザー設定の詳細については、「<a href="#">AWS アカウントでの IAM ユーザーの作成</a>」を参照してください。</p> <p>IAM ユーザーのアクセスキーの詳細については、「<a href="#">IAM ユーザーのアクセスキーの管理</a>」を参照してください。</p> <p>AWS CodeCommit または Amazon Keyspaces サービス固有の認証情報の詳細については、「<a href="#">CodeCommit での IAM の使用: Git 認証情報、SSH キー、および AWS アクセスキー</a>」および「<a href="#">Amazon Keyspaces での IAM の使用 ( Apache Cassandra 用 )</a>」を参照してください。</p>

ユーザーアクセスのタイプ	なぜ、使用するのか？	どこで、詳細情報を取得できるか？
	「 <a href="#">アクセスキーの更新</a> 」を参照してください。	

## アクセス管理の概要: アクセス許可とポリシー

AWS Identity and Access Management (IAM) のアクセス管理では、アカウントでプリンシパルエンティティに許可する操作を定義します。プリンシパルエンティティとは、IAM エンティティ (ユーザーまたはロール) を使用して認証されているユーザーまたはアプリケーションを指します。アクセス管理は、認可と呼ばれることもあります。AWS でのアクセスを管理するには、ポリシーを作成し、IAM アイデンティティ (ユーザー、ユーザーのグループ、ロール) または AWS リソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパルが IAM エンティティ (ユーザーまたはロール) を使用してリクエストを行うと、それらのポリシーを評価します。ポリシーでの許可により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。ポリシーのタイプと用途の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

### ポリシーとアカウント

AWS で 1 つのアカウントを管理する場合は、ポリシーを使用してそのアカウント内でアクセス許可を定義します。複数のアカウントをまたいでアクセス許可を管理する場合、ユーザーのアクセス許可の管理はより難しくなります。IAM ロール、リソースベースのポリシー、アクセスコントロールリスト (ACL) はクロスアカウントのアクセス許可で使用できます。ただし、複数のアカウントを所有している場合は、このようなアクセス許可を管理しやすいように AWS Organizations サービスを使用することをお勧めします。詳細については、「[Organizations ユーザーガイド](#)」の「AWS Organizations とは」を参照してください。

### ポリシーとユーザー

IAM ユーザーはサービスのアイデンティティです。IAM ユーザーを作成する場合、ユーザーはアクセス許可を付与されるまでアカウントのいずれのリソースにもアクセスできません。アクセス許可をユーザーに付与するには、アイデンティティベースのポリシーを作成し、ユーザー、またはユーザーが属するグループにアタッチします。以下の例で示す JSON ポリシーでは、`dynamodb:*` リージョ

ン内の Books アカウントの 123456789012 テーブルに対してすべての Amazon DynamoDB アクション (us-east-2) を実行することをユーザーに許可します。

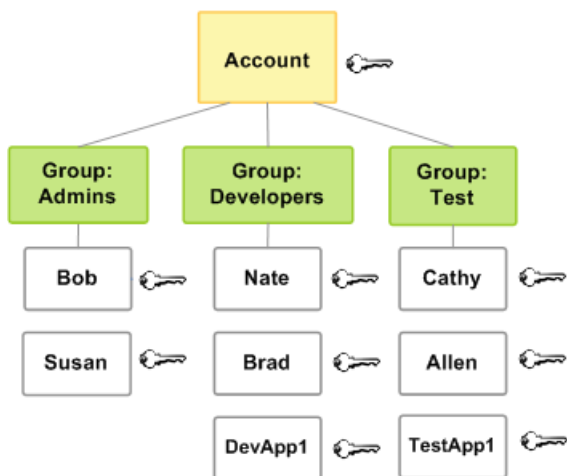
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

このポリシーを IAM ユーザーにアタッチすると、ユーザーの許可はこれらの DynamoDB 許可のみになります。大半のユーザーは複数のポリシーを持ち、これらが一体となってユーザーのアクセス許可を構成します。

明示的に許可されていないアクションやリソースは、デフォルトで拒否されます。たとえば、ユーザーにアタッチされているポリシーが、前述のポリシーのみである場合、ユーザーは Books テーブルに対してのみ DynamoDB アクションを実行できます。他のすべてのテーブルでのアクションは禁止されます。同様に、ユーザーは Amazon EC2、Amazon S3、またはその他の AWS のサービスで一切のアクションを実行できません。これらのサービスを使用するアクセス許可がユーザーのポリシーに含まれていないためです。

## ポリシーとグループ

ユーザーを IAM グループにまとめ、そのグループにポリシーをアタッチできます。その場合、個々のユーザーは、まだ独自の認証情報を持っていますが、そのグループのすべてのユーザーは、グループにアタッチされているアクセス許可を持っています。アクセス許可の管理を簡素化するため、また「[IAM でのセキュリティのベストプラクティス](#)」に従うために、グループを使用します。



ユーザーまたはグループには、さまざまなアクセス許可を付与する複数のポリシーをアタッチできます。その場合、ユーザーのアクセス許可はポリシーの組み合わせに基づいて評価されます。この場合もやはり、基本的な原則が適用されます。ユーザーにアクションとリソースへの明示的にアクセス許可が付与されていない場合、そのユーザーにはそれらのアクセス許可はありません。

## フェデレーテッドユーザーとロール

フェデレーションユーザーは、IAM ユーザーとは異なり、AWS アカウントに永続的な ID を持っていません。フェデレーテッドユーザーにアクセス許可を割り当てるには、ロールと呼ばれるエンティティを作成し、ロールのアクセス許可を定義できます。フェデレーテッドユーザーが AWS にサインインすると、そのユーザーはロールに関連付けられ、ロールで定義されているアクセス許可が付与されます。詳細については、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

## アイデンティティベースのポリシーとリソースベースのポリシー

アイデンティティベースのポリシーは、IAM アイデンティティ (IAM ユーザー、グループ、ロールなど) にアタッチするアクセス許可ポリシーです。リソースベースのポリシーは、Amazon S3 バケットなどのリソースまたは IAM ロール信頼ポリシーにアタッチする許可ポリシーです。

アイデンティティベースのポリシーでは、アイデンティティが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーはさらに次のように分類できます。

- 管理ポリシー - AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンのアイデンティティベースのポリシーです。次の 2 種類の管理ポリシーを使用できます。

- **AWS 管理ポリシー** – AWS が作成および管理する管理ポリシー。ポリシーを初めて利用する場合は、AWS 管理ポリシーから開始することをお勧めします。
- **カスタマー管理ポリシー** – AWS アカウント で作成および管理する管理ポリシー。カスタマー管理ポリシーでは、AWS 管理ポリシーに比べて、より正確にポリシーを管理できます。ビジュアルエディタで および IAM ポリシーを作成および編集することも、JSON ポリシードキュメントを直接作成することもできます。詳細については、[IAM ポリシーの作成](#) および [IAM ポリシーの編集](#) を参照してください。
- **インラインポリシー** – お客様が作成および管理するポリシーであり、単一のユーザー、グループ、またはロールに直接埋め込まれます。通常、インラインポリシーを使用することは推奨されていません。

リソースベースのポリシーでは、指定されたプリンシパルが実行できるリソースと条件を制御します。リソースベースのポリシーはインラインポリシーであり、管理リソースベースのポリシーはありません。クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。

IAM サービスは、ロールの信頼ポリシーと呼ばれるリソースベースのポリシーのタイプを 1 つのみサポートします。これが、IAM ロールにアタッチされています。IAM ロールは、リソースベースのポリシーをサポートするアイデンティティかつリソースであるため、信頼ポリシーとアイデンティティベースのポリシーのいずれも IAM ロールにアタッチする必要があります。信頼ポリシーでは、ロールを引き受けることができるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーテッドユーザー) を定義します。IAM ロールと、他のリソースベースのポリシーとの違いについては、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

リソースベースのポリシーをサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。リソースベースのポリシーの詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。

## AWS の ABAC とは

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。これらの属性は、AWS でタグと呼ばれています。タグは、IAM エンティティ (ユーザーまたはロール) を含めた IAM リソース、および AWS リソースにアタッチできます。IAM プリンシパルに対して、単一の ABAC ポリシー、または少数のポリシーのセットを作成できます。これらの ABAC ポリシーは、プリンシパルのタグがリソースタグと一致するときに操作を許可するように設計することができます。ATBAC は、急成長する環境や、ポリシー管理が煩雑になる状況で役に立ちます。

例えば、`access-project` タグキーを使用して 3 つのロールを作成できます。最初のロールのタグ値を `Heart`、2 番目を `Star`、3 番目を `Lightning` に設定します。そうすることで、`access-project` に関してロールとリソースが同じ値でタグ付けされているときにアクセスを許可する単一のポリシーを使用できます。AWS の ABAC の使用方法を説明する詳細なチュートリアルについては、「[IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#)」を参照してください。ABACをサポートするサービスについては、[IAM と連携する AWS のサービス](#) を参照してください。

## ABAC と従来の RBAC モデルの比較

IAM で使用される従来の認証モデルは、ロールベースのアクセスコントロール (RBAC) と呼ばれます。RBAC は、AWS の外部でロールとして知られる、ユーザーの職務機能に基づいてアクセス許可を定義します。AWS 内では、ロールは通常 IAM ロールを指します。IAM ロールは、引き受けることができる IAM の ID です。IAM には、RBACモデルのジョブ機能にアクセス許可を調整する[ジョブ機能の管理ポリシー](#)が含まれています。

IAM では、職務機能ごとに異なるポリシーを作成して RBAC を実装します。次に、ポリシーを ID (IAM ユーザー、ユーザーのグループ、または IAM ロール) にアタッチします。[ベストプラクティス](#)として、職務機能に必要な最小限のアクセス許可を付与します。これは、[最小アクセス許可の付与](#)と呼ばれます。これを行うには、職務機能がアクセスできる特定のリソースを一覧表示します。従来の RBAC モデルを使用することの欠点は、従業員が新しいリソースを追加するときに、それらのリソースへのアクセスを許可するようにポリシーを更新する必要があることです。

たとえば、従業員が取り組んでいる `Lightning`、`Heart`、`Star`、という名前の 3 つのプロジェクトがあるとします。各プロジェクトの IAM ロールを作成します。次に、ポリシーを各 IAM ロールにアタッチして、ロールを引き受けることができるすべてのユーザーがアクセスできるリソースを定義します。従業員が社内でジョブを変更した場合は、別の IAM ロールに割り当てます。ユーザーまたはプログラムは複数のロールに割り当てることができます。ただし、`Star` プロジェクトでは、新しい Amazon EC2 コンテナなどの追加のリソースが必要になる場合があります。その場合は、`Star` ロールにアタッチされたポリシーを更新して、新しいコンテナリソースを指定する必要があります。そうしないと、`Star` プロジェクトメンバーは新しいコンテナにアクセスできません。

ABAC には、従来の RBAC モデルと比べて以下の利点があります。

- ABAC のアクセス許可は、イノベーションに合わせてスケールされます。新しいリソースへのアクセスを許可するために、管理者が既存のポリシーを更新する必要はありません。たとえば、`access-project` タグを使用して ABAC 戦略を設計したとします。開発者は、`access-project = Heart` タグでロールを使用します。`Heart` プロジェクトのユーザーが追加の Amazon



EC2 リソースを必要とする場合、開発者は `access-project = Heart` タグを使用して新しい Amazon EC2 instances インスタンスを作成できます。その後は、タグ値が一致するため、Heart プロジェクトのすべてのユーザーがこれらのインスタンスを起動および停止できます。

- ABAC では必要なポリシーが少なくなります。職務機能ごとに異なるポリシーを作成する必要がないため、作成するポリシーは少なくなります。これらのポリシーは管理しやすくなります。
- ABAC を使用することで、チームは迅速に変化し、成長することができます。これは、新しいリソースの許可が属性に基づいて自動的に付与されるためです。たとえば、会社が ABAC を使用して Heart プロジェクトと Star プロジェクトをすでにサポートしている場合、新しい Lightning プロジェクトは簡単に追加できます。IAM 管理者は、`access-project = Lightning` タグを使用して新しいロールを作成します。新しいプロジェクトをサポートするためにポリシーを変更する必要はありません。ロールを引き受けるアクセス許可を持っているユーザーは、`access-project = Lightning` でタグ付けされたインスタンスを作成および表示できます。また、チームメンバーが Heart プロジェクトから Lightning プロジェクトに移動する場合があります。管理者は、ユーザーを別の IAM ロールに割り当てます。アクセス許可ポリシーを変更する必要はありません。
- ABAC を使用すると、きめ細かなアクセス許可が可能です。ポリシーを作成するときは、[最小限のアクセス許可を付与](#)することがベストプラクティスです。従来の RBAC では、特定のリソースへのアクセスのみを許可するポリシーを記述する必要があります。ただし、ABAC を使用する場合、リソースタグがプリンシパルのタグと一致する場合のみ、すべてのリソースでアクションを許可できます。
- ABAC を使用して、社内ディレクトリの従業員属性を使用します。セッションタグを AWS に渡すよう SAML または OIDC プロバイダーを設定できます。従業員が AWS にフェデレートすると、その属性が AWS で結果のプリンシパルに適用されます。その後、ABAC を使用して、これらの属性に基づいてアクセス許可を許可または拒否できます。

AWS の ABAC の使用方法を説明する詳細なチュートリアルについては、「[IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#)」を参照してください。

## IAM の外部でのセキュリティ機能

IAM を使用して、AWS Management Console、[AWS コマンドラインツール](#)、または [AWS SDK](#) を使用するサービス API オペレーションで実行されるタスクへのアクセスをコントロールします。一部の AWS 製品には、それらのリソースを保護する他の方法も用意されています。以下のリストでは、いくつかの例を示していますが、すべての機能を挙げてはおりません。

## Amazon EC2

Amazon Elastic Compute Cloud では、キーペア (Linux インスタンスの場合)、またはユーザー名とパスワード (Microsoft Windows インスタンスの場合) を使用して、インスタンスにログインします。

詳細については、次のドキュメントを参照してください。

- 「Amazon EC2 ユーザーガイド」の「[Amazon EC2 Linux インスタンスの開始方法](#)」
- 「Amazon EC2 ユーザーガイド」の「[Amazon EC2 Windows インスタンスの開始方法](#)」

## Amazon RDS

Amazon Relational Database Service では、そのデータベースに関連付けられているユーザー名とパスワードを使用して、データベースエンジンにログインします。

詳細については、Amazon RDS ユーザーガイドの「[Amazon RDS の開始方法](#)」を参照してください。

## Amazon EC2 および Amazon RDS 1 x

Amazon EC2 および Amazon RDSでは、セキュリティグループを使用して、インスタンスまたはデータベースへのトラフィックをコントロールします。

詳細については、次のドキュメントを参照してください。

- 「Amazon EC2 ユーザーガイド」の「[Linux インスタンス用の Amazon EC2 セキュリティグループ](#)」
- 「Amazon EC2 ユーザーガイド」の「[Windows インスタンス用の Amazon EC2 セキュリティグループ](#)」
- [Amazon RDS ユーザーガイド](#) の「Amazon RDS セキュリティグループ」

## WorkSpaces

Amazon WorkSpaces では、ユーザーはユーザー名とパスワードを使用して、デスクトップにサインインします。

詳細については、[Amazon WorkSpaces 管理ガイド](#)の「Amazon WorkSpaces の開始方法」を参照してください。

## Amazon WorkDocs

Amazon WorkDocs では、ユーザーはユーザー名とパスワードを使用してサインインすることで、共有ドキュメントへのアクセスを取得します。



Amazon WorkDocs の詳細については、Amazon WorkDocs 管理ガイドの「[Amazon WorkDocs の開始方法](#)」を参照してください。

これらのアクセスコントロール方法は IAM の一部ではありません。IAM では、これらのAWS 製品の管理に含まれるのは、Amazon EC2 インスタンスの作成または終了、新しい WorkSpaces デスクトップの設定などです。確かに IAM は、Amazon Web Services にリクエストして実行されるタスクをコントロールするためにも、AWS Management Console へのアクセスをコントロールするためにも役立ちます。しかし IAM は、オペレーティングシステム (Amazon EC2)、データベース (Amazon RDS)、デスクトップ (Amazon WorkSpaces)、またはコラボレーションサイト (Amazon WorkDocs) にサインインするようなタスクに対するセキュリティの管理には役立ちません。

特定の AWS 製品を使用する場合、その製品に属するすべてのリソースのセキュリティオプションについては、その製品のドキュメントを参照してください。

## 一般的なタスクへのクイックリンク

以下のリンクを使用して、IAM に関連する一般的なタスクのヘルプを参照します。

### さまざまなユーザータイプのサインイン

[IAM user] (IAM ユーザー) を選択し、AWS アカウント ID またはアカウントエイリアスを入力して [IAM コンソール](#) にサインインします。その次のページで、IAM ユーザー名とパスワードを入力します。

IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM アイデンティティセンターのユーザーを使用してサインインする方法については、「AWS サインイン User Guide」の「[Signing in to the AWS access portal](#)」を参照してください。

[ルートユーザー] を選択し、AWS アカウント のメールアドレスを入力して、アカウント所有者として [AWS Management Console](#) にサインインします。次のページでパスワードを入力します。

ユーザータイプとサインインページの決定については、「AWS サインイン ユーザーガイド」の「[AWS サインインとは](#)」を参照してください。

### ユーザーのパスワードを管理する

請求情報へのアクセスを含め、AWS Management Console にアクセスするには、パスワードが必要です。

AWS アカウントのルートユーザー については、「AWS Account Management リファレンスガイド」の「[AWS アカウントのルートユーザー のパスワードを変更する](#)」を参照してください。

IAM ユーザーについては、「[IAM ユーザーのパスワードの管理](#)」を参照してください。

## ユーザーのアクセス許可を管理する

ポリシーを使用して、お客様の AWS アカウント の IAM ユーザーにアクセス許可を付与します。IAM ユーザーには、作成時にアクセス許可がないため、AWS リソースの使用を許可するためのアクセス許可を付与する必要があります。

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center のユーザーとグループ:

アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可一式を作成](#)」の手順を実行します。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「[サードパーティ ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

- IAM ユーザー:

- ユーザーに設定できるロールを作成します。手順については、「IAM ユーザーガイド」の「[IAM ユーザー用ロールの作成](#)」を参照してください。

- (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。「IAM ユーザーガイド」の「[ユーザー \(コンソール\) へのアクセス許可の追加](#)」の指示に従います。

詳細については、「[IAM ポリシーを管理する](#)」を参照してください。

お客様の AWS アカウント のユーザーを表示し、それらのユーザーの認証情報について情報を取得する

「[AWS アカウント の認証情報レポートの取得](#)」を参照してください。

## 多要素認証 (MFA) を追加する

仮想 MFA デバイスを追加するには、以下のいずれかを参照してください。

- [AWS アカウントのルートユーザー \(コンソール\) の仮想 MFA デバイスを有効にします](#)
- [IAM ユーザーの仮想 MFA デバイスの有効化 \(コンソール\)](#)

FIDO セキュリティキーを追加するには、以下のいずれかを参照してください。

- [AWS アカウントのルートユーザー のパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)
- [別の IAM ユーザーのパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)

ハードウェア MFA デバイスを追加するには、以下のいずれかを参照してください。

- [AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする \(コンソール\)](#).
- [別の IAM ユーザーのハードウェア TOTP トークンを有効にする \(コンソール\)](#)

## アクセスキーを取得する

アクセスキーを使用すると、[AWS SDK](#)、[AWS コマンドラインツール](#)、または API オペレーションを使用して AWS リクエストを実行できます。

### Important

[ベストプラクティス](#)は、アクセスキーのような長期的認証情報を作成するのではなく、IAM ロールなどの一時的なセキュリティ認証情報を使用することです。アクセスキーを作成する前に、[長期的なアクセスキーの代替案](#)を確認してください。

アクセスキーを保護するのに役立つガイダンスについては、「[アクセスキーの保護](#)」を参照してください。

IAM ユーザーのアクセスキーの管理については、「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

AWS アカウント で使用できるセキュリティ認証情報の詳細については、「[AWS セキュリティ認証情報](#)」を参照してください。

## IAM リソースのタグ付け

次の IAM リソースにタグ付けができます。

- IAM ユーザー
- IAM ロール
- カスタマー管理ポリシー
- ID プロバイダ

- サーバー証明書
- 仮想 MFA デバイス

IAM のタグの詳細については、「[IAM リソースのタグ付け](#)」を参照してください。

タグを使用して AWS リソースへのアクセスを制御する方法については、「[タグを使用した AWS リソースへのアクセスの制御](#)」を参照してください。

すべてのサービスのアクション、リソース、条件キーを表示します

この一連のリファレンスドキュメントは、詳細な IAM ポリシーの作成に役立ちます。各 AWS サービスは、IAM ポリシーで使用するアクション、リソース、および条件コンテキストキーを定義します。詳細については、「[AWS サービスのアクション、リソース、および条件キー](#)」を参照してください。

AWS をすべて開始する

このドキュメントでは、主に IAM サービスを取り扱います。AWS の開始方法と、複数のサービスを使用して問題 (最初のプロジェクトの構築および起動) を解決する方法については、「[開始方法リソースセンター](#)」を参照してください。

## IAM コンソール検索

IAM コンソールの検索ページは、IAM リソースを見つけるためのより速いオプションとして使用します。コンソールの検索を使用して、アカウントに関連付けられたアクセスキー、IAM エンティティ (ユーザー、グループ、ロール、ID プロバイダーなど)、名前などでポリシーなどを見つけることができます。

IAM コンソールの検索機能では、以下のいずれかを見つけることができます。

- 検索キーワードに一致する IAM エンティティ名 (ユーザー、グループ、ロール、ID プロバイダー、およびポリシーが対象)
- 検索キーワードに一致するタスク

IAM コンソール検索機能では、IAM Access に関する情報は返されません。

検索結果のすべての行は、有効なリンクです。たとえば、検索結果でユーザー名を選択すると、ユーザーの詳細ページに移動することができます。または、[ユーザーの作成] などのアクションリンクを選択して、[ユーザーの作成] ページに移動できます。

**Note**

アクセスキーの検索では、検索ボックスに完全なアクセスキー ID を入力する必要があります。検索結果には、そのキーに関連付けられたユーザーが表示されます。ここからは、そのユーザーのページに直接移動してユーザーのアクセスキーを管理することができます。

## IAM コンソール検索の使用

IAM コンソールの [Search (検索)] ページを使用して、そのアカウントに関連する項目を見つけます。



IAM コンソールで項目を検索するには

1. AWS サインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. ナビゲーションペインで、[検索] を選択します。
4. [検索] ボックスに検索キーワードを入力します。
5. 検索結果リストのリンクを選択すると、コンソールの該当する部分に移動します。

## IAM コンソール検索結果のアイコン

次のアイコンにより、検索によって見つかった項目のタイプがわかります。

アイコン	説明
	IAM ユーザー
	IAM グループ
	IAM ロール
	IAM ポリシー

アイコン	説明
	"ユーザーの作成"、"ポリシーのアタッチ"などのタスク
	キーワード <b>delete</b> の結果

## サンプルの検索語句

IAM 検索で次の語句を使用できます。斜体の用語は、検索する実際の IAM ユーザー、グループ、ロール、アクセスキー、ポリシー、または ID プロバイダーの名前と置き換えます。

- ***user\_name*** または ***group\_name*** または ***role\_name*** または ***policy\_name*** または ***identity\_provider\_name***
- ***access\_key***
- **add user *user\_name* to groups**、または **add users to group *group\_name***
- **remove user *user\_name* from groups**
- **delete *user\_name*** または **delete *group\_name***、あるいは **delete *role\_name***、あるいは **delete *policy\_name***、あるいは **delete *identity\_provider\_name***
- **manage access keys *user\_name***
- **manage signing certificates *user\_name***
- **users**
- **manage MFA for *user\_name***
- **manage password for *user\_name***
- **create role**
- **password policy**
- **edit trust policy for role *role\_name***
- **show policy document for role *role\_name***
- **attach policy to *role\_name***
- **create managed policy**
- **create user**

- **create group**
- **attach policy to *group\_name***
- **attach entities to *policy\_name***
- **detach entities from *policy\_name***

## AWS CloudFormation での AWS Identity and Access Management リソースの作成

AWS Identity and Access Management は、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスである AWS CloudFormation と統合されています。必要なすべての AWS リソース (アクセスキー、グループ、グループポリシー、インスタンスプロファイル、管理ポリシー、OIDC プロバイダー、インラインポリシー、ロール、ロールポリシー、SAML プロバイダー、サーバー証明書、サービスにリンクされたロール、ユーザー (およびグループへのユーザーの追加)、ユーザーポリシー、仮想 MFA デバイスなど) を記述するテンプレートを作成すると、AWS CloudFormation はユーザーに代わって、これらのリソースをプロビジョニングして設定します。

AWS CloudFormation を使用すると、テンプレートを再利用して IAM リソースをいつでも繰り返しセットアップできます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウントとリージョンで何度でもプロビジョニングできます。

### IAM および AWS CloudFormation テンプレート

IAM および関連サービスのリソースをプロビジョニングして設定するには、[AWS CloudFormation](#) テンプレートについて理解しておく必要があります。テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートには、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation デザイナー を使えば、AWS CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS CloudFormation デザイナー とは](#)」を参照してください。

IAM は、アクセスキー、グループ、グループポリシー、インスタンスプロファイル、管理ポリシー、OIDC プロバイダー、インラインポリシー、ロール、ロールポリシー、SAML プロバイダー、サーバー証明書、サービスにリンクされたロール、ユーザー (およびグループへのユーザーの追加)、ユーザーポリシー、および仮想 MFA デバイスを AWS CloudFormation で作成することをサポートしています。IAM リソースの JSON テンプレートと YAML テンプレートの例を含む詳細情報について



は、「AWS CloudFormation ユーザーガイド」の「[AWS Identity and Access Management リソースタイプのリファレンス](#)」を参照してください。

ロールや管理ポリシーなどの関連リソースを作成するテンプレートを作成することもできます。

## AWS CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、以下のリソースを参照してください。

- [AWS CloudFormation](#)
- [AWS CloudFormation ユーザーガイド](#)
- [AWS CloudFormation API リファレンス](#)
- [AWS CloudFormation コマンドラインインターフェイスユーザーガイド](#)

## AWS CloudShell を使用して AWS Identity and Access Management と連携する

AWS CloudShell はブラウザベースの事前に認証されたシェルで、AWS Management Console から直接起動できます。任意のシェル (Bash、PowerShell、Z シェルなど) を使用して、AWS サービス (AWS Identity および Access Management を含む) に対して AWS CLI コマンドを実行できます。この手順は、コマンドラインツールのダウンロードもインストールも不要です。

[AWS Management Console から AWS CloudShell を起動](#)すると、コンソールへのサインインに使用した AWS 認証情報は、新しいシェルセッションで自動的に利用できます。AWS CloudShell ユーザーのこの事前認証により、AWS CLI バージョン 2 (シェルのコンピューティング環境にプリインストール済み) を使用している IAM など AWS のサービスとやり取りするときに、認証情報の設定をスキップできます。

## AWS CloudShell の IAM アクセス許可の取得

AWS Identity and Access Management 管理者によって提供されるアクセス管理リソースを使用して、管理者は IAM ユーザーが AWS CloudShell にアクセスして環境の機能を使用できるアクセス許可を付与します。

管理者がユーザーにアクセス権を付与する最も簡単な方法は、AWS マネージドポリシーを介した方法です。[AWS マネージドポリシー](#)は、AWS が作成および管理するスタンドアロンポリシーです。CloudShell 用に次の AWS マネージドポリシーを IAM アイデンティティにアタッチできます。



- `AWSCloudShellFullAccess`: すべての機能へのフルアクセス権のある AWS CloudShell を使用するためのアクセス許可を付与します。

IAM ユーザーが AWS CloudShell を使用して実行できるアクションの範囲を制限する場合、テンプレートとして `AWSCloudShellFullAccess` マネージドポリシーを使用するカスタムポリシーを作成できます。CloudShell でユーザーが使用できるアクションを制限する方法の詳細については、「AWS CloudShell ユーザーガイド」の「[IAM ポリシーを使用して AWS CloudShell へのアクセスと使用を管理する](#)」を参照してください。

## AWS CloudShell を使用して IAM とやりとりする

AWS Management Console から AWS CloudShell を起動すると、コマンドラインインターフェイスを使用して IAM との通信をすぐに開始できます。

### Note

AWS CloudShell で AWS CLI を使用する場合、追加のリソースをダウンロードまたはインストールする必要はありません。さらに、ユーザーはシェル内で既に認証されているので、呼び出しを行う前に認証情報を設定する必要はありません。

AWS CloudShell を使用して IAM グループを作成し、IAM ユーザーをグループに追加します。

次の例では、CloudShell を使用して IAM グループを作成し、IAM ユーザーをグループに追加して、コマンドが成功したことを確認します。

1. AWS Management Console から、ナビゲーションバーに表示される次のオプションを選択することで CloudShell を起動できます。
  - CloudShell アイコンを選択します。
  - 検索ボックスに「cloudshell」を入力し始めてから [CloudShell] オプションを選択します。
2. IAM グループを作成するには、CloudShell コマンドラインで次のコマンドを入力します。この例では、グループに「east\_coast」という名前を付けました。

```
aws iam create-group --group-name east_coast
```

コールが成功すると、コマンドラインに次の出力に似たサービスからのレスポンスが表示されません。

```
{
  "Group": {
    "Path": "/",
    "GroupName": "east_coast",
    "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
    "Arn": "arn:aws:iam::111122223333:group/east_coast",
    "CreateDate": "2023-09-11T21:02:21+00:00"
  }
}
```

- 作成したグループにユーザーを追加するには、以下のコマンドを使用してグループ名とユーザー名を指定します。この例では、グループに「east\_coast」、ユーザーに「johndoe」という名前を付けています。

```
aws iam add-user-to-group --group-name east_coast --user-name johndoe
```

- ユーザーがグループに属していることを確認するには、以下のコマンドを使用してグループ名を指定します。この例では、引き続き east\_coast というグループを使用します。

```
aws iam get-group --group-name east_coast
```

コールが成功すると、コマンドラインに次の出力に似たサービスからのレスポンスが表示されます。

```
{
  "Users": [
    {
      "Path": "/",
      "UserName": "johndoe",
      "UserId": "AIDAYBDBW4JBXGEXAMPLE",
      "Arn": "arn:aws:iam::552108220995:user/johndoe",
      "CreateDate": "2023-09-11T20:43:14+00:00",
      "PasswordLastUsed": "2023-09-11T20:59:14+00:00"
    }
  ],
  "Group": {
    "Path": "/",
    "GroupName": "east_coast",
```

```
"GroupId": "AGPAYBDBW4JBY3EXAMPLE",  
"Arn": "arn:aws:iam::111122223333:group/east_coast",  
"CreateDate": "2023-09-11T21:02:21+00:00"  
}  
}
```


## AWS SDK での IAM の使用

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
<a href="#">AWS SDK for C++</a>	<a href="#">AWS SDK for C++ コードの例</a>
<a href="#">AWS CLI</a>	<a href="#">AWS CLI コードの例</a>
<a href="#">AWS SDK for Go</a>	<a href="#">AWS SDK for Go コードの例</a>
<a href="#">AWS SDK for Java</a>	<a href="#">AWS SDK for Java コードの例</a>
<a href="#">AWS SDK for JavaScript</a>	<a href="#">AWS SDK for JavaScript コードの例</a>
<a href="#">AWS SDK for Kotlin</a>	<a href="#">AWS SDK for Kotlin コードの例</a>
<a href="#">AWS SDK for .NET</a>	<a href="#">AWS SDK for .NET コードの例</a>
<a href="#">AWS SDK for PHP</a>	<a href="#">AWS SDK for PHP コードの例</a>
<a href="#">AWS Tools for PowerShell</a>	<a href="#">Tools for PowerShell のコード例</a>
<a href="#">AWS SDK for Python (Boto3)</a>	<a href="#">AWS SDK for Python (Boto3) コードの例</a>
<a href="#">AWS SDK for Ruby</a>	<a href="#">AWS SDK for Ruby コードの例</a>
<a href="#">AWS SDK for Rust</a>	<a href="#">AWS SDK for Rust コードの例</a>
<a href="#">AWS SDK for SAP ABAP</a>	<a href="#">AWS SDK for SAP ABAP コードの例</a>

SDK ドキュメント	コードの例
<a href="#">AWS SDK for Swift</a>	<a href="#">AWS SDK for Swift コードの例</a>

IAM に固有の例については、「[AWS SDK を使用した IAM のコード例](#)」を参照してください。

 可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

# IAM の準備作業

## Important

IAM [ベストプラクティス](#)では、長期的な認証情報を持つIAMユーザーを使用するのではなく、一時的な認証情報を使用して AWS にアクセスするために、IDプロバイダーとのフェデレーションを使用することを人間ユーザーに求めることを推奨します。

AWS Identity and Access Management (IAM) により、Amazon Web Services (AWS) およびアカウントリソースへのアクセスを安全に管理することができます。IAM では、サインイン認証情報をプライベートに保持することもできます。IAM を使用するため、特別にサインアップしません。IAM の使用は無料です。

ユーザーやロールなどの ID に対し、アカウントのリソースに対するアクセス権を付与するには、IAM を使用します。例えば、AWS の外部で管理している社内ディレクトリの既存のユーザーに対し IAM を使用することや、AWS IAM Identity Center を使用して AWS 内にユーザーを作成することが考えられます。フェデレーション ID は、定義済の IAM ロールを引き受け、必要なリソースにアクセスします。IAM アイデンティティセンターの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM アイデンティティセンターとは)を参照してください。

## Note

IAM は複数の AWS 製品と統合されています。IAM をサポートするサービスのリストについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

## トピック

- [AWS アカウントへのサインアップ](#)
- [管理アクセス権を持つユーザーを作成する](#)
- [最小特権アクセス許可に備える](#)
- [IAM 管理メソッド](#)
- [AWS アカウント ID とそのエイリアス](#)

# AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

AWS アカウントにサインアップすると、AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべてのAWS のサービスとリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して[ルートユーザーアクセスが必要なタスク](#)を実行してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。<https://aws.amazon.com/> の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

## 管理アクセス権を持つユーザーを作成する

AWS アカウントにサインアップしたら、AWS アカウントのルートユーザーをセキュリティで保護し、AWS IAM Identity Center を有効にして、管理ユーザーを作成します。これにより、日常的なタスクにルートユーザーを使用しないようにします。

AWS アカウントのルートユーザーをセキュリティで保護する

1. [ルートユーザー] を選択し、AWS アカウントのメールアドレスを入力して、アカウント所有者として [AWS Management Console](#) にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとしてサインインする](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAM ユーザーガイドの「[AWS アカウントのルートユーザーの仮想 MFA デバイスを有効にする \(コンソール\)](#)」を参照してください。

## 管理アクセス権を持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[AWS IAM Identity Center の有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセス権を付与します。

IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「[デフォルト IAM アイデンティティセンターディレクトリを使用したユーザーアクセスの設定](#)」を参照してください。

## 管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[AWS アクセスポータルにサインインする](#)」を参照してください。

## 追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

「AWS IAM Identity Center ユーザーガイド」の「[Create a permission set](#)」の手順に従ってください。

2. ユーザーをグループに割り当ててから、そのグループにシングルサインオンアクセスを割り当てます。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[Add groups](#)」を参照してください。

## 最小特権アクセス許可に備える

最小特権のアクセス許可の使用は、IAM のベストプラクティスの推奨事項です。最小特権アクセス許可のコンセプトは、タスクを実行するにあたり必要となるアクセス権限のみをユーザーに付与することです。設定したら、最小特権アクセス許可をどのようにサポートするかを検討してください。



ルートユーザーと管理者ユーザーの両方に、日常のタスクには必要ない強力なアクセス許可があります。AWS について学び、さまざまなサービスをテストしている間は、IAM Identity Center で、異なるシナリオで使用できる、より少ないアクセス許可を持つ少なくとも 1 人の追加ユーザーを作成することをお勧めします。IAM ポリシーを使用して、特定の条件下で特定のリソースに対して実行できるアクションを定義し、より少ない特権アカウントでそれらのリソースに接続することができます。

IAM Identity Center を使用している場合は、IAM Identity Center の許可セットを使用して開始することを検討してください。詳細については、IAM Identity Center ユーザーガイドの「[権限セットの作成](#)」を参照してください。

IAM Identity Center を使用しない場合は、IAM ロールを使用してさまざまな IAM エンティティにアクセス許可を定義します。詳細については、「[IAM ロールの作成](#)」を参照してください。

IAM ロールと IAM Identity Center 許可セットはどちらも、職務に基づく AWS 管理ポリシーを使用できます。これらのポリシーによって付与される許可の詳細については、「[AWS ジョブ機能の管理ポリシー](#)」を参照してください。

#### Important

AWS 管理ポリシーは、すべての AWS のユーザーが使用できるため、特定のユースケースに対して最小特権のアクセス許可が付与されない場合があることに留意してください。セットアップが完了したら、IAM Access Analyzer を使用して、AWS CloudTrail に記録しているアクセスアクティビティに基づいて、最小特権ポリシーを生成することをお勧めします。ポリシー生成の詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

## IAM 管理メソッド

IAM は、AWS コンソール、AWS コマンドラインインターフェイス、または関連する SDK のアプリケーションインターフェイス (API) のいずれかを使用して管理できます。セットアップ時には、どの方法をサポートし、さまざまなユーザーをどのようにサポートするかを検討してください。

### トピック

- [AWS コンソール](#)
- [AWS コマンドラインインターフェイス \(CLI\) と Software Development Kit \(SDK\)](#)

## AWS コンソール

AWS 管理コンソールは、AWS リソースを管理するためのサービスコンソールの広範なコレクションで構成され、そのコレクションを参照するウェブアプリケーションです。最初にサインインすると、コンソールのホームページが表示されます。各サービスコンソールにアクセスできるホームページは、AWS に関連するタスクを実行するための情報にアクセスするための単一の場所として機能します。コンソールにサインインした後に、どのサービスとアプリケーションを利用できるようになるかは、アクセス権限を持っている AWS リソースによって異なります。リソースへのアクセス許可は、ロールを引き受けるか、権限を付与されたグループのメンバーになるか、明示的に権限を付与されるかのいずれかによって付与されます。スタンドアロン AWS アカウントでは、ルートユーザーまたは IAM 管理者がリソースへのアクセスを設定します。AWS Organizations では、管理アカウントまたは委任管理者がリソースへのアクセスを設定します。

AWS Management Console を使用して AWS リソースを管理する予定の場合は、セキュリティ上の [ベストプラクティス](#) として、一時的な認証情報を使用してユーザーを設定することをおすすめします。ロールを引き受けた IAM ユーザー、フェデレーションユーザー、IAM Identity Center のユーザーには一時的な認証情報があり、IAM ユーザーとルートユーザーには長期的な認証情報があります。ルートユーザーの認証情報は AWS アカウント へのフルアクセスを提供し、他のユーザーは IAM ポリシーによって付与されたリソースへのアクセスを提供する認証情報を持っています。

サインインの操作は、AWS Management Console ユーザーのタイプによって異なります。

- IAM ユーザーとルートユーザーは、メインの AWS サインイン URL (<https://signin.aws.amazon.com>) からサインインします。サインインすると、権限が付与されているアカウントのリソースにアクセスできるようになります。

ルートユーザーとしてサインインするには、ルートユーザーのメールアドレスとパスワードが必要です。

IAM ユーザーとしてサインインするには、AWS アカウント 番号またはエイリアス、IAM ユーザー名、IAM ユーザーパスワードが必要です。

アカウント内の IAM ユーザーは、緊急アクセスなど、長期的な認証情報を必要とする特定の状況に限定し、ルートユーザーは [ルートユーザーの認証情報を必要とするタスク](#) にのみ使用することをおすすめします。

利便性のため、AWS サインインページはブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。次回、ユーザーが任意のページに移動したとき、AWS Management

Console コンソールは Cookie を使用して、ユーザーをアカウントのサインインページにリダイレクトします。

セッションが終了したらコンソールからサインアウトして、前回のサインインが再利用されないようにします。

- IAM Identity Center ユーザーは、組織固有の特定の AWS アクセスポータルを使用してサインインします。サインインすると、アクセスするアカウントまたはアプリケーションを選択できます。アカウントにアクセスする場合は、管理セッションに使用するアクセス許可セットを選択します。
- 外部 ID プロバイダーで管理されているフェデレーションユーザーは、カスタムのエンタープライズアクセスポータルを使用して AWS アカウント サインインにリンクされています。フェデレーションユーザーが利用できる AWS リソースは、組織が選択したポリシーによって異なります。

#### Note

セキュリティレベルを高めるために、ルートユーザー、IAM ユーザー、IAM Identity Center のユーザーは、AWS リソースへのアクセスを許可する前に AWS による多要素認証 (MFA) の検証を受けることができます。MFA が有効になっている場合は、サインインするために MFA デバイスへのアクセス権も必要となります。

さまざまなユーザーが管理コンソールにサインインする方法については、AWS サインインユーザーガイドの「[AWS 管理コンソールへのサインイン](#)」を参照してください。

## AWS コマンドラインインターフェイス (CLI) と Software Development Kit (SDK)

IAM Identity Center と IAM ユーザーは、CLI または関連する SDK のアプリケーションインターフェイス (API) を使用して認証を行う場合、異なる方法を使用して認証情報を認証します。

認証情報と構成設定は、システム環境変数、ユーザー環境変数、ローカルの AWS 設定ファイルなど複数の場所にあり、コマンドラインでパラメータとして明示的に宣言される場合もあります。特定の場所が他の場所よりも優先されます。

IAM Identity Center と IAM はどちらも CLI または SDK で使用できるアクセスキーを提供します。IAM Identity Center アクセスキーは、自動的に更新できる一時的な認証情報であり、IAM ユーザーに関連する長期的なアクセスキーよりも推奨されます。

CLI または SDK を使用して AWS アカウント を管理するには、ブラウザから AWS CloudShell を使用できます。CloudShell を使用して CLI または SDK コマンドを実行する場合は、まずコンソールにサインインする必要があります。AWS リソースにアクセスするためのアクセス許可は、コンソールへのサインインに使用した認証情報に基づいています。経験によっては、CLI の方がより効率的な AWS アカウント の管理方法である場合があります。

アプリケーション開発では、CLI または SDK をコンピューターにダウンロードし、コマンドプロンプトまたは Docker ウィンドウからサインインできます。このシナリオでは、CLI スクリプトまたは SDK アプリケーションの一部として認証情報とアクセス認証情報を設定します。環境と利用可能なアクセス許可に応じて、リソースへのプログラムによるアクセスはさまざまな方法で設定できます。

- AWS サービスを使用してローカルコードを認証するための推奨オプションは、IAM ID センターと IAM Roles Anywhere です。
- AWS 環境内で実行されるコードを認証するための推奨オプションは、IAM ロールを使用するか、IAM Identity Center 認証情報を使用することです。

IAM Identity Center を使用している場合は、AWS アクセス許可セットを選択するアクセスポータルからスタートページから短期的な認証情報を取得できます。これらの認証情報には有効期限が定義されており、自動的に更新されることはありません。これらの認証情報を使用する場合は、AWS ポータルにサインインした後に、AWS アカウント を選択し、権限セットを選択します。[コマンドラインまたはプログラムによるアクセス] を選択すると、プログラムまたは CLI から AWS リソースにアクセスするために使用できるオプションが表示されます。これらの方法の詳細については、IAM Identity Center ユーザーガイドの「[一時的な認証情報の取得と更新](#)」を参照してください。これらの認証情報は、アプリケーションの開発中にコードをすばやくテストするためによく使用されます。

AWS リソースへのアクセスを自動化すると自動的に更新される IAM Identity Center 認証情報を使用することをお勧めします。IAM Identity Center でユーザーと権限セットを設定している場合、`aws configure sso` コマンドを使ってコマンドラインウィザードを使用すると、利用可能な認証情報を特定してプロファイルに保存できます。プロファイルの設定について詳しくは、バージョン 2 用 AWS コマンドラインインターフェイスユーザーガイドの「[aws configure sso ウィザードによるプロファイルの設定](#)」を参照してください。

#### Note

多くのサンプルアプリケーションでは、IAM ユーザーまたはルートユーザーに関連する長期アクセスキーを使用しています。長期的な認証情報は、学習演習の一環としてサンドボックス環境内でのみ使用してください。[長期的なアクセスキーの代替手段](#)を確認し、コードをできるだけ早く移行することで、IAM Identity Center 認証情報や IAM ロールなどの代替認証情

報を使用するようにすることを計画してください。コードを移行したら、アクセスキーを削除します。

CLI の設定の詳細については、AWS コマンドラインインターフェイス (バージョン 2) ユーザーガイドの「[AWS CLI の最新バージョンをインストールまたは更新する](#)」および、AWS コマンドラインインターフェイスユーザーガイドの「[認証とアクセス認証情報](#)」を参照してください。

SDK の設定について詳しくは、AWS SDK とツールリファレンスガイドの「[IAM Identity Center 認証](#)」および、AWS SDK とツールリファレンスガイドの「[IAM Roles Anywhere](#)」を参照してください。

## AWS アカウント ID とそのエイリアス

アカウントの IAM ユーザーは、アカウントエイリアスまたはアカウント ID を含むウェブ URL でサインインします。URL がない場合は、AWS サインインページで AWS アカウント エイリアスまたはアカウント ID を指定する必要があります。

アカウント ID またはエイリアスがわからない場合:

- ブラウザの履歴を確認してください。以前にログインしたことがある場合は、最近アクセスしたウェブサイトに保存されている可能性があります。
- アカウント認証情報で AWS CLI または AWSSDK を構成済みの場合は、設定ファイルからアカウント ID を取得できます。
- ローカル管理者またはアカウント所有者に問い合わせてください。AWS はアカウント ID をユーザーに提供することはできません。

### Tip

ウェブブラウザでアカウントのサインインページのブックマークを作成するには、サインイン URL を手動でブックマークエントリに入力する必要があります。Web ブラウザの「このページをブックマーク」機能は使用しないでください。この機能を使用すると、現在のブラウザセッション固有の情報がキャプチャされ、今後のログインページへのアクセスが妨げられます。

## トピック

- [AWS アカウント ID を表示する](#)
- [アカウントのエイリアスについて](#)
- [AWS アカウント のエイリアスの作成、削除および一覧表示](#)

## AWS アカウント ID を表示する

AWS アカウント のアカウント ID は、以下の方法で確認できます。

### コンソールを使用したアカウント ID の確認

アカウント ID は、AWS アカウントセクションの IAM ダッシュボードに表示されます。ユーザータイプにより、コンソールでアカウント ID を表示する別の方法があります。既にロールを引き受けている場合、[セキュリティ認証情報] は使用できません。

ユーザーのタイプ	手順
ルートユーザー	右上のナビゲーションバーでユーザー名を選択した後、[セキュリティ認証情報] を選択します。アカウント番号は [アカウント識別子] の下に表示されます。
IAM ユーザー	右上のナビゲーションバーでユーザー名を選択すると、アカウント ID がユーザー名の上に表示されます。[Security credentials] (セキュリティ認証情報) を選択します。アカウント番号は [アカウント詳細] の下に表示されます。
フェデレーテッドユーザー	右上のナビゲーションバーでユーザー名を選択すると、アカウント ID がユーザー名の上に表示されます。
役割を引き受けた	右上のナビゲーションバーで [サポート] アイコンを選択したら、リストから [サポートセンター] を選択します。現在サインインしている 12 桁のアカウント番号 (ID) は、サポートセンターナビゲーションペインを使用します。



## AWS CLI を使用したアカウント ID の確認

次のコマンドを使用して、ユーザー ID、アカウント ID、およびユーザー ARN を表示します。

- [aws sts get-caller-identity](#)

## API を使用したアカウント ID の確認

次の API を使用して、ユーザー ID、アカウント ID、およびユーザー ARN を表示します。

- [GetCallerIdentity](#)

## アカウントのエイリアスについて

サインインページの URL に AWS アカウント ID の代わりに自社名 (または他のわかりやすい識別子) を含める場合は、アカウントのエイリアスを作成できます。このセクションでは、AWS アカウント エイリアスについて説明し、エイリアスの作成に使用する API オペレーションのリストを示します。

サインインページのデフォルトの URL は以下の形式です。

```
https://Your_Account_ID.signin.aws.amazon.com/console/
```

AWS アカウント ID の AWS アカウント エイリアスを作成すると、サインインページの URL は次の例のようになります。

```
https://Your_Account_Alias.signin.aws.amazon.com/console/
```

### 考慮事項

- AWS アカウント で使用できるエイリアスは 1 つのみです。AWS アカウントの新しいエイリアスを作成すると、新しいエイリアスによって以前のエイリアスが上書きされ、以前のエイリアスを含む URL は機能しなくなります。
- アカウントエイリアスは、数字、小文字、ハイフンのみです。AWS アカウントのエンティティの制限に関する詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。
- アカウントエイリアスは、指定されたネットワークパーティション内のすべての Amazon Web Services 製品で一貫した必要があります。

パーティションは AWS リージョンのグループです。各 AWS アカウントのスコープは 1 つのパーティションです。

サポートされているパーティションは以下のとおりです。

- aws - AWS リージョン
- aws-cn - 中国リージョン
- aws-us-gov - AWS GovCloud (US) リージョン

## AWS アカウント のエイリアスの作成、削除および一覧表示

AWS Management Console、IAM API、またはコマンドラインインターフェイスを使用し、AWS アカウント の別名を作成または削除することができます。

### Note

アカウントエイリアスはシークレットではなく、公開されているサインインページの URL に表示されます。アカウントのエイリアスには、機密情報を含めないでください。AWS アカウント ID を含む本来の URL は、AWS アカウント エイリアスの作成後も有効です。

## アカウントエイリアスを作成または編集する (コンソール)

アカウントエイリアスは、AWS Management Console で作成、編集、および削除できます。

### 最小アクセス許可

次の手順を実行するには、少なくとも以下の IAM アクセス許可が必要です。

- iam:ListAccountAliases
- iam:CreateAccountAlias

## アカウントエイリアスを作成または編集するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。



2. ナビゲーションペインで、ダッシュボード を選択します。
3. [AWS アカウント] セクションで、[アカウントエイリアス] を選択し、[作成] を選択します。エイリアスが既に存在する場合は、[Edit] (編集) を選択します。
4. ダイアログボックス内でエイリアスの名前を入力し、[変更の保存] を選択します。

#### Note

一度に AWS アカウント に関連付けできるエイリアスは 1 つのみです。新しいエイリアスを作成すると、以前のエイリアスが削除され、以前のエイリアスに関連付けられているサインイン URL は機能しなくなります。

## アカウントエイリアスを削除する (コンソール)

アカウントエイリアスは、AWS Management Console で削除できます。

#### 最小アクセス許可

次の手順を実行するには、少なくとも以下の IAM アクセス許可が必要です。

- iam:ListAccountAliases
- iam:CreateAccountAlias
- iam>DeleteAccountAlias

## アカウントエイリアスを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、ダッシュボード を選択します。
3. [アカウントエイリアス] の横にある [AWS アカウント] セクションで、[削除] を選択します。

#### Note

アカウントの唯一のサインイン URL は、アカウント ID に基づいています。エイリアス URL に接続しようとしても、リダイレクトされません。

## エイリアスの作成、削除、および一覧表示 (AWS CLI)

### Note

次のコマンドを実行するには、少なくとも以下の IAM アクセス許可が必要です。

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`
- `iam>DeleteAccountAlias`

AWS Management Console のサインインページの URL のエイリアスを作成するには、以下のコマンドを実行します。

- [`aws iam create-account-alias`](#)

AWS アカウント ID のエイリアスを削除するには、以下のコマンドを実行します。

- [`aws iam delete-account-alias`](#)

AWS アカウント ID のエイリアスを表示するには、以下のコマンドを実行します。

- [`aws iam list-account-aliases`](#)

### Example エイリアスコマンド

AWS アカウント ID のエイリアスを表示するには、以下のコマンドを実行します。

```
$ aws iam list-account-aliases
{
  "AccountAliases": [
    "myaccountalias"
  ]
}
```

AWS Management Console のサインインページの URL のエイリアスを作成するには、以下のコマンドを実行します。

```
$ aws iam create-account-alias \
```

```
--account-alias myaliasname
```

このコマンドは成功時に出力を生成しません。

AWS アカウント ID のエイリアスを削除するには、以下のコマンドを実行します。

```
$ aws iam delete-account-alias \  
--account-alias myaliasname
```

このコマンドは成功時に出力を生成しません。

## エイリアスの作成、削除、および一覧表示 (AWS API)

### Note

次の API 操作を実行するには、少なくとも以下の IAM アクセス許可が必要です。

- iam:ListAccountAliases
- iam:CreateAccountAlias
- iam>DeleteAccountAlias

AWS Management Console のサインインページの URL のエイリアスを作成するには、以下のオペレーションを呼び出します。

- [CreateAccountAlias](#)

AWS アカウント ID のエイリアスを削除するには、以下のオペレーションを呼び出します。

- [DeleteAccountAlias](#)

AWS アカウント ID のエイリアスを表示するには、以下のオペレーションを呼び出します。

- [ListAccountAliases](#)

# IAM の使用開始

このチュートリアルを使用して、AWS Identity and Access Management (IAM) を開始できます。AWS Management Console を使用して、ロール、ユーザー、ポリシーを作成する方法を学習します。

AWS Identity and Access Management は追加料金なしで提供される AWS アカウント の機能です。IAM ユーザーによる他の AWS 製品の使用に対してのみ請求されます。他の AWS 製品の料金設定については、[Amazon Web Services 料金設定ページ](#)を参照してください。

## Note

このドキュメントでは、主に IAM サービスを取り扱います。AWS の開始方法と、複数のサービスを使用して問題 (最初のプロジェクトの構築および起動) を解決する方法については、「[開始方法リソースセンター](#)」を参照してください。

## 目次

- [前提条件](#)
- [最初の IAM ユーザーを作成する](#)
- [最初のロールを作成します](#)
- [最初の IAM ポリシーの作成](#)
- [プログラマ的なアクセス](#)

## 前提条件

開始する前に、[IAM の準備作業](#) の手順を完了するようにしてください。このチュートリアルでは、その手順で作成した管理者アカウントを使用します。

## 最初の IAM ユーザーを作成する

[IAM ユーザー](#)は、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント 内のアイデンティティです。ユーザーは、同じ許可を共有するグループに編成できます。

**Note**

セキュリティ上の**ベストプラクティス**として、IAM ユーザーを作成するのではなく、ID フェデレーションを通じてリソースへのアクセスを提供することをお勧めします。IAM ユーザーが必要な特定の状況についての情報は、「[IAMユーザー \(ロールの代わりに\) を作成する場合](#)」を参照してください。

このチュートリアルでは、IAM ユーザーの作成プロセスに慣れるために、緊急アクセス用の IAM ユーザーとグループを作成するステップを説明します。

最初の IAM ユーザーを作成するには

1. AWS サインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. ナビゲーションペインで [ユーザー]、[ユーザーを追加] の順に選択します。

**Note**

IAM Identity Center を有効にしている場合は、AWS Management Console には、IAM Identity Center でユーザーのアクセスを管理するのが最善であることを示すメッセージが表示されます。このチュートリアルでは、作成する IAM ユーザーは、IAM Identity Center の認証情報のユーザーが使用できない場合にのみ使用できます。

4. User name (ユーザー名) に「**EmergencyAccess**」と入力します。名前にスペースを含めることはできません。
5. [AWS Management Console へのユーザーアクセスを提供する — オプション] の横にあるチェックボックスを選択し、[IAM ユーザーを作成する] を選択します。
6. [コンソールパスワード] で、[自動生成パスワード] を選択します。
7. [ユーザーは次回サインイン時に新しいパスワードを作成する必要があります (推奨)] の横にあるチェックボックスをオンにします。この IAM ユーザーは緊急アクセス用であるため、信頼できる管理者がパスワードを保持し、必要な場合にのみ提供します。
8. [権限の設定] ページの [権限オプション] の、[ユーザーをグループに追加] を選択します。次に、[ユーザーグループ] で [グループの作成] を選択します。

9. [ユーザーグループの作成] ページの [ユーザーグループ名] に、**EmergencyAccessGroup** と入力します。次に、[許可ポリシー] で [AdministratorAccess] を選択します。
10. [ユーザーグループの作成] を選択して [許可の設定] ページに戻ります。
11. [ユーザーグループ] で、以前に作成した **EmergencyAccessGroup** の名前を選択します。
12. [次へ] を選択して [確認と作成] ページに進みます。
13. [確認と作成] ページで、新しいユーザーに追加するユーザーグループメンバーシップのリストを確認します。続行する準備ができたなら、[ユーザーの作成] を選択します。
14. [パスワードの取得] ページで、[.csv ファイルのダウンロード] を選択し、ユーザーの認証情報 (接続 URL、ユーザー名、パスワード) を含む .csv ファイルを保存します。
15. このファイルを保存して、IAM にサインインする必要があるが、フェデレーション ID プロバイダーにアクセスできない場合に使用します。

新しい IAM ユーザーが [ユーザー] リストに表示されます。[ユーザー名] リンクを選択すると、ユーザーの詳細が表示されます。[概要] で、ユーザーの ARN をクリップボードにコピーします。ARN をテキストドキュメントに貼り付けて、次の手順で使用できるようにします。

## 最初のロールを作成します

IAM ロールは、信頼できるエンティティにアクセス権限を付与するための安全な方法です。IAM ロールは、IAM ユーザーといくつかの類似点を持っています。ロールとユーザーは、両方とも、ID が AWS でできることとできないことを決定するアクセス許可ポリシーを持つプリンシパルです。ただし、ユーザーは 1 人の特定の人に一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。また、ロールには標準の長期認証情報 (パスワードやアクセスキーなど) も関連付けられません。代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。ロールを使用すると、IAM のベストプラクティスに従うのに役立ちます。ロールは、以下の用途で使用できます。

- AWS IAM Identity Center を使用して AWS Management Console にアクセスすることを。従業員の ID と Identity Center 対応アプリケーションに対し許可します。
- ユーザー代わってアクションを実行するためのアクセス許可を AWS のサービスに委任します。
- AWS リソースへのアクセスや変更を、Amazon EC2 インスタンスで実行されているアプリケーションコードに対し許可します。
- 別の AWS アカウント にアクセス権を付与します。

**Note**

マシン ID へのアクセス権を付与するために、AWS Identity and Access Management Roles Anywhere を使用できます。IAM Roles Anywhere を使用すると、AWS の外部で実行されるワークロードの長期的な認証情報を管理する必要がなくなります。詳細については、「AWS Identity and Access Management Roles Anywhere ユーザーガイド」の「[AWS Identity and Access Management Roles Anywhere とは?](#)」を参照してください。

IAM Identity Center およびその他の AWS サービスは、サービスのロールを自動的に作成します。IAM ユーザーを使用している場合は、ユーザーがサインインするときに引き継ぐロールを作成することをお勧めします。これにより、長期間のアクセス許可ではなく、セッション中の一時的な許可が付与されます。

ロールの作成手順を案内する AWS Management Console ウィザードは、IAM ユーザー、AWS サービス、フェデレーションユーザーのいずれにロールを作成するかによって、手順が少し異なるように表示されます。組織内の AWS アカウント への定期的なアクセスは、フェデレーションアクセスを使用して提供する必要があります。緊急アクセスやプログラムによるアクセスなど、特定の目的で IAM ユーザーを作成する場合は、その IAM ユーザーにのみロールを引き受ける許可を付与し、その IAM ユーザーをロール固有のグループに配置します。

この手順では、EmergencyAccess IAM ユーザーに SupportUser アクセスを提供するロールを作成します。この手順を開始する前に、IAM ユーザーの ARN をクリップボードにコピーします。

IAM ユーザーのロールを作成するには

1. AWS サインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. IAM コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
4. [AWS アカウント] のロールタイプを選択します。
5. [信頼できるエンティティの選択] の [信頼できるエンティティの種類] で、[カスタム信頼ポリシー] を選択します。
6. [カスタム信頼ポリシー] セクションで、基本的な信頼ポリシーを確認してください。このロールではこれを使用します。[ステートメントの編集] エディターを使用して信頼ポリシーを更新します。
  1. [STS のアクションを追加] で、[役割を引き受ける] を選択します。



2. [プリンシパルの追加] の横にある [追加] を選択します。[プリンシパルの追加] ウィンドウが開きます。

[プリンシパルの種類] で、[IAM ユーザー] を選択します。

[ARN] の下に、コピーした IAM ユーザー ARN をクリップボードに貼り付けます。

[プリンシパルを追加] を選択します。

3. 信頼ポリシーの Principal 行に、指定した ARN が含まれていることを確認します。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:user/username" }
```

7. [ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。
8. [アクセス許可の追加] で、適用するアクセス許可ポリシーの横にあるチェックボックスを選択します。このチュートリアルでは、[SupportUser] 信頼ポリシーを選択します。その後、このロールを使用して、AWS アカウントとのトラブルシューティングと問題解決を行ったり、AWS とのサポートケースを開いたりできます。現時点では、[アクセス許可の境界](#)を設定するつもりはありません。
9. [Next] (次へ) をクリックします。
10. [名前、確認、作成] で、これらの設定を完了します。
  - [ロール名] に、SupportUserRole など、このロールであることを識別できる名前を入力します。
  - [説明] に、ロールの使用目的を説明してください。

他の AWS リソースがロールを参照している場合があるため、作成後はロールの名前を変更できません。

11. [ロールの作成] を選択します。

ロールが作成されたら、このロールを必要とするユーザーとロール情報を共有します。ロール情報は次の方法で共有できます。

- **ロールリンク:** 詳細がすべて既に入力されている [Switch Role] (ロールの切り替え) ページへのリンクをユーザーに送信します。
- **アカウント ID またはエイリアス:** ロール名とアカウント ID 番号またはアカウントのエイリアスを各ユーザーに提供します。これにより、ユーザーは [ロールの切り替え] ページに移動し、詳細を手動で追加できます。

- ロールリンク情報を EmergencyAccess ユーザーの認証情報と共に保存します。

詳細については、「[ユーザーへの情報の提供](#)」を参照してください。

## 最初の IAM ポリシーの作成

IAM ポリシーは、IAM ID (ユーザー、ユーザーのグループ、またはロール) または AWS リソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けられると、これらの許可を定義します。

最初の IAM ポリシーを作成するには

1. AWS サインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. ナビゲーションペインで [Policies] (ポリシー) を選択します。

[Policies] (ポリシー) を初めて選択する場合は、[Welcome to Managed Policies] (マネージドポリシーによるこそ) ページが表示されます。[Get Started] (今すぐ始める) を選択します。

4. [Create policy] (ポリシーを作成) を選択します。
5. [ポリシーを作成] ページで、[アクション] を選択し、[ポリシーをインポート] を選択します。
6. [ポリシーをインポート] ウィンドウの [ポリシーを検索] ボックスで、ポリシーのリストを減らすために **power** と入力します。[PowerUserAccess] ポリシーを選択します。
7. [ポリシーをインポート] を選択します。ポリシーは [JSON] タブに表示されます。
8. [Next] (次へ) をクリックします。
9. [確認および作成] ページで、[ポリシー名] に **PowerUserExamplePolicy** と入力します。説明に「**Allows full access to all services except those for user management**」と入力します。次に、[ポリシーの作成] を選択してポリシーを保存します。

このポリシーをロールにアタッチすると、そのロールを引き受けるユーザーにこのポリシーに関連する権限を与えることができます。[PowerUserAccess] ポリシーは、開発者にアクセスを提供するためによく使用されます。

## プログラムのなアクセス

AWS Management Console の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラムによるアクセスを許可する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

- IAM Identity Center で ID を管理する場合、AWS API にはプロファイルが必要で、AWS Command Line Interface にはプロファイルまたは環境変数が必要です。
- IAM ユーザーがいる場合、AWS API と AWS Command Line Interface にはアクセスキーが必要です。可能な限り、アクセスキー ID、シークレットアクセスキー、および認証情報の失効を示すセキュリティトークンが含まれる一時的な認証情報を作成します。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラマチックアクセス権を必要とするユーザー	To	方法
ワークフォースアイデンティティ  (IAM Identity Center で管理されているユーザー)	短期認証情報を使用して、AWS CLI または AWS API (直接または AWS SDK を使用して) へのプログラムによるリクエストに署名します。	使用するインターフェイスの指示に従ってください。 <ul style="list-style-type: none"> <li>• AWS CLI については、AWS IAM Identity Center ユーザーガイドの「<a href="#">Getting IAM role credentials for CLI access</a>」(CLI アクセス用の IAM ロール認証情報の取得)の指示に従ってください。</li> <li>• AWS API については、「AWS SDK およびツールリファレンスガイド」の「<a href="#">SSO 認証情報</a>」の指示に従ってください。</li> </ul>
IAM	短期認証情報を使用して、AWS CLI または AWS API (直接または AWS SDK を	「 <a href="#">AWS リソースでの一時的な認証情報の使用</a> 」の指示に従ってください。

プログラマチックアクセス権を必要とするユーザー	To	方法
	使用して) へのプログラムによるリクエストに署名します。	
IAM	長期間の認証情報を使用して、AWS CLI または AWS API (直接またはAWS SDK を使用して) へのプログラムによるリクエストに署名します。  (非推奨)	<a href="#">「IAM ユーザーのアクセスキーの管理」</a> の指示に従ってください。

# AWS Identity and Access Management のセキュリティのベストプラクティスとユースケース

AWS Identity and Access Management (IAM) には、独自のセキュリティポリシーを開発および実装する際に考慮する必要のあるいくつかのセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを提供するものではありません。これらのベストプラクティスは顧客の環境に必ずしも適切または十分でない可能性があるため、処方箋ではなく、あくまで有用な検討事項とお考えください。

IAM から最大の利点を得るために、時間をかけて推奨ベストプラクティスを習得してください。この方法の 1 つとして、実世界のシナリオにおいて他の AWS サービスと連携しながらどのように IAM を活用するかについて確認します。

## トピック

- [IAM でのセキュリティのベストプラクティス](#)
- [AWS アカウントのルートユーザーのベストプラクティス](#)
- [IAM のビジネスユースケース](#)

## IAM でのセキュリティのベストプラクティス

 [Follow us on Twitter](#)

AWS Identity and Access Management ベストプラクティスは 2022 年 7 月 14 日に更新されました。

AWS のリソースを保護するには、AWS Identity and Access Management (IAM) を使用する際の以下のベストプラクティスに従ってください。

## トピック

- [人間のユーザーが一時的な認証情報を使用して AWS にアクセスする場合に ID プロバイダーとのフェデレーションを使用することを必須とする](#)
- [ワークロードが AWS にアクセスする場合に IAM ロールで一時的な資格情報を使用することを必須とする](#)
- [多要素認証 \(MFA\) を必須とする](#)

- [長期的な認証情報を必要とするユースケースのためにアクセスキーを必要な時に更新する](#)
- [ルートユーザーの認証情報を保護するためのベストプラクティスに沿う](#)
- [最小特権アクセス許可を適用する](#)
- [AWS 管理ポリシーの使用を開始し、最小特権のアクセス許可に移行する](#)
- [IAM Access Analyzer を使用して、アクセスアクティビティに基づいて最小特権ポリシーを生成する](#)
- [未使用のユーザー、ロール、アクセス許可、ポリシー、および認証情報を定期的に確認して削除する](#)
- [IAM ポリシーで条件を指定して、アクセスをさらに制限する](#)
- [IAM Access Analyzer を使用して、リソースへのパブリックアクセスおよびクロスアカウントアクセスを確認する](#)
- [IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保する](#)
- [複数のアカウントにまたがるアクセス許可のガードレールを確立する](#)
- [アクセス許可の境界を使用して、アカウント内のアクセス許可の管理を委任する](#)

## 人間のユーザーが一時的な認証情報を使用して AWS にアクセスする場合に ID プロバイダーとのフェデレーションを使用することを必須とする

人間のユーザーとは、別名人間 ID と呼ばれ、人、管理者、デベロッパー、オペレーター、およびアプリケーションのコンシューマーを指します。人間のユーザーは AWS の環境とアプリケーションにアクセスするための ID を持っている必要があります。組織のメンバーである人間のユーザーは、ワークフォースアイデンティティとも呼ばれます。人間のユーザーには、あなたと共同作業を行う外部ユーザーや、あなたの AWS のリソースを操作する外部ユーザーも含まれます。この操作は、ウェブブラウザ、クライアントアプリケーション、モバイルアプリ、またはインタラクティブなコマンドラインツールを介して実行できます。

人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用が必要です。一時的な資格情報を提供するロールを引き受けることで、人間のユーザーの ID プロバイダーを使用した AWS アカウントへのフェデレーションアクセスが可能になります。一元的なアクセス管理を行うには、[AWS IAM Identity Center \(IAM Identity Center\)](#) を使用して、ご自分のアカウントへのアクセスと、それらのアカウント内でのアクセス許可を管理することをお勧めします。ユーザー ID を、IAM Identity Center を使用して管理することも、外部 ID プロバイダーから、IAM Identity Center 内のユーザー

ID に付与するアクセス許可を管理することもできます。詳細については、[AWS IAM Identity Center ユーザーガイド](#)の AWS IAM Identity Center とはを参照してください。

ロールの詳細については、「[ロールに関する用語と概念](#)」をご参照ください。

## ワークロードが AWS にアクセスする場合に IAM ロールで一時的な資格情報を使用することを必須とする

ワークロードとは、アプリケーションやバックエンドプロセスなど、ビジネス価値を提供するリソースやコードの集合体のことです。ワークロードには、AWS のサービス へのリクエスト (データの読み取りリクエストなど) を行うために ID を必要とするアプリケーション、運用ツール、コンポーネントが含まれている場合があります。これらの ID には、Amazon EC2 インスタンスや AWS Lambda 関数などの AWS 環境で実行中のマシンが含まれます。

また、アクセスを必要とする外部の関係者のマシン ID を管理することもできます。マシン ID へのアクセスを許可するには、IAM ロールを使用できます。IAM ロールは特定のアクセス許可を持ち、ロールセッションで一時的なセキュリティ認証情報に依存することで AWS にアクセスする方法を提供します。さらに、AWS 環境にアクセスする必要がある AWS の外部のマシンが含まれる場合もあります。AWS の外部で実行されるマシンには、[AWS Identity and Access Management Roles Anywhere](#)を使用できます。ロールの詳細については、「[IAM ロール](#)」をご参照ください。ロールを使用して AWS アカウント へのアクセス許可を委任する方法については「[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)」を参照してください。

## 多要素認証 (MFA) を必須とする

AWS リソースにアクセスする人間のユーザーとワークロードの IAM ロールを使用して、一時的な認証情報を使用することをお勧めします。ただし、アカウントに IAM ユーザー またはルートユーザーが必要なシナリオでは、セキュリティを強化するために MFA が必要になります。MFA では、ユーザーは認証チャレンジに対するレスポンスを生成するデバイスを所有します。サインインプロセスを完了するには、各ユーザーの認証情報とデバイス生成のレスポンスの両方が必要です。詳細については、[AWS での多要素認証 \(MFA\) の使用](#)を参照してください。

IAM Identity Center を使用して人間のユーザーによるアクセスを一元的に管理する場合、ID ソースが IAM Identity Center の ID ストア、AWS 管理の Managed Microsoft AD、または AD Connector で設定されていれば、IAM Identity Center の MFA 機能を使用することができます。IAM Identity Center での MFA の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[Multi-factor authentication](#)」(多要素認証)を参照してください。



## 長期的な認証情報を必要とするユースケースのためにアクセスキーを必要な時に更新する

可能であれば、アクセスキーなどの長期的な認証情報を作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、プログラムによるアクセスと長期的な認証情報を持つ IAM ユーザーが必要なシナリオでは、従業員が退職したときなど、必要に応じてアクセスキーを更新することをお勧めします。IAM アクセス最終使用者情報を利用して、アクセスキーを安全に更新して削除することをお勧めします。詳細については、「[アクセスキーの更新](#)」を参照してください。

AWS の IAM ユーザーとの長期的な認証情報が必要な特定のユースケースがあります。ユースケースには次のようなものがあります。

- IAM ロールを使用できないワークロード – AWS へのアクセスが必要な場所から、ワークロードを実行する場合があります。状況によっては、IAM ロールを使用して WordPress プラグインなどに対して一時的な認証情報を提供することができない場合もあります。このような状況では、そのワークロードに対して IAM ユーザーの長期的なアクセスキーを使用して、AWS への認証を行います。
- サードパーティー AWS クライアント – IAM Identity Center を使用したアクセスがサポートされていないツール (AWS でホストされていないサードパーティー AWS クライアントまたはベンダーなど) を使用している場合は、IAM ユーザーの長期的なアクセスキーを使用します。
- AWS CodeCommit アクセス – CodeCommit を使用してコードを保存している場合、CodeCommit の SSH キーまたはサービス固有の認証情報を持つ IAM ユーザーを使用して、リポジトリへの認証を行うことができます。通常の認証に IAM Identity Center のユーザーを使用することに加えて、これを行うことをお勧めします。IAM Identity Center のユーザーとは、お客様の AWS アカウント またはクラウドアプリケーションにアクセスする必要がある従業員のことです。IAM ユーザーを設定せずに CodeCommit リポジトリへのアクセス許可をユーザーに付与するには、`git-remote-codecommit` ユーティリティを設定します。IAM および CodeCommit の詳細については、「[CodeCommit での IAM の使用: Git 認証情報、SSH キー、および AWS アクセスキー](#)」を参照してください。`git-remote-codecommit` ユーティリティの設定についての詳細は、「[AWS CodeCommit ユーザーガイド](#)」の「[認証情報のローテーションを使用した AWS CodeCommit リポジトリへの接続](#)」を参照してください。
- Amazon Keyspaces (Apache Cassandra 向け) へのアクセス – IAM Identity Center 内のユーザーを使用できない状況 (Cassandra との互換性をテストする場合など) では、サービス専用の認証情報を持つ IAM ユーザーを使用して Amazon Keyspaces で認証できます。IAM Identity Center のユーザーとは、お客様の AWS アカウント またはクラウドアプリケーションにアクセスする必要がある従業員のことです。一時的な認証情報を使用して Amazon Keyspaces に接続することも

できます。詳細については、「Amazon Keyspaces (Apache Cassandra 向け) デベロッパガイド」の「[Using temporary credentials to connect to Amazon Keyspaces using an IAM role and the SigV4 plugin](#)」(一時的な認証情報を使用して IAM ロールと SigV4 プラグインを使用して Amazon Keyspaces に接続する)を参照してください。

## ルートユーザーの認証情報を保護するためのベストプラクティスに沿う

AWS アカウント を作成すると、AWS Management Console にサインするためのルートユーザーの認証情報が設定されます。他の機密性の高い個人情報を保護するのと同じ方法で、ルートユーザーの認証情報を保護します。ルートユーザープロセスを保護してスケールする方法の詳細については、「[AWS アカウントのルートユーザーのベストプラクティス](#)」を参照してください。

## 最小特権アクセス許可を適用する

IAM ポリシーでアクセス許可を設定するときは、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。その際、ワークロードやユースケースに必要なアクセス許可を検討しながら、大まかなアクセス許可から始めるとよいでしょう。ユースケースが成熟してきたら、最小特権になるように付与する権限を減らしていくことができます。IAM を使用してアクセス許可を適用する方法については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

## AWS 管理ポリシーの使用を開始し、最小特権のアクセス許可に移行する

ユーザーやワークロードへのアクセス許可の付与を開始するには、多くの一般的なユースケースに対してアクセス許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。AWS 管理ポリシーは、すべての AWS のユーザーが使用できるため、特定のユースケースに対して最小特権のアクセス許可が付与されない場合があることに留意してください。そのため、ユースケースに応じた[カスタマー管理ポリシー](#)を定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「[AWS マネージドポリシー](#)」を参照してください。特定のジョブ機能用に設計された AWS 管理ポリシーの詳細については、[AWSジョブ機能の 管理ポリシー](#)を参照してください。

## IAM Access Analyzer を使用して、アクセスアクティビティに基づいて最小特権ポリシーを生成する

タスクの実行に必要なアクセス許可のみを付与するには、AWS CloudTrail にログインしているアクセスアクティビティに基づいてポリシーを生成することができます。[IAM Access Analyzer](#) は、IAM

ロールが使用するサービスとアクションを分析し、使用可能な詳細なポリシーを生成します。生成された各ポリシーをテストしたら、ポリシーを本番環境にデプロイできます。これにより、ワークロードに必要なアクセス許可のみが付与されます。ポリシー生成の詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

## 未使用のユーザー、ロール、アクセス許可、ポリシー、および認証情報を定期的に確認して削除する

AWS アカウントには、必要なくなった IAM ユーザー、ロール、アクセス許可、ポリシー、または認証情報がある可能性があります。IAM から提供される最後にアクセスした情報をもとに、この情報から不要になったユーザー、ロール、アクセス許可、ポリシー、および認証情報を特定し、削除できます。これにより、監視する必要のあるユーザー、ロール、アクセス許可、ポリシー、および認証情報の数を減らすことができます。この情報をもとに、最小特権のアクセス許可により適切に準拠できるように IAM ポリシーを調整することもできます。詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## IAM ポリシーで条件を指定して、アクセスをさらに制限する

ポリシーステートメントが有効になる条件を指定することができます。これにより、アクションやリソースへのアクセスを許可することができますが、これは、アクセスのリクエストが特定の条件を満たしている場合に限られます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定することができます。また、条件を使用してサービスアクションへのアクセスを許可することもできますが、これは、AWS CloudFormation などの特定の AWS のサービスを介して使用する場合に限られます。詳細については、「[IAM JSON ポリシー要素Condition](#)」を参照してください。

## IAM Access Analyzer を使用して、リソースへのパブリックアクセスおよびクロスアカウントアクセスを確認する

AWS でパブリックアクセスまたはクロスアカウントアクセスのアクセス許可を付与する前に、そのアクセス許可が必要かどうかを確認することをお勧めします。IAM Access Analyzer を使用すると、サポートされているリソースタイプのパブリックアクセスとクロスアカウントアクセスをプレビューおよび分析できます。これを行うには、IAM Access Analyzer が生成する[調査結果](#)を確認します。これらの調査結果から、リソースアクセス制御が期待した通りにアクセスを許可しているかどうかを確認できます。さらに、パブリックおよびクロスアカウントのアクセス許可を更新すると、リソースに新しいアクセス制御をデプロイする前に、変更の効果を検証できます。また、IAM Access Analyzer は、サポートされているリソースタイプを継続的に監視し、パブリックアクセスまたはクロスアカウ

ントアクセスを許可するリソースの調査結果を生成します。詳細については、「[Previewing access with IAM Access Analyzer APIs](#)」(IAM Access Analyzer API を使用したアクセスのプレビュー) を参照してください。

## IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保する

作成したポリシーを検証して、ポリシーが [IAM ポリシー言語](#) (JSON) と IAM のベストプラクティスに準拠していることを確認します。IAM Access Analyzer ポリシーチェックを使用して、ポリシーを検証できます。IAM Access Analyzer は 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーを作成できるようサポートします。コンソールで新しいポリシーの作成や、既存のポリシーの編集を行う際に、IAM Access Analyzer は、ポリシーが保存される前にポリシーを調整して検証するのに役立つ推奨事項を提供します。既存のポリシーをすべて見直し、検証することをお勧めします。詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。IAM Access Analyzer が提供するポリシーチェックの詳細については、「[IAM Access Analyzer ポリシーチェックリファレンス](#)」を参照してください。

## 複数のアカウントにまたがるアクセス許可のガードレールを確立する

ワークロードをスケールするときには、AWS Organizations で管理されている複数のアカウントを使用してワークロードを分離します。Organizations の [サービスコントロールポリシー](#) (SCP) を使用して、アカウント全体のすべてのIAMユーザーとロールのアクセスを制御するためのアクセス許可ガードレールを確立することをお勧めします。SCP は組織ポリシーの一種で、AWS 組織、OU、またはアカウントレベルで組織内のアクセス許可を管理するために使用することができます。確立したアクセス許可のガードレールは、対象となるアカウント内のすべてのユーザーとロールに適用されます。しかし、組織のアカウントにアクセス許可を付与するには、SCP だけでは不十分です。管理者は、[アイデンティティベースのポリシーまたはリソースベースのポリシー](#)を IAM ユーザー、IAM ロール、またはアカウント内のリソースにアタッチする必要があります。詳細については、「[AWS Organizations、アカウント、および IAM ガードレール](#)」を参照してください。

## アクセス許可の境界を使用して、アカウント内のアクセス許可の管理を委任する

シナリオによっては、アカウント内のアクセス許可の管理を他の人に委任する場合があります。例えば、デベロッパーが自分のワークロードのロールを作成および管理できるようにすることができます。アクセス許可を他のユーザーに委任するときは、アクセス権限の境界を使用して、委任する権限の上限を設定します。アクセス許可の境界は、管理ポリシーを使用してアイデンティティベースのポリシーが IAM ロールに付与できるアクセス許可の上限を設定する高度な機能です。アクセス許可の

境界自体は、アクセス許可を付与しません。詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。

## AWS アカウントのルートユーザーのベストプラクティス

AWS アカウントを初めて作成するとき、アカウントのすべての AWS リソースに完全にアクセスできるデフォルトの認証情報を使用します。この ID は、[AWS アカウントのルートユーザー](#)と呼ばれます。[ルートユーザーの認証情報が必要なタスク](#)がある場合を除き、AWS アカウントのルートユーザーにはアクセスしないことを強くお勧めします。ルートユーザーの認証情報とアカウント復旧メカニズムを保護して、高い特権のある認証情報が不正使用で悪用されないようにする必要があります。

ルートユーザーにアクセスする代わりに、日常的なタスク用の管理者ユーザーを作成します。

- 単一のスタンドアロン AWS アカウントの場合、「[管理アクセス権を持つユーザーを作成する](#)」を参照してください。
- AWS Organizations を介して管理されている複数の AWS アカウントの場合、「[IAM Identity Center 管理者ユーザーの AWS アカウントアクセスを設定する](#)」を参照してください。

管理者ユーザーを使用して、AWS アカウントのリソースにアクセスする必要のあるユーザーの追加 ID を作成できます。AWS にアクセスするユーザーには一時的な認証情報での認証を要求することを強くお勧めします。

- 単一のスタンドアロン AWS アカウントの場合、[IAM ロール](#) を使用して特定のアクセス許可を持つ ID をアカウントに作成します。ロールは、それを必要とするすべてのユーザーに割り当てることが可能です。また、ロールには、パスワードやアクセスキーなどの標準的な長期認証情報が関連付けられていません。その代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。IAM ロールとは異なり、[IAM ユーザー](#)にはパスワードやアクセスキーなどの長期的な認証情報があります。可能であれば、[ベストプラクティス](#)では、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。
- Organizations を介して管理される複数の AWS アカウントの場合、IAM Identity Center ワークフォースユーザーを使用します。IAM Identity Center を使用すると、AWS アカウントにアクセスするユーザーとアカウント内のアクセス許可を一元管理できます。ユーザー ID は IAM Identity Center または外部の ID プロバイダーから管理します。詳細については、[AWS IAM Identity Center ユーザーガイド](#)の AWS IAM Identity Center とは を参照してください。

### トピック



- [ルートユーザーの認証情報を保護して不正使用を防止する](#)
- [アクセスを保護するために、強度の高いルートユーザーパスワードを設定します](#)
- [多要素認証 \(MFA\) でルートユーザーのサインインを保護する](#)
- [ルートユーザーのアクセスキーは作成しないでください](#)
- [可能な場合、ルートユーザーのサインインには複数人による承認を求る](#)
- [ルートユーザーの認証情報にグループ E メールアドレスを使用する](#)
- [アカウント回復メカニズムへのアクセスを制限する](#)
- [Organizations アカウントのルートユーザー認証情報を保護する](#)
- [アクセスおよび使用状況をモニタリングする](#)

## ルートユーザーの認証情報を保護して不正使用を防止する

ルートユーザーの認証情報を保護し、[それを必要とするタスク](#)にのみ使用します。不正使用を防ぐため、ルートユーザーにアクセスするためのビジネス上の厳しい要件がある場合を除き、ルートユーザーのパスワード、MFA、アクセスキー、CloudFront キーペア、署名証明書は誰とも共有しないでください。

ルートユーザーのパスワードは、同じパスワードを使用してアクセスするアカウント内の AWS のサービスに依存するツールで使用しないでください。ルートユーザーのパスワードを紛失または忘れた場合、これらのツールにアクセスできなくなります。回復性を優先し、保管場所へのアクセスに 2 人以上のユーザーによる承認を求めることを検討することをお勧めします。パスワードとその保存場所へのアクセスはすべてログに記録し、モニタリングする必要があります。

## アクセスを保護するために、強度の高いルートユーザーパスワードを設定します

強力でユニークなパスワードを使用することをおすすめします。強力なパスワード生成アルゴリズムを備えたパスワードマネージャーなどのツールは、これらの目標を達成するのに役立ちます。AWS では、パスワードは次の条件を満たす必要があります。

- 8~128 文字で構成されていること。
- 英字の大文字と小文字、数字、および ! @ # \$ % ^ & \* ( ) < > [ ] { } | \_ + = の記号を含める必要があります。
- AWS アカウント アカウント名またはメールアドレスと同じでないこと。

詳細については、「[AWS アカウントのルートユーザー のパスワードを変更する](#)」を参照してください。

## 多要素認証 (MFA) でルートユーザーのサインインを保護する

ルートユーザーは特権的アクションを実行できるので、E メールアドレスとパスワードに加えて、サインインの認証情報としてルートユーザーの MFA を 2 番目の認証要素として追加することが重要です。セキュリティ戦略の柔軟性と回復性を高めるためにルートユーザーの認証情報に複数の MFA を有効にすることを強くお勧めします。AWS アカウント ルートユーザーに対し、現在サポートされている MFA タイプの任意の組み合わせで最大 8 台の MFA デバイスを登録できます。

- サードパーティプロバイダーから提供される FIDO 認定のハードウェアセキュリティキー。詳細については、「[AWS アカウントルートユーザーの FIDO セキュリティキーを有効にする](#)」を参照してください。
- タイムベースドワンタイムパスワード (TOTP) アルゴリズムに基づいて 6 桁の数値コードを生成するハードウェアデバイス。詳細については、「[AWS アカウントのルートユーザー用にハードウェア TOTP トークンを有効にします](#)」を参照してください。
- 電話などのデバイスで実行され、物理デバイスをエミュレートする仮想認証機能アプリケーション。詳細については、「[AWS アカウントアカウントのルートユーザーの仮想 MFA デバイスを有効にします](#)」を参照してください。

## ルートユーザーのアクセスキーは作成しないでください

アクセスキーを使用すると、AWS コマンドラインインターフェイス (AWS CLI) でコマンドを実行することや、いずれかの AWS SDK から API オペレーションを使用することができます。ルートユーザーには請求情報を始めとするアカウントのすべての AWS のサービスとリソースへのフルアクセスがあるので、ルートユーザー用のアクセスキーペアを作成しないことを強くお勧めします。

ルートユーザーが必要とするタスクはごくわずかです。通常、そのようなタスクは頻繁に実行されないため、AWS Management Consoleにサインインしてルートユーザーのタスクを実行することをお勧めします。アクセスキーを作成する前に、[長期的なアクセスキーの代替案](#)を確認してください。

## 可能な場合、ルートユーザーのサインインには複数人による承認を求る

ルートユーザーの MFA とパスワードの両方に 1 人のユーザーがアクセスできないようにするには、複数人による承認を求めることを検討してください。一部の企業では、パスワードにアクセスできる管理者グループと MFA にアクセスできる管理者グループを設定することでセキュリティを強化し



ています。ルートユーザーの認証情報を使用してサインインするには、各グループから 1 人のメンバーが必要です。

## ルートユーザーの認証情報にグループ E メールアドレスを使用する

ビジネスで管理され、受信したメッセージがユーザーのグループに直接転送されるメールアドレスを使用します。AWS がアカウントの所有者に連絡する必要がある場合、このアプローチを取ることで、担当者の誰かが休暇中である場合、病欠である場合、または既に退職している場合でも応答の遅延のリスクを削減できます。ルートユーザーに使用されている E メールアドレスを他の目的で使用しないでください。

## アカウント回復メカニズムへのアクセスを制限する

管理者アカウントの乗っ取りなどの緊急時にアクセスが必要になった場合に備えて、ルートユーザーの認証情報回復メカニズムを管理するプロセスを準備しておいてください。

- [ルートユーザーのパスワードを紛失した場合や忘れた場合にリセット](#)できるように、ルートユーザーの E メールを受信トレイにアクセスできることを確認します。
- AWS アカウントのルートユーザーの MFA デバイスが紛失した場合、破損した場合、または機能しない場合は、同じルートユーザー認証情報に登録されている別の MFA デバイスを使用してサインインできます。すべての MFA にアクセスできなくなった場合、MFA を回復するには、アカウント登録時に使用した電話番号とメールアドレスの両方が最新でアクセス可能である必要があります。詳細については、「[ルートユーザー MFA デバイスの回復](#)」を参照してください。
- ルートユーザーのパスワードと MFA を保管しない場合、ルートユーザーの認証情報を復元する代替方法としてアカウントに登録されている電話番号を使用できます。連絡先の電話番号にアクセスできることを確認し、電話番号を最新の状態に保ち、電話番号を管理できるユーザーを制限します。

E メールを受信トレイと電話番号は、どちらもルートユーザーのパスワードを回復するための検証手段であるため、誰も両方にアクセスできないようにする必要があります。これらのチャネルは、2 つのグループに分け管理することが重要です。1 つのグループはメインのメールアドレスにアクセスし、もう 1 つのグループはメインの電話番号にアクセスしてルートユーザーとしてアカウントへのアクセスを回復します。

## Organizations アカウントのルートユーザー認証情報を保護する

Organizations のマルチアカウント戦略に移行すると、各 AWS アカウントにはセキュリティで保護する必要のある独自のルートユーザー認証情報が設定されます。組織を使用するために使用するアカウントは管理アカウントです。組織内の残りのアカウントはメンバーアカウントです。

### メンバーアカウント用のルートユーザー認証情報を保護する

Organizations を使用して複数のアカウントを管理する場合、Organizations 内のルートユーザーアクセスを保護するために講じることができる 2 つの戦略があります。

- Organizations アカウントのルートユーザー認証情報を MFA で保護する
- アカウントのルートユーザーパスワードはリセットせず、パスワードリセットプロセスを使用して必要に応じてアクセスのみを回復します。組織でメンバーアカウントを作成すると、Organizations でメンバーアカウントに IAM ロールが自動的に作成され、管理アカウントがメンバーアカウントに一時的にアクセスできるようになります。

詳細については、「Organizations ユーザーガイド」の「[組織のメンバーアカウントへのアクセス](#)」を参照してください。

### サービスコントロールポリシー (SCP) を使用して、Organizations に予防的セキュリティコントロールを設定する

Organizations を使用して複数のアカウントを管理する場合、SCP を適用してメンバーアカウントのルートユーザーへのアクセスを制限できます。特定のルート専用アクションを除き、メンバーアカウントのすべてのルートユーザーアクションを拒否することで不正アクセスを防ぐことができます。詳細については、「[SCP を使用し、メンバーアカウントのルートユーザーで行えることを制限する](#)」を参照してください。

### アクセスおよび使用状況をモニタリングする

現在の追跡メカニズムを使用して、ルートユーザーのログインと使用状況をモニタリング、アラート、レポートすることをお勧めします。これには、ルートユーザーのサインインと使用を知らせるアラートも含まれます。以下のサービスは、ルートユーザーの認証情報の使用状況を追跡し、不正使用を防ぐセキュリティチェックを実行するのに役立ちます。

- アカウント内のルートユーザーのサインインアクティビティに関する通知を受け取る場合は、Amazon CloudWatch を利用して、ルートユーザーの認証情報が使用されたことを検出し、

セキュリティ管理者への通知をトリガーするイベントルールを作成できます。詳細については、「[AWS アカウントのルートユーザーアクティビティのモニタリングと通知](#)」を参照してください。

- 承認されたルートユーザーアクションを通知するアラートを設定する場合は、Amazon EventBridge と Amazon SNS を組み合わせて、特定のアクションのルートユーザーの使用状況を追跡し、Amazon SNS トピックを使用して通知する EventBridge ルールを作成できます。詳細については、「[Amazon S3 オブジェクトが作成されたときに通知を送信する](#)」を参照してください。
- 脅威検出サービスとして既に GuardDuty を使用している場合は、その[機能を拡張](#)して、アカウントでルートユーザーの認証情報が使用されていることを通知することができます。

アラートには、ルートユーザーの E メールアドレスなどの情報を含めるよう設定します。ルートユーザーアクセスに関するアラートを受信する担当者がそのルートユーザーアクセスが予期されたものであることを確認する方法を理解し、セキュリティインシデントが発生していると判断した場合にエスカレーションする方法を理解できるようにアラートへの対応方法を準備しておく必要があります。設定例については、「[AWS アカウントのルートユーザーアクティビティのモニタリングと通知](#)」を参照してください。

## ルートユーザーの MFA コンプライアンスを評価する

- AWS Config は、ルートユーザーのベストプラクティスの適用に役立つルールを使用します。AWS マネージドルールを使用して[ルートユーザーに多要素認証 \(MFA\) の有効化を要求する](#)ことができます。AWS Config は、[ルートユーザーのアクセスキーを識別](#)することもできます。
- Security Hub は、AWS のセキュリティ状態を包括的に把握するビューを提供し、AWS 環境をセキュリティ業界標準およびベストプラクティスに照らして評価するために役立ちます。使用可能なルールの詳細については、「Security Hub ユーザーガイド」の「[AWS Identity and Access Management コントロール](#)」を参照してください。
- Trusted Advisor は、ルートユーザーアカウントで MFA が有効になっていることを確認するためのセキュリティチェックを提供します。詳細については、「AWS Support ユーザーガイド」の「[ルートアカウントの MFA](#)」を参照してください。

アカウントのセキュリティ問題を報告する必要がある場合は、「[不信な E メールを報告](#)」または「[脆弱性レポート](#)」を参照してください。または、[AWS に問い合わせ](#)て、サポートや追加のガイダンスを求めることもできます。

# IAM のビジネスユースケース

IAM の簡単なビジネスユースケースを紹介します。お客様のユーザーが持っている AWS アカウントをコントロールするサービスを実装できる基本的な方法を解説します。ユースケースでは、お客様が求めている結果を得るために IAM API を使用する方法を、一般的に説明しています。

このユースケースでは、Example Corp という架空の企業が IAM を使用する典型的な 2 つの方法を検討します。最初のシナリオは、Amazon Elastic Compute Cloud (Amazon EC2) です。2 つ目は、Amazon Simple Storage Service (Amazon S3) です。

IAM と AWS の他のサービスとの連携の詳細については、「[IAM と連携する AWS のサービス](#)」を参照してください。

## トピック

- [Example Corp の初期設定](#)
- [Amazon EC2 での IAM ユースケース](#)
- [Amazon S3 での IAM のユースケース](#)

## Example Corp の初期設定

Nikki Wolf と Mateo Jackson は Example Corp の創設者です。彼らは、同社の設立時に AWS アカウントを作成し、さらに AWS リソースで使用する管理者アカウントを作成するために、AWS IAM Identity Center (IAM Identity Center) をセットアップします。管理ユーザーのアカウントへのアクセス権をセットアップすると、対応する IAM ロールが IAM Identity Center により作成されます。この、IAM Identity Center が制御するロールは、関連する AWS アカウントに作成され、[AdministratorAccess] アクセス許可セットで指定されたポリシーがアタッチされています。

現在、Nikki と Mateo は管理用アカウントが使用できるようになったので、ルートユーザーにより AWS アカウントにアクセスする必要がなくなりました。ルートユーザーは、ルートユーザーのみが実行可能なタスク用だけに使用する予定です。セキュリティ上のベストプラクティスを見直した後、彼らはルートユーザーの認証情報のために多要素認証 (MFA) を設定し、ルートユーザーの保護方法を確立します。

会社が成長するにつれ、彼らは、開発者、管理者、テスト担当、マネージャー、そしてシステム管理者として働く従業員を招き入れます。運用を担当するのは Nikki で、Mateo はエンジニアリングチームを指揮します。従業員のアカウントの管理と、社内リソースへのアクセス管理には、Active Directory ドメインサーバーをセットアップします。

従業員に AWS リソースへのアクセス権を付与するためには、IAM Identity Center を使用して、会社の Active Directory を AWS アカウント に接続しています。

Active Directory が IAM Identity Center が接続されているため、ユーザー、グループ、およびグループのメンバーシップを定義し同期することができます。AWS リソースへのアクセス権限を適切なレベルでユーザに付与するには、許可セットとロールをさまざまなグループに割り当てる必要があります。これらの許可セットを作成するためには、AWS Management Console から [AWSジョブ機能の管理ポリシー](#) を使用します。

- 管理者
- 請求
- 開発者
- ネットワーク管理者
- データベース管理者
- システム管理者
- サポートユーザー

次に、これらの許可セットを、自社の Active Directory グループに割り当てられたロールに割り当てます。

IAM Identity Center の初期設定についてのステップバイステップガイドは、「AWS IAM Identity Center ユーザーガイド」の「[Getting started](#)」(使用開始)で確認してください。IAM Identity Center ユーザーアクセスのプロビジョニングの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[AWS アカウントへのシングルサインオンアクセス](#)」を参照してください。

## Amazon EC2 での IAM ユースケース

一般的に、Example Corp のような会社は IAM を使用して Amazon EC2 などのサービスとやり取りします。ユースケースの当該部分を理解するには、Amazon EC2 の基本的な理解が必要となります。Amazon EC2 の詳細については、「[Amazon EC2 ユーザーガイド](#)」を参照してください。

### ユーザーグループの Amazon EC2 アクセス許可

「境界」の制御を構築するため、John は AllUsers ユーザーグループにポリシーをアタッチします。このポリシーは、元の IP アドレスが Example Corp の企業ネットワークの外部にある場合、ユーザーからの AWS 要求を拒否します。

Example Corp では、ユーザーグループごとに異なるアクセス許可が必要です。

- システム管理者 – AMI、インスタンス、スナップショット、ボリューム、セキュリティグループなどの作成および管理のアクセス許可が必要です。Nikki は、ユーザーグループのメンバーがすべての Amazon EC2 アクションを使用できるように許可する AWS マネージドポリシー AmazonEC2FullAccess を、SysAdmins ユーザーグループにアタッチします。
- デベロッパー – インスタンスを使用する機能のみが必要です。従って Mateo は、DescribeInstances、RunInstances、StopInstances、StartInstances、TerminateInstances の呼び出し許可を開発者に与えるためにポリシーを作成し、それを Developers ユーザーグループにアタッチします。

#### Note

Amazon EC2 は SSH キー、Windows パスワードおよびセキュリティグループを使用して、特定の Amazon EC2 インスタンスのオペレーティングシステムへのアクセス許可を持つユーザーを制御します。IAM システムには、特定のインスタンスのオペレーティングシステムへのアクセスを許可または拒否するメソッドは存在しません。

- サポートユーザー – 現在使用可能な Amazon EC2 リソースの一覧表示を除き、いずれの Amazon EC2 アクションも実行することはできません。そのため、Nikki は Amazon EC2 の「Describe」API オペレーションのみを呼び出せるようにするポリシーを作成してサポートユーザーグループにアタッチします。

これらのポリシーの記述例については、「Amazon EC2 ユーザーガイド」の「[IAM アイデンティティベースのポリシーの例](#)」および「[AWS Identity and Access Management の使用](#)」を参照してください。

## ユーザーのジョブ関数の変更

ある時点において、開発者の 1 人である Paulo Santos の職務機能が変更され、マネージャーになりました。マネージャーである Paulo は、サポートユーザーグループの一員となり、開発者のためのサポートケースを開けるようになります。Mateo は、Paulo を Developers ユーザーグループからサポートユーザーグループに移動します。この移動の結果、Amazon EC2 インスタンスを操作する際の彼の権限は制限されます。Don はインスタンスの起動や開始をすることができません。また Don がインスタンスを起動または開始させた人物だったとしても、既存のインスタンスの停止や終了をすることができません。Don は、Example Corp のユーザーが起動するインスタンスを表示することのみできます。



## Amazon S3 での IAM のユースケース

Example Corp のような会社では、IAM と Amazon S3 も使用するのが一般的です。John は aws-s3-bucket という会社用の Amazon S3 バケットを作成しました。

### その他のユーザーおよびユーザーグループの作成

Zhang Wei と Mary Major は、従業員として会社のバケットに自分用のデータを作成する必要があります。また、すべての開発者が作業する共有データを読み取りや書き込みをする必要があります。これを可能にするため、Mateo は次の図のように Amazon S3 キープレフィックススキームを使用して、「aws-s3-bucket」にあるデータを論理的に配置します。

```
/aws-s3-bucket
  /home
    /zhang
    /major
  /share
    /developers
    /managers
```

Mateo は、/aws-s3-bucket を、各従業員用の一連のホームディレクトリと、開発者グループおよびマネージャーグループ用の共有エリアに分割します。

そして、Mateo はユーザーおよびユーザーグループにアクセス許可を割り当てるための一連のポリシーを作成します。

- Zhang のホームディレクトリアクセス – Mateo は、Amazon S3 キープレフィックス /aws-s3-bucket/home/zhang/ の付いたすべてのオブジェクトの読み取り、書き込み、一覧表示を許可するポリシーを Wei にアタッチします。
- Major のホームディレクトリアクセス – Mateo は、Amazon S3 キープレフィックス /aws-s3-bucket/home/major/ の付いたすべてのオブジェクトの読み取り、書き込み、一覧表示を許可するポリシーを Mary にアタッチします。
- 開発者ユーザーグループによる共有ディレクトリへのアクセス – Mateo は、/aws-s3-bucket/share/developers/ 内のすべてのオブジェクトの読み取り、書き込み、一覧表示を開発者に許可するポリシーを、このユーザーグループにアタッチします。
- マネージャーユーザーグループの共有ディレクトリアクセス – Mateo は、/aws-s3-bucket/share/managers/ 内のオブジェクトの読み取り、書き込み、一覧表示をマネージャーに許可するポリシーを、このユーザーグループにアタッチします。



**Note**

Amazon S3はバケットまたはオブジェクトを作成するユーザーに、そのバケットまたはオブジェクトに対してその他のアクションを実行するアクセス許可を自動的に付与することはありません。したがって、IAM ポリシーでは、作成する Amazon S3 リソースを使用する明示的なアクセス許可をユーザーに付与する必要があります。

これらのポリシーの例については、[Amazon Simple Storage Service ユーザーガイド](#)の「アクセスコントロール」を参照してください。ランタイムでのポリシーの評価方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。

## ユーザーのジョブ関数の変更

ある時点において、開発者の1人である Zhang Wei の職務機能が変更され、マネージャーに変わります。彼は、share/developers ディレクトリ内のドキュメントにアクセスする必要がなくなるとみなされます。管理者である Mateo は、Wei を Developers ユーザーグループから外し Managers ユーザーグループへ移動させます。この簡単な再割り当てだけで、Wei には自動的に Managers ユーザーグループへのすべてのアクセス許可が与えられますが、share/developers ディレクトリ内のデータにはアクセスできなくなります。

## サードパーティービジネスとの統合

組織は、頻繁に提携会社、コンサルタント、請負業者と組んで仕事をすることがあります。Example Corp は Widget Company と提携しています。Widget Company の従業員である Shirley Rodriguez は、Example Corp の使用しているバケットにデータを入力する必要があります。Nikki は、WidgetCo というユーザーグループ、そして Shirley というユーザーを作成し、この WidgetCo ユーザーグループに Shirley を追加します。また、Nikki は Shirley が使用するための特別なバケット、aws-s3-bucket1 も作成します。

Nikki は、パートナーの Widget Company に提供するために、既存のポリシーの更新または新規ポリシーの追加を行います。例えば、Nikki は WidgetCo ユーザーグループのメンバーに対し、書き込み以外のいかなるアクションも拒否するという新規ポリシーを作成することができます。このポリシーは、広範囲にわたる一連の Amazon S3 アクションへのアクセス許可をすべてのユーザーに与えるような広義のポリシーが存在する場合にのみ必要となります。

# IAM のチュートリアル

次のチュートリアルでは、AWS Identity and Access Management (IAM) の一般的なタスクにおけるエンドツーエンドの一連の手順について説明します。これらはラボ型環境向けであり、架空の企業名やユーザー名などを含みます。目的は、一般的なガイダンスを提供することです。お客様の組織環境に固有のニーズを満たすかどうかの十分な確認や調整をすることなく、本番環境で直接使用するためのものではありません。

## チュートリアル

- [IAM チュートリアル: 請求コンソールへのアクセス権の付与](#)
- [IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)
- [IAM チュートリアル: はじめてのカスタマー管理ポリシーの作成とアタッチ](#)
- [IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#)
- [IAM チュートリアル: ユーザーに自分の認証情報および MFA 設定を許可する](#)

## IAM チュートリアル: 請求コンソールへのアクセス権の付与

AWS アカウントの所有者 ([AWS アカウントのルートユーザー](#)) は、IAM ユーザーとロールに AWS アカウントの AWS Billing and Cost Management データへのアクセス権を付与できます。このチュートリアルの手順は、事前にテストされたシナリオの設定に役立ちます。このシナリオでは、メインの AWS 本番稼働用アカウントに影響を与えることなく、請求アクセス許可の設定を実務的に体験できます。

### 前提条件

このチュートリアルの手順を実行する前に、次の準備を行います。

- テスト用の AWS アカウントを作成します。
- ルートユーザーとしてテスト用の AWS アカウントにサインインします。
- チュートリアルで使用できるように、テストアカウントの AWS アカウント番号を記録します。このチュートリアルでは、アカウント番号の例として 111122223333 を使用します。ステップでそのアカウント番号を使用する場合は必ず、テストアカウント番号に置き換えてください。

## ステップ 1: テスト用の AWS アカウントで請求データへの IAM アクセスを有効にする

このシナリオでは、ルートユーザーとしてテスト用の AWS アカウントにサインインし、IAM に請求情報へのアクセスを付与します。IAM に請求情報へのアクセスを付与すると、IAM ユーザーとロールが AWS Billing and Cost Management コンソールにアクセスできるようになります。この設定では、IAM ユーザーとロールにこれらのコンソールページに必要なアクセス許可が付与されるのではなく、必要な IAM ポリシーがある IAM ユーザーとロールがアクセスできるようにします。ポリシーが既に IAM ユーザーまたはロールにアタッチされていても、この設定が有効になっていない場合、それらのポリシーによって付与されたアクセス許可は有効になりません。

### Note

AWS Organizations を使用して作成した AWS アカウント では、請求情報への IAM アクセスがデフォルトで有効になっています。

## ステップ 2: テスト用のユーザーとグループを作成する

このシナリオでは、IAM ユーザーに請求コンソールへのアクセス権を付与し、2 人のユーザーを作成します。

- Pat Candella

Pat は財務部門のメンバーで、請求と支払いを担当しています。Pat には、AWS アカウントでの請求情報へのフルアクセスが必要です。

- Terry Whitlock

Terry は IT サポート部門の一員です。ほとんどの場合、Terry は請求コンソールにアクセスする必要はありませんが、財務部門の社員の質問に答えるためにアクセスが必要な場合があります。

## ステップ 3: AWS Billing コンソールへのアクセス権を付与するロールを作成する

IAM ロールは、特定の許可があり、アカウントで作成できる IAM アイデンティティです。IAM ロールは IAM ユーザーと似ていて、どちらも AWS で実行できることとできないことを判別するアクセス許可ポリシーが付加された AWS ID です。ただし、ユーザーは 1 人の特定の人に一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。また、ロールには標準の長期認証情報 (パスワードやアクセスキーなど) も関連付けられません。代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。ロールを使用して、通常は AWS リソースへのアクセス権のないユーザー、アプリケー

ション、サービスにそのアクセス権を委任できます。このシナリオでは、Terry Whitlock が請求コンソールにアクセスできるようにするロールを作成します。

## ステップ 4: コンソールへのアクセスをテストする

中心となるタスクを完了したら、ポリシーをテストすることができます。テストにより、ポリシーが期待したとおりに動作することを確認できます。各ユーザーのアクセスをテストすることで、ユーザーが体験することを比較できます。

## 前提条件

このチュートリアルの手順を実行する前に、次の準備を行います。

- テスト用の AWS アカウントを作成します。
- ルートユーザーとしてテスト用の AWS アカウント にサインインします。
- チュートリアルで使用できるように、テストアカウントの AWS アカウント 番号を記録します。このチュートリアルでは、アカウント番号の例として 111122223333 を使用します。ステップでそのアカウント番号を使用する場合は必ず、テストアカウント番号に置き換えてください。

## ステップ 1: テスト用の AWS アカウントで請求データへの IAM アクセスを有効にする

このシナリオでは、ルートユーザーとしてテスト用の AWS アカウントにサインインし、IAM に請求情報へのアクセスを付与します。請求情報へのアクセスを付与すると、IAM ユーザーとロールが AWS Billing and Cost Management コンソールにアクセスできるようになります。この設定では、IAM ユーザーとロールにこれらのコンソールページに必要なアクセス許可が付与されるのではなく、必要な IAM ポリシーがある IAM ユーザーとロールがアクセスできるようにするだけです。

### Note

AWS Organizations を使用して作成した AWS アカウント では、請求情報への IAM アクセスがデフォルトで有効になっています。

請求情報とコスト管理コンソールへの IAM ユーザーおよびロールのアクセスをアクティベートするには

1. ルートユーザー認証情報 (AWS アカウントの作成に使用した E メールアドレスとパスワード) で AWS Management Console にサインインします。
2. ナビゲーションバーでアカウント名を選択してから、[\[アカウント\]](#) を選択します。
3. ページを下にスクロールして、[\[請求情報への IAM ユーザーとロールのアクセス\]](#) セクションが見つかったら、[\[編集\]](#) を選択します。
4. [\[Activate IAM Access\]](#) (アクセスのアクティブ化) チェックボックスをオンにして、Billing and Cost Management ページへのアクセスをアクティブ化します。
5. [\[更新\]](#) を選択します。

このページには、[\[IAM ユーザーとロールの請求情報へのアクセスがアクティブ化されています\]](#) というメッセージが表示されます。

このチュートリアルの次のステップでは、特定の請求機能へのアクセスを許可または拒否する IAM ポリシーをアタッチします。

## ステップ 2: テスト用のユーザーとグループを作成する

テスト用の AWS アカウントには、ルートユーザー以外の ID は定義されていません。請求情報にアクセスできるようにするために、追加の ID を作成して、そこに請求情報へのアクセスを許可できるようにします。

テスト用のユーザーとグループを作成する


1. [\[Root user\]](#) (ルートユーザー) を選択して AWS アカウント の E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

### Note

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[\[IAM ユーザーのサインイン\]](#) ページが表示された場合、ページ下部付近で [\[ルートユーザーの電子メールを使用してサインインする\]](#) を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイド

の「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

- ナビゲーションペインで [ユーザー]、[ユーザーを追加] の順に選択します。

 Note

IAM Identity Center を有効にしている場合は、AWS Management Console には、IAM Identity Center でユーザーのアクセスを管理するのが最善であることを示すメッセージが表示されます。このチュートリアルでは、請求情報にアクセスできるようにする方法について習得するために IAM ユーザーを作成します。IAM Identity Center でユーザーを作成した場合は、IAM ではなく IAM Identity Center を使用してこれらのユーザーまたはグループに [請求] アクセス許可セットを割り当てます。

- [User name] (ユーザー名) に「**pcandella**」と入力します。名前にスペースを含めることはできません。
- [ユーザーが AWS Management Console にアクセスできるようにする – オプション] の横にある選択ボックスを選択し、[IAM ユーザーを作成する] を選択します。
- [コンソールパスワード] で、[自動生成パスワード] を選択します。
- [ユーザーは次回のサインイン時に新しいパスワードを作成する必要があります (推奨)] の横にある選択ボックスのチェックを外してから、[次へ] を選択します。この IAM ユーザーはテスト用なので、認証手続きで使用するパスワードをダウンロードします。
- [権限の設定] ページの [権限オプション] の、[ユーザーをグループに追加] を選択します。次に、[ユーザーグループ] で [グループの作成] を選択します。
- [ユーザーグループの作成] ページの [ユーザーグループ名] に、**BillingGroup** と入力します。次に、[アクセス許可ポリシー] で、AWS 管理職務機能ポリシーの [請求] を選択します。
- [ユーザーグループの作成] を選択して [許可の設定] ページに戻ります。
- [ユーザーグループ] で、作成した **BillingGroup** の選択ボックスを選択します。
- [次へ] を選択して [確認と作成] ページに進みます。
- [確認して作成] ページで、新しいユーザーのユーザーグループメンバーシップのリストを確認します。続行する準備ができたなら、[ユーザーの作成] を選択します。
- [パスワードを取得] ページで、[.csv ファイルをダウンロード] を選択し、ユーザーのサインイン情報 (接続 URL、ユーザー名、パスワード) を含む .csv ファイルを保存します。

この IAM ユーザーとして AWS にサインインするときの参照用に、このファイルを保存します。

14. [ユーザーリストに戻る] を選択します。
15. この手順を以下のように変更して繰り返し、Terry Whitlock 用のユーザーとサポートユーザー用のグループを作成します。
  - a. ステップ 3 で、[ユーザー名] に、「**twhitlock**」と入力します。
  - b. ステップ 8 で、[ユーザーグループ名] に、「**SupportGroup**」と入力します。次に、[アクセス許可ポリシー] で、AWS 管理職務機能ポリシーの [SupportUser] を選択します。

新しい IAM ユーザー、グループ、ロールはコンソールリストで確認できます。作成した各アイテムについて、名前を選択して詳細を表示できます。ユーザーの詳細を表示すると、コンソールには、**pcandella** の [アクセス許可ポリシー] として [請求] が、**twhitlock** の [アクセス許可ポリシー] として [SupportUser] が一覧表示されます。

ポリシーを使用して IAM ユーザーに AWS Billing and Cost Management 機能へのアクセス権を付与する方法の詳細については、「AWS Billing ユーザーガイド」の「[AWS Billing でアイデンティティベースのポリシー \(IAM ポリシー\) を使用する](#)」を参照してください。

## ステップ 3: AWS Billing コンソールへのアクセス権を付与するロールを作成する

ロールを使用して IAM ユーザーに請求コンソールへのアクセス権を付与できます。ロールは、ユーザーが必要なときに引き受けることができる一時的な認証情報を提供します。このチュートリアルでは、財務部門からのサポート依頼で問題の調査が必要な場合に、ユーザー **twhitlock** が請求情報にアクセスできる必要があります。

1. [Root user] (ルートユーザー) を選択して AWS アカウントの E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

### Note

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[IAM ユーザーのサインイン] ページが表示された場合、ページ下部付近で [ルートユーザーの電子メールを使用してサインインする] を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイド



の「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

- ナビゲーションペインで [ユーザー] を選択し、ユーザー **twhitlock** を選択してユーザーの詳細を表示します。ユーザー **twhitlock** の ARN をクリップボードにコピーします。
- ナビゲーションペインで [ロール]、[ロールを作成] の順に選択します。
- [信頼されたエンティティを選択] ページで、[カスタム信頼ポリシー] を選択し、[ステートメントを編集] で次の項目を入力します。
  - [STS のアクションを追加] - AssumeRole が選択されていることを確認します。
  - [プリンシパルを追加] - [追加] を選択すると [プリンシパルを追加] ダイアログボックスが表示されます。[プリンシパルタイプ] には、[IAM ユーザー] を選択し、[ARN] にはステップ 16 でクリップボードにコピーしたユーザー **twhitlock** の ARN を貼り付けます。そして、[プリンシパルを追加] を選択します。
- [次へ] を選択して [許可を追加] ページに移動します。
- フィルターボックスの [許可ポリシー] に、「**Billing**」と入力して AWS 管理職務機能ポリシー [請求] を選択します。
- [次へ] を選択して、[名前、確認、作成] ページに移動します。[ロール名] に「**TempBillingAccess**」と入力し、[ロールを作成] を選択します。

ロールが作成されたことが通知されます。ロールを表示すると、ロールの詳細が表示されます。[概要]セクションでは、次の情報をメモしておきます。


- デフォルトでは、セッションの最大継続時間は 1 時間です。その期間が過ぎると、ロールを引き受けたユーザーはアカウントの基本的なアクセス許可に戻ります。ユーザーがロールのアクセス許可を引き続き使用したい場合は、ロールを再度切り替える必要があります。ロールを編集して最大期間を延長できます。セッションの期間は最長 12 時間まで可能です。
- コンソールでロールを切り替えるためのリンク。リンクをコピーして、信頼ポリシーにプリンシパルとして追加したユーザーに直接提供できます。[信頼関係] タブから信頼ポリシーを表示および編集できます。

## ステップ 4: コンソールへのアクセスをテストする

テストユーザーとしてサインインしてアクセスをテストし、ユーザーが体験することを確認することをお勧めします。次のステップを使用して、アクセス権限の違いを確認するために、両方のテストアカウントを使用してサインインします。

両方のテストユーザーでサインインして請求へのアクセスをテストするには

1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

 Note

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある場合は、ページの下部にある[別のアカウントにサインイン]を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようにすることができます。

2. 以下に示すステップを使用して各ユーザーにサインインし、さまざまなユーザー体験を比較できるようにします。

#### フルアクセス

- a. ユーザー **pcandella** として AWS アカウント にサインインします。
- b. ナビゲーションバーで `pcandella@111122223333` を選択し、次に [請求ダッシュボード] を選択します。
- c. 各ページを参照し、さまざまなボタンを選択して、完全な変更アクセス許可があることを確認します。

#### アクセス権なし

- a. ユーザー **twhitlock** として AWS アカウント にサインインします。
- b. ナビゲーションバーで `[twhitlock@111122223333]` を選択し、次に [請求ダッシュボード] を選択します。
- c. 「You need permissions」というメッセージが表示されます。請求データは表示されません。

#### ロールを切り替えてアクセス権を拡大

- a. ユーザー **twhitlock** として AWS アカウント にサインインします。

- b. ナビゲーションバーで [twhitlock@111122223333] を選択し、次に [ロールの切り替え] を選択します。

[ロールの切り替え] ページが開きます。情報を以下のように入力します。

- [アカウント]-111122223333
- [ロール]-TempBillingAccess

[ロールの切り替え] を選択します。

または、[コンソールでロールを切り替えるためのリンク] に記載されている URL を使用して、[ロールの切り替え] ページを開くこともできます。

- c. コンソールには [AWS Billing ダッシュボード] が表示され、ナビゲーションバーには TempBillingAccess@111122223333 と表示されます。

## まとめ

これで、IAM ユーザーに AWS Billing コンソールへのアクセス権を付与するために必要な手順が完了しました。その結果、ユーザーが請求コンソールで体験することがどのようなものかを直接確認できました。これで、このロジックを本稼働環境にいつでも実装できます。

## 関連リソース

AWS Billing ユーザーガイドの関連情報については、以下の関連リソースを参照してください。

- [AWS Billing コンソールへのアクセスのアクティブ化](#)
- [AWS 請求ポリシーの例](#)
- [AWS 請求でアイデンティティベースのポリシー \(IAM ポリシー\) を使用する](#)
- [AWS Billing 向けのアクセスコントロールの移行](#)

IAM ユーザーガイドの関連情報については、以下の関連リソースを参照してください。

- [管理ポリシーとインラインポリシー](#)
- [AWS Management Console への IAM ユーザーアクセスのコントロール](#)
- [IAM ユーザーグループへのポリシーのアタッチ](#)

# IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任

このチュートリアルでは、ロールを使用して、お客様が所有する作成および開発という異なる AWS アカウントのリソースへのアクセスを委任する方法について説明します。特定のアカウントにあるリソースを別のアカウントのユーザーと共有します。このようにクロスアカウントアクセスを設定することで、お客様はアカウントごとに IAM ユーザーを作成する必要がなくなります。また、ユーザーは、異なる AWS アカウントのアカウントのリソースにアクセスするために、あるアカウントからサインアウトして別のアカウントにサインインする必要がなくなります。ロールを設定した後、AWS Management Console、AWS CLI、API からロールを使用する方法について説明します。

## Note

IAM ロールとリソースベースのポリシーは、単一のパーティション内のアカウント間でのみアクセスを委任します。たとえば、標準 aws パーティションの米国西部 (北カリフォルニア) にアカウントがあるとし、aws-cn パーティションの中国 (北京) にもアカウントがあります。標準 aws アカウントで、中国 (北京) のアカウントの Amazon S3 リソースベースのポリシーを使用して、ユーザーにアクセスを許可することはできません。

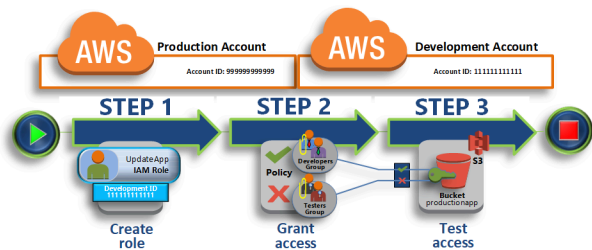
このチュートリアルでは、本番稼働用アカウントがライブアプリケーションを管理しています。デベロッパーやテスターは、開発アカウントをサンドボックスとして使用して、自由にアプリケーションをテストします。各アカウントでは、Amazon S3 バケットにアプリケーション情報を保存します。開発アカウントの IAM ユーザーを管理しており、開発とテスターの 2 つの IAM グループがあります。どちらのユーザーグループにも、開発用アカウントで作業し、そのリソースにアクセスするアクセス許可があります。ときどき、デベロッパーが本番稼働用アカウントのライブアプリケーションを更新する必要があります。デベロッパーは、これらのアプリケーションを productionapp という Amazon S3 バケットに保存します。

このチュートリアルを終了すると、次のようになります。

- 本番稼働用アカウントで特定のロールを引き受けることができる開発アカウント (信頼できるアカウント) のユーザー。
- 特定の Amazon S3 バケットへのアクセスを許可される本番稼働用アカウント (信頼するアカウント) のロール。
- 本番稼働用アカウントの productionapp バケット。

デベロッパーは、AWS Management Console でロールを使用して本番稼働用アカウントの productionapp バケットにアクセスできます。さらに、ロールにより提供される一時的な認証情報によって認証された API コールを使用してバケットにアクセスすることもできます。テスターがそのロールを使用して同様の操作を行うことはできません。

このワークフローに 3 つの基本的なステップがあります：



## 本番稼働用アカウントでロールを作成する

まず、AWS Management Console を使用して本番稼働用アカウント (ID 番号 999999999999) と開発用アカウント (ID 番号 111111111111) との間の信頼を確立します。続いて、UpdateApp という IAM ロールを作成します。ロールの作成時に、開発用アカウントを信頼されたエンティティとして定義し、信頼されたユーザーに productionapp バケットを更新する許可を与えるアクセス許可ポリシーを特定します。

## ロールにアクセス許可を付与する

チュートリアルのこのステップでは、IAM ユーザーグループを変更して、テスターの UpdateApp ロールへのアクセスを拒否するようにします。このシナリオではテスターに PowerUser アクセス許可があるため、ロールの使用を明示的に拒否する必要があります。

## ロールを切り換えてアクセスをテストする

最後にデベロッパーとして、UpdateApp ロールを使用して本番稼働用アカウントの productionapp バケットを更新します。AWS コンソール、AWS CLI、API を使用してロールにアクセスする方法について説明します。

## 前提条件

このチュートリアルでは、以下を実行済みであることを前提としています。

- 使用できる AWS アカウント を、それぞれ開発用アカウントと本番稼働用アカウントを表す 2 つのアカウントに分けます。
- 開発用アカウントのユーザーとユーザーグループは、次のように作成され、設定されます。

ユーザー	ユーザーグループ	許可
David	開発者	どちらのユーザーも開発用アカウントでサインインでき、AWS Management Console を使用できます。
Jane	テスター	

- 本番稼働用アカウントにはユーザーまたはユーザーグループを作成する必要はありません。
- 本番稼働用アカウントで作成された Amazon S3 バケット。このチュートリアルでは ProductionApp と呼びますが、S3 バケット名はグローバルに一意である必要があるため、別の名前のバケットを使用する必要があります。

## 本番稼働用アカウントでロールを作成する

ある AWS アカウント のユーザーに別の AWS アカウント のリソースへのアクセスを許可できます。これを行うには、ロールを作成し、そのロールにアクセスできるユーザーと、そのロールに切り替えるユーザーに付与するアクセス許可を定義します。

チュートリアルのこのステップでは、本番稼働用アカウントにロールを作成し、開発アカウントを信頼されたエンティティとして指定します。また、ロールのアクセス許可を productionapp バケットでの読み書き専用で制限します。ロールを使用するアクセス許可が付与されるすべてのユーザーは、productionapp バケットに対する読み取りと書き込みを実行できます。

ロールを作成する前に、AWS アカウント 開発のアカウント ID が必要です。各 AWS アカウント には、固有の識別子であるアカウント ID が割り当てられています。

開発 AWS アカウント ID を取得するには

1. 開発用アカウントの管理者として AWS Management Console にサインインし、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. ナビゲーションバーで、[サポート]、[サポートセンター] の順に選択します。現在サインインしている 12 桁のアカウント番号 (ID) は、サポートセンターナビゲーションペインを使用します。このシナリオでは、アカウント ID 111111111111 を開発用アカウントとして使用します。ただし、テスト環境でこのシナリオを使用する場合は、有効なアカウント ID を使用する必要があります。

## 開発アカウントで使用できるロールを本番稼働用アカウントで作成するには

1. 本番稼働用アカウントの管理者として AWS Management Console にサインインし、IAM コンソールを開きます。
2. ロールを作成する前に、ロールに必要なアクセス許可を定義する管理ポリシーを準備します。後の手順で、このポリシーをロールにアタッチします。

productionapp バケットの読み込みおよび書き込みのアクセスを設定します。AWS には Amazon S3 管理ポリシーがありますが、単一の Amazon S3 バケットへの読み取りおよび書き込みアクセスを提供するポリシーはありません。代わりにポリシーを自作することができます。

ナビゲーションペインで、[Policies (ポリシー)] を選択し、[Create Policy (ポリシーの作成)] を選択します。

3. [JSON] タブを選択し、以下の JSON ポリシードキュメントからテキストをコピーします。[JSON] (JSON) テキストボックスにこのテキストを貼り付け、リソース ARN (arn:aws:s3:::productionapp) を、Amazon S3 バケットの実際の ARN に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```



```
    }  
  ]  
}
```

ListAllMyBuckets アクションは、リクエストの認証済み送信者によって所有されているすべてのバケットを一覧表示するアクセス許可を付与します。ListBucket アクセス許可は、ユーザーに productionapp バケットにあるオブジェクトの閲覧を許可します。GetObject、PutObject、DeleteObject アクセス許可によって、ユーザーは productionapp バケットの内容を表示、更新、および削除できます。

4. [ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

5. [確認および作成] ページで、ポリシー名として「**read-write-app-bucket**」と入力します。ポリシーによって割り当てられたアクセス許可を確認し、[ポリシーの作成] を選択して作業を保存します。

新しいポリシーが管理ポリシーの一覧に表示されます。

6. ナビゲーションペインで [ロール] を選択した後、[ロールの作成] を選択します。
7. [AWS アカウント] のロールタイプを選択します。
8. [Account ID] (アカウント ID) で、開発アカウント ID を入力します。

このチュートリアルでは、開発アカウントにサンプルのアカウント ID **111111111111** を使用します。有効なアカウント ID を使用してください。使用しているアカウント ID が無効 (**111111111111** など) の場合は、IAM で新しいロールを作成することはできません。

現時点では、ロールを引き受けるために、外部 ID や多要素認証 (MFA) は必要ありません。これらのオプションは選択していない状態にしておきます。詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

9. [Next: Permissions] (次へ: アクセス許可) を選択して、そのロールに関連するアクセス許可を設定します。

10. 以前に作成したポリシーの横にあるチェックボックスを選択します。

#### ヒント

[Filter] (フィルター) で、[Customer managed] (カスタマー管理ポリシー) を選択してリストをフィルター処理し、自分で作成したポリシーのみが含まれるようにします。これにより、AWS が作成したポリシーが非表示になり、必要なポリシーを見つけるのが簡単になります。

次に、[次へ] を選択します。

11. (オプション) タグをキーバリューのペアとしてアタッチして、メタデータをロールに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
12. (オプション) [Description] (説明) には、新しいロールの説明を入力します。
13. ロールを確認したら、[ロールの作成] を選択します。

ロールのリストで、UpdateApp ロールが表示されます。

ここで、ロールに固有の識別子である Amazon リソースネーム (ARN) を取得しなければいけません。デベロッパーやテスターのユーザーグループのポリシーを変更するとき、アクセス許可を許可または拒否するためのロールの ARN を指定します。

UpdateApp の ARN を取得するには

1. IAM コンソールのナビゲーションペインで [Roles] (ロール) をクリックします。
2. ロールのリストで、[UpdateApp] ロールを選択します。
3. 詳細ペインの [Summary (概要)] セクションで、[ロール ARN] の値をコピーします。

Production アカウントのアカウント ID が 999999999999 であるため、ロール ARN は `arn:aws:iam::999999999999:role/UpdateApp` となります。本番稼働用アカウントでは、実際の AWS アカウント ID を指定してください。

この時点で、本番稼働用アカウントと開発用アカウント間の信頼が確立されました。そのためには、開発用アカウントを信頼されたプリンシパルとして識別するロールを本番稼働用アカウントに作

成します。また、UpdateApp ロールに切り替えるユーザーが何を実行できるかについても定義しました。

次は、ユーザーグループのアクセス許可を修正します。

## ロールにアクセス許可を付与する

この時点では、Testers と Developers のユーザーグループのメンバーにはいずれも、開発アカウントで自由にアプリケーションをテストできるアクセス許可があります。ロールの切り替えを許可するアクセス許可を追加するには、この必要な手順に従ってください。

UpdateApp ロールに切り替えることを許可するように Developers ユーザーグループを変更するには

1. 開発用アカウントの管理者としてサインインし、IAM コンソールを開きます。
2. [ユーザーグループ]、[Developers (開発者)] の順に選択します。
3. [アクセス許可] タブを選択し、[アクセス許可の追加] を選択してから、[インラインポリシーの作成] を選択します。
4. [JSON] タブを選択します。
5. 次のポリシーステートメントを追加して、Production アカウントの UpdateApp ロールに対する AssumeRole アクションを許可します。必ず Resource 要素の **PRODUCTION-ACCOUNT-ID** を、Production アカウントの実際の AWS アカウント ID に変更してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
  }
}
```

Allow エフェクトは、Developers グループが Production アカウントの UpdateApp ロールにアクセスすることを明示的に許可します。どのデベロッパーがこのロールにアクセスしようとしても成功します。

6. [ポリシーの確認] を選択します。
7. **allow-assume-S3-role-in-production** などの名前を入力します。
8. [Create policy] (ポリシーを作成) を選択します。

ほとんどの環境では、次の手順は必要ありません。ただし、PowerUserAccess アクセス許可を使用している場合、グループによってはロールを切り替えることができます。次の手順は、ロールを引き受けることができないように Testers グループに "Deny" アクセス許可を追加する方法を示しています。お客様の環境でこの手順が必要ない場合は、追加しないことをお勧めします。"Deny" アクセス許可を追加すると、アクセス許可の全体的な状況が複雑になり、管理しづらくなります。"Deny" アクセス許可は、他に良いオプションがない場合のみ使用してください。

**UpdateApp** ロールを引き継ぐためのアクセス許可を拒否するようにテスターユーザーグループを変更するには

1. [ユーザーグループ]、[Testers] の順に選択します。
2. [アクセス許可] タブを選択し、[アクセス許可の追加] を選択してから、[インラインポリシーの作成] を選択します。
3. [JSON] タブを選択します。
4. 次のポリシーステートメントを追加して、AssumeRole ロールに対する UpdateApp アクションを拒否します。必ず Resource 要素の **PRODUCTION-ACCOUNT-ID** を、Production アカунトの実際の AWS アカунト ID に変更してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
  }
}
```

Deny エフェクトは、Testers グループが Production アカунトの UpdateApp ロールにアクセスすることを明示的に拒否します。そのロールにアクセスしようとするテスターは、アクセス拒否のメッセージを受け取ります。

5. [ポリシーの確認] を選択します。
6. **deny-assume-S3-role-in-production** のような名前を入力します。
7. [Create policy] (ポリシーを作成) を選択します。

これで、Developers ユーザーグループは、Production アカунトの UpdateApp ロールを使用するためのアクセス許可を持つようになりました。Testers ユーザーグループは、UpdateApp ロールを使用できません。

次に、デベロッパーの David が本番稼働用アカウントの productionapp バケットにアクセスする方法について説明します。David は、AWS Management Console、AWS CLI、または AWS API からバケットにアクセスできます。

## ロールを切り換えてアクセスをテストする

このチュートリアル最初の 2 つのステップを完了した時点で、本番稼働用アカウントのリソースに対するアクセス許可を付与するロールがあります。また、開発用アカウントには、そのロールを使用できるユーザーで構成されたユーザーグループもあります。このステップでは、AWS Management Console、AWS CLI、および AWS API のロールへの切り替えについて説明します。

### Important

ロールの切り替えは、IAM ユーザーまたはフェデレーテッドユーザーとしてサインインした場合にのみ可能です。さらに、Amazon EC2 インスタンスを起動してアプリケーションを実行すると、そのアプリケーションはそのインスタンスプロファイルを通じてロールを引き受けることができます。AWS アカウントのルートユーザーとしてサインインしているときに、ロールを切り替えることはできません。

## ロールの切り替え (コンソール)

David が AWS Management Console で本番稼働用環境を操作する必要がある場合は、[Switch Role] (ロールの切り替え) を使用することができます。アカウント ID またはエイリアスとロール名を指定すれば、David のアクセス権限は、ロールによって許可されているものに直ちに切り替わります。次に、David はコンソールを使用して productionapp バケットを操作することができますが、本稼働環境の他のリソースは一切操作できません。David がロールを使用中に、開発用アカウントのパワーユーザーアクセス許可を使用することはできません。これは、同時に有効にできるアクセス許可のセットは 1 つのみであるためです。

### Important

AWS Management Console を使ったロールの切り替えは、ExternalId を必要としないアカウントでのみ機能します。たとえば、アカウントへのアクセス権を第三者に付与し、アクセス許可ポリシーの Condition 要素に ExternalId を要求するとします。その場合、第三者は AWS API またはコマンドラインツールを使用してのみ、お客様のアカウントにアクセスできます。第三者は ExternalId の値を指定できないため、コンソールを使用できません。このシナリオの詳細については、『[AWS リソースへのアクセス権を第三者に付与す](#)

[るときに外部 ID を使用する方法](#) セキュリティログ』の「AWS Management Console」および「[AWS へのクロスアカウントアクセスを有効にする方法](#)」を参照してください。

David が IAM で [Switch Role] (ロールの切り替え) ページに移動するには、2 つの方法があります。

- David は事前定義されたロールの切り替え設定を指すリンクを管理者から受け取ります。リンクは [ロールの作成] ウィザードの最終ページ、またはクロスアカウントロールの [ロールの概要] ページで管理者に提供されます。このリンクを選択すると、[ロールの切り替え] ページに移動します。このページには、[Account ID (アカウント ID)] および [ロール名] フィールドがすでに入力されています。David は、[Switch Roles] (ロールの切り替え) ボタンを選択するだけです。
- 管理者はメールでリンクを送信しませんが、代わりに [Account ID (アカウント ID)] 番号および [ロール名] の値を送信します。役割を切り替えるには、デビッドは手動でその値を入力する必要があります。これを次の手順に示します。

ロールを割り当てるには

1. David は、開発用ユーザーグループの通常ユーザーを使用して AWS Management Console にサインインします。
2. 管理者がユーザーに E メールするリンクを選択します。これにより、David は、[Switch Role] (ロールの切り替え) ページに移動されます。このページには、アカウント ID、エイリアス、およびロール名情報がすでに入力されています。

-または-

David は、ナビゲーションバーのユーザー名 (ID メニュー) を選択してから、[Switch Roles.] (ロールの切り替え) を選択します。

David がこの方法で初めて [Switch Role] (スイッチロール) ページにアクセスする場合は、初回アクセス用の [Switch Role] (スイッチロール) ページが表示されます。このページでは、ロールを切り替えて AWS アカウント 全体にわたるリソースを管理できるようにする方法についての追加情報が説明されます。David がこの手順の残りを完了するには、このページの [Switch Role] (ロールの切り替え) ボタンを選択する必要があります。

3. 次に、ロールにアクセスするために、David は Production アカウント ID 番号 (999999999999) およびロール名 (UpdateApp) を入力する必要があります。

またDavid は、現在 IAM でアクティブなロールと、関連するアクセス許可をモニタリングしたいと考えています。この情報を追跡するために、[Display Name (表示名)] テキストボックスに



「PRODUCTION」と入力し、赤色のオプションを選択して、[Switch Role (スイッチロール)] を選択します。

- これで、David は Amazon S3 コンソールを使用して、UpdateApp ロールがアクセス許可を持つバケットなどのリソースを操作できます。
- 完了すると、David は元のアクセス権限に戻ることができます。そのためには、ナビゲーションバーのロール表示名 [PRODUCTION] (プロダクション) を選択してから、[Back to David @ 111111111111] (David @ 111111111111 に戻る) を選択します。
- David が次回にロールに切り替えるために、ナビゲーションバーの [Identity] (ID) メニューを選択すると、PRODUCTION エントリが前回からそのままになっています。そのエントリを選択するだけで、アカウント ID とロール名を再入力することなく、すぐにロールに切り替えることができます。

## ロールの切り替え (AWS CLI)

David がコマンドラインで本稼働環境を操作する必要がある場合、[AWS CLI](#) を使用してこの切り替えを行うことができます。aws sts assume-role コマンドを実行し、ロールの ARN を渡して、そのロールの一時的なセキュリティ認証情報を取得します。次に、環境変数でそれらの認証情報を設定し、それ以降の AWS CLI コマンドが、ロールのアクセス許可を使用して動作するようにします。David がロールを使用中に、開発用アカウントのパワーユーザーの権限を使用することはできません。これは、一度に有効にできるアクセス許可のセットは 1 つのみであるためです。

すべてのアクセスキーとトークンは例にすぎず、実際にはそのように使用できないことに注意してください。ライブ環境の適切な値に置き換えてください。

ロールを割り当てるには

- David はコマンドプロンプトウィンドウを開き、次のコマンドを実行して、AWS CLI クライアントが動作していることを確認します。

```
aws help
```

### Note

David のデフォルト環境では、David コマンドで作成したデフォルトのプロファイルから、aws configure ユーザー認証情報を使用します。詳細については、『[AWS](#)



[Command Line Interface ユーザーガイド](#)』の「AWS Command Line Interface の設定。」を参照してください。

- David は次のコマンドを実行してロールの切り替えプロセスを開始し、本番稼働用アカウントで UpdateApp ロールに切り替えます。ロールを作成した管理者から、ロールの ARN を取得しました。コマンドでは、セッション名も提供する必要もあります。セッション名には任意のテキストを選択できます。

```
aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/UpdateApp" --role-session-name "David-ProdUpdate"
```

出力には次のように表示されます。

```
{
  "Credentials": {
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEEeYjs1M2FUIGIJx9tQqNMBEXAMPLE
CvSryh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB51TYLto9dyBgSDy
EXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPy0j59pFA41NKCIkVgkREXAMPLEj1zxQ7y52gekeVEXAMPLEDiB9ST3Uuysg
sKdEXAMPLE1TVastU1A0SKFEXAMPLEEiywCC/Cs8EXAMPLEEpZg0s+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEsnf87e
NhyDHq6ikBQ==",
    "Expiration": "2014-12-11T23:08:07Z",
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

- 出力の [Credentials] (認証情報) セクションには、3 つの必要な情報が表示されます。

- AccessKeyId
- SecretAccessKey
- SessionToken

以降の呼び出しでこれらのパラメータを使用するには、AWS CLI 環境を設定する必要があります。認証情報を設定するさまざまな方法の詳細については、「[AWS Command Line Interface の設定](#)」を参照してください。セッショントークンの取得をサポートしていないため、aws

configure コマンドを使用することはできません。ただし、設定ファイルに手動で情報を入力することができます。これらは有効期限が比較的短い一時的な認証情報なので、現在のコマンドラインセッションの環境に追加するのが最も簡単です。

4. David は、環境に 3 つの値を追加するため、前のステップの出力を切り取り、次のコマンドに貼り付けます。セッショントークン出力の改行の問題に対応するため、シンプルなテキストエディターで切り取りと貼り付けを行います。ここではわかりやすくするために改行して表示されていますが、1 行の長い文字列として入力する必要があります。

#### Note

以下の例では「set」が環境変数を作成するためのコマンドとなっている Windows 環境で指定されたコマンドを示しています。Linux または Mac コンピュータでは、代わりに「export」コマンドを使用します。この例における他の部分はすべて、3 つの環境で有効です。

Windows Powershell 用ツールの使用方法の詳細については、[IAM ロールへの切り替え \(Tools for Windows PowerShell\)](#) を参照してください。

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
Ryh0FW7jEXAMPLEw+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEEmj4wxS04L/
uZEXAMPLEcihzFB51TYLto9dyBgSDyEXA
MPLEKEY9/
g7QRUhZp4bqbEXAMPLENwGPy0j59pFA41NKCIkVgkREXAMPLEj1zxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1A0SKFEXAMPLEiYWCC/Cs8EXAMPLEpZg0s+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLENhykxiHen
DHq6ikBQ==
```

この時点で、次のコマンドはいずれも、これらの認証情報によって識別されたロールのアクセス権限で実行されます。David の場合は、UpdateApp ロールです。

5. Production アカウントのリソースにアクセスするコマンドを実行します。この例では、David は次のコマンドを使用して S3 バケットのコンテンツの一覧を示します。

```
aws s3 ls s3://productionapp
```

Amazon S3 バケット名は共通にユニークであるため、バケットを所有するアカウント ID を指定する必要はありません。その他の AWS サービスのリソースにアクセスするには、そのリソースを参照するために必要なコマンドと構文について記載されている対象サービスの AWS CLI のドキュメントを参照してください。

## AssumeRole (AWS API) の使用

David は、コードから本番稼働用アカウントを更新する必要がある場合、AssumeRole を呼び出し、UpdateApp ロールを取得します。この呼び出しは、David が本番稼働用アカウントにある productionapp バケットにアクセスするために使用できる一時的な認証情報を返します。これらの認証情報を使用して、David は productionapp バケットを更新する API 呼び出しを行うことができます。ただし、開発用アカウントのパワーユーザーアクセス許可を持っていても、本番稼働用アカウントの他のリソースにアクセスする API コールを行うことはできません。

ロールを割り当てるには

1. デイビッドは、アプリケーションの一部として AssumeRole を呼び出します。ユーザーは、UpdateApp ARN: `arn:aws:iam::999999999999:role/UpdateApp` と指定する必要があります。

AssumeRole 呼び出しからのレスポンスには、AccessKeyId と SecretAccessKey を含む一時的な認証情報が含まれています。また、認証情報の有効期限を示す Expiration 時間も含まれており、新しい認証情報をリクエストする必要があります。

2. 一時的な認証情報を使用して、デイビッドは `s3:PutObject` 呼び出しを行い、productionapp バケットを更新します。ユーザーは、AuthParams パラメータとして API 呼び出しに認証情報を渡します。ロールの一時的な認証情報は productionapp バケットでの読み取りおよび書き込み専用のアクセスであるため、本番稼働用アカウントでの他のアクションは拒否されます。

コードの例 (Python を使用) については、「[IAM ロール \(AWS API\) の切り替え](#)」を参照してください。

## 関連リソース

- IAM ユーザーとユーザーグループの詳細については、「[IAM ID \(ユーザー、ユーザーグループ、ロール\)](#)」を参照してください。

- Amazon S3 バケットの詳細については、Amazon Simple Storage Service ユーザーガイドの「[バケットの作成](#)」を参照してください。
- 信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

## まとめ

これでクロスアカウント API アクセスのチュートリアルが終了しました。他のアカウントと信頼を構築するロールを作成し、信頼されたエンティティが実行できるアクションを定義しました。次に、どの IAM ユーザーがロールを取得できるのかを制御するグループポリシーを修正しました。その結果、開発アカウントのデベロッパーは、一時的な認証情報を使用して本番稼働用アカウントにある productionapp バケットを更新することができます。

## IAM チュートリアル: はじめてのカスタマー管理ポリシーの作成とアタッチ

このチュートリアルでは、AWS Management Console を使用して[カスタマー管理ポリシー](#)を作成し、AWS アカウントのすべての IAM ユーザーにアタッチします。作成するポリシーでは、IAM テストユーザーに、AWS Management Console に直接サインインする読み取り専用アクセス許可を付与します。

このワークフローに 3 つの基本的なステップがあります:

### [ステップ 1: ポリシーを作成する](#)

デフォルトでは IAM ユーザーにはアクセス許可はありません。ユーザーは許可されるまで、AWS マネジメントコンソールにアクセスしたり、その中のデータを管理したりすることはできません。このステップでは、アタッチされたユーザーにコンソールへのサインインを許可するカスタマー管理ポリシーを作成します。

### [ステップ 2: ポリシーのアタッチ](#)

ユーザーにポリシーをアタッチする場合、ユーザーはポリシーに関連付けられているすべてのアクセス権限を継承します。このステップでは、テストユーザーに新しいポリシーをアタッチします。

### [ステップ 3: ユーザーアクセスのテスト](#)

ポリシーがアタッチされると、ユーザーとしてサインインし、ポリシーをテストできます。

## 前提条件

このチュートリアルステップを実行するには、以下を持っている必要があります:

- 管理者アクセス許可を持つ IAM ユーザーとしてサインインできる AWS アカウント。
- 以下のような権限またはメンバーシップが割り当てられていないテスト IAM ユーザー。

[User name] (ユーザー名)	[Group] (グループ)	許可
PolicyUser	<なし>	<なし>

### ステップ 1: ポリシーを作成する


このステップで作成するカスタマー管理ポリシーでは、アタッチされたユーザーに、AWS Management Console にサインインして IAM データにアクセスする読み取り専用アクセス許可を付与します。

テストユーザーのポリシーを作成するには

1. 管理者権限を持つユーザーを使用して、<https://console.aws.amazon.com/iam/>で IAM コンソールにサインインします。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. コンテンツペインで、[ポリシーの作成] を選択します。
4. [JSON] オプションを選択し、以下の JSON ポリシードキュメントからテキストをコピーします。このテキストを [JSON] ボックスに貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": [
      "iam:GenerateCredentialReport",
      "iam:Get*",
      "iam:List*"
    ],
    "Resource": "*"
  } ]
}
```

5. **ポリシーの検証**中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。

 Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual editor] タブで [Review policy] を変更または選択した場合、IAM はポリシーを再構成してビジュアルエディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

6. [確認および作成] ページで、ポリシー名として「**UsersReadOnlyAccessToIAMConsole**」と入力します。ポリシーによって割り当てられたアクセス許可を確認し、[ポリシーの作成] を選択して作業を保存します。

新しいポリシーが管理ポリシーの一覧に表示され、アタッチの準備ができます。

## ステップ 2: ポリシーのアタッチ

次に、作成したポリシーをテスト IAM ユーザーにアタッチします。

テストユーザーにポリシーをアタッチするには

1. IAM コンソールのナビゲーションペインから、[ポリシー] を選択します。
2. ポリシーリストの上部にある検索ボックスに、ポリシーが表示されるまで「**UsersReadOnlyAccessToIAMConsole**」と入力します。次に、リストの [UsersReadOnlyAccessToIAMConsole] の横にあるラジオボタンをオンにします。
3. [Actions] (アクション) ボタンを選択して、[Attach] (アタッチ) を選択します。
4. IAM エンティティでは、ユーザーをフィルタリングするオプションを選択します。
5. 検索ボックスで、そのユーザーがリストに表示されるまで「**PolicyUser**」と入力します。次に、リスト内のそのユーザーの横にあるチェックボックスをオンにします。
6. [ポリシーのアタッチ] を選択します。

これで IAM テストユーザーにポリシーがアタッチされました。これは、ユーザーが読み取り専用アクセス許可で IAM コンソールにアクセスできることを意味します。

## ステップ 3: ユーザーアクセスのテスト

このチュートリアルでは、テストユーザーとしてサインインしてアクセスをテストし、ユーザーの体験を確認できるようにすることをお勧めします。

テストユーザーにサインインしてアクセスをテストするには

1. PolicyUser テストユーザーを使用して、<https://console.aws.amazon.com/iam/>で IAM コンソールにサインインします。
2. コンソールのページを参照して、新しいユーザーまたはグループを作成します。PolicyUser はデータを表示できますが、IAM データを作成したり既存のデータを変更したりできなくなっています。

## 関連リソース

関連情報については、以下のリソースを参照してください。

- [管理ポリシーとインラインポリシー](#)
- [AWS Management Console への IAM ユーザーアクセスのコントロール](#)

## まとめ

これで、カスタマー管理ポリシーを作成してアタッチするために必要なすべてのステップが完了しました。その結果、テストアカウントで IAM コンソールにサインインして、ユーザーのエクスペリエンスがどのように表示されるかを確認できます。

## IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する

属性ベースのアクセス制御 (ABAC) は、属性に基づいて権限を定義する認可戦略です。これらの属性は、AWS でタグと呼ばれています。タグは、IAM エンティティ (ユーザーまたはロール) を含む IAM リソースと、AWS リソースにアタッチできます。タグ条件キーを使用して、タグに基づいてプリンシパルにアクセス許可を付与するポリシーを定義できます。タグを使用して AWS リソースへのアクセスを制御すると、AWS ポリシーの変更を減らしてチームとリソースを拡張できます。ABAC ポリシーは、個々のリソースをリストする従来の AWS ポリシーよりも柔軟性があります。ABAC の詳細と、従来のポリシーに対する利点については、「[AWS の ABAC とは](#)」を参照してください。



**Note**

セッションタグごとに 1 つの値を渡す必要があります。AWS Security Token Service は、複数値を持つセッションタグをサポートしていません。

## トピック

- [チュートリアル概要](#)
- [前提条件](#)
- [ステップ 1: テストユーザーを作成する](#)
- [ステップ 2: ABAC ポリシーを作成する](#)
- [ステップ 3: ロールを作成する](#)
- [ステップ 4: シークレットの作成をテストする](#)
- [ステップ 5: シークレットの表示をテストする](#)
- [ステップ 6: スケーラビリティをテストする](#)
- [ステップ 7: シークレットの更新と削除をテストする](#)
- [\[概要\]](#)
- [関連リソース](#)
- [IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)

## チュートリアル概要

このチュートリアルでは、プリンシパルタグを持つ IAM ロールが、一致するタグを持つリソースにアクセスすることを許可するポリシーを作成およびテストする方法を示します。プリンシパルが AWS にリクエストを行うと、プリンシパルとリソースタグが一致するかどうかに基づいてアクセス許可が付与されます。この戦略により、個人は自分のジョブに必要な AWS リソースのみを表示または編集できます。

## シナリオ

Example Corporation という大企業のリーダー開発者であり、経験豊富な IAM 管理者であるとして、IAM ユーザー、ロール、ポリシーの作成と管理に精通しています。開発エンジニアと品質保証チームのメンバーが、必要なリソースにアクセスできるようにする必要があります。また、企業の成長に合わせてスケールする戦略も必要です。

AWS リソースタグと IAM ロールプリンシパルタグを使用して、AWS Secrets Manager で始まるサービスをサポートする ABAC 戦略を実装することを選択します。タグに基づく認証をサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照してください。各サービスのアクションおよびリソースを含むポリシーで使用できるタグ付け条件キーについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。[セッションタグ](#)を AWS に渡すよう SAML ベースまたはウェブ ID プロバイダーを設定できます。従業員が AWS にフェデレートすると、その属性が AWS で結果のプリンシパルに適用されます。その後、ABAC を使用して、これらの属性に基づいてアクセス許可を許可または拒否できます。SAML フェデレーテッド ID でのセッションタグの使用とこのチュートリアルとの違いについては、「[IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)」を参照してください。

エンジニアリングおよび品質保証チームのメンバーは、[Pegasus] または [Unicorn] のいずれかのプロジェクトに参加しています。次の 3 文字のプロジェクトおよびチームタグ値を選択します。

- access-project = [Pegasus] プロジェクトの場合は peg
- access-project = [Unicorn] プロジェクト場合は uni
- access-team = エンジニアリングチームの場合は eng
- access-team = 品質保証チームの場合は qas

また、カスタム AWS 請求レポートを有効にするために、cost-center コスト配分タグを要求することを選択します。詳細については、AWS Billing and Cost Management ユーザーガイドの [[コスト配分タグの使用](#)] をご参照ください。

## 主要な機能の概要

- 従業員は、ユーザー認証情報を使用してサインインし、チームとプロジェクトの IAM ロールを引き受けます。会社に独自の ID システムがある場合は、従業員が IAM ユーザーなしでロールを引き受けることができるようにフェデレーションを設定できます。詳細については、「[IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)」を参照してください。
- すべてのロールに同じポリシーがアタッチされます。アクションは、タグに基づいて許可または拒否されます。
- 従業員は、ロールに適用される同じタグをリソースにアタッチする場合に限り、新しいリソースを作成できます。これにより、従業員は作成後にリソースを表示できます。管理者は、新しいリソースの ARN を使用してポリシーを更新する必要がなくなりました。
- 従業員は、プロジェクトに関係なく、チームが所有するリソースを読み取ることができます。
- 従業員は、自分のチームとプロジェクトが所有するリソースを更新および削除できます。

- IAM 管理者は、新しいプロジェクトに新しいロールを追加できます。新しい IAM ユーザーを作成してタグ付けし、適切なロールへのアクセスを許可できます。管理者は、新しいプロジェクトまたはチームメンバーをサポートするためにポリシーを編集する必要はありません。

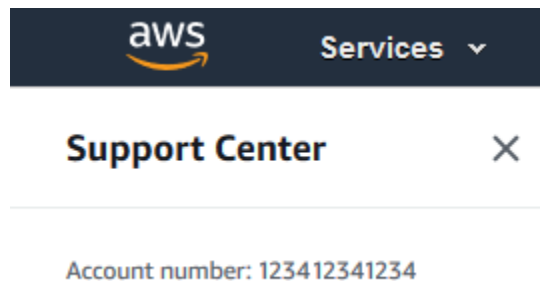
このチュートリアルでは、各リソースにタグ付けし、プロジェクトロールにタグ付けして、前述の動作を許可するポリシーをロールに追加します。結果として得られるポリシーではロール Create、Read、Update、および Delete は、同じプロジェクトタグとチームタグが付けられたリソースへのアクセスを許可します。このポリシーでは、同じチームでタグ付けされたリソースに対するクロスプロジェクト Read アクセスも許可されます。

## 前提条件

このチュートリアルのステップを実行するには、以下を持っている必要があります:

- 管理者アクセス許可を持つユーザーとしてサインインできる AWS アカウント。
- ステップ 3 でロールを作成するために使用する 12 桁のアカウント ID。

AWS Management Console で AWS アカウント ID 番号を確認するには、ナビゲーションバーの右上にある [サポート] を選択してから、[サポートセンター] を選択します。左側のナビゲーションペインにアカウント番号 (ID) が表示されます。



- AWS Management Console での IAM ユーザー、ロール、ポリシーの作成と編集の経験。ただし、IAM 管理プロセスの記憶にサポートが必要な場合は、このチュートリアルでは、ステップバイステップの手順を参照できるリンクを提供します。

## ステップ 1: テストユーザーを作成する

テストでは、同じタグでロールを引き受けるアクセス許可を持つ 4 人の IAM ユーザーを作成します。これにより、チームにユーザーを追加しやすくなります。ユーザーにタグを付けると、ユーザーは正しいロールを引き受けるためのアクセス権を自動的に取得します。ユーザーが 1 つのプロジェクトとチームのみで作業する場合、ロールの信頼ポリシーにユーザーを追加する必要はありません。

1. そのために、次のカスタマー管理ポリシーを `access-assume-role` という名前で作成します。JSON ポリシーの作成の詳細については、「[IAM ポリシーの作成](#)」を参照してください。

ABAC ポリシー: すべての ABAC ロールを引き受けるが、ユーザーとロールタグが一致する場合のみ

次のポリシーでは、ユーザーは、`access-` 名前プレフィックスを持つアカウント内の任意のロールを引き受けることを許可します。ロールには、ユーザーと同じプロジェクト、チーム、およびコストセンタータグでタグ付けする必要があります。

このポリシーを使用するには、イタリック体のプレースホルダーテキストをお客様のアカウント情報と置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TutorialAssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::account-ID-without-hyphens:role/access-*",
      "Condition": {
        "StringEquals": {
          "iam:ResourceTag/access-project": "${aws:PrincipalTag/access-project}",
          "iam:ResourceTag/access-team": "${aws:PrincipalTag/access-team}",
          "iam:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
        }
      }
    }
  ]
}
```

このチュートリアルを多数のユーザーにスケールするには、ポリシーをグループにアタッチし、各ユーザーをグループに追加します。詳細については、[IAM ユーザーグループの作成](#)および[IAM グループへのユーザーの追加と削除](#)を参照してください。

2. 以下の IAM ユーザーを作成し、`access-assume-role` アクセス許可ポリシーをアタッチします。[AWS Management Console へのユーザーアクセスを提供] が選択されていることを確認し

てから、次のタグを追加してください。新しいユーザーの作成とタグ付けの詳細については、「[IAM ユーザーの作成 \(コンソール\)](#)」を参照してください。

## ABAC ユーザー

[User name] (ユーザー名)	ユーザータグキー	ユーザータグ値
access-Arn timer-peg-engine	access-project	peg
	access-team	eng
	cost-center	987654
access-Mary-peg-qas	access-project	peg
	access-team	qas
	cost-center	987654
access-Saanvi-university-engine	access-project	uni
	access-team	eng
	cost-center	123456
access-Carlos-university-qas	access-project	uni
	access-team	qas
	cost-center	123456

## ステップ 2: ABAC ポリシーを作成する

**access-same-project-team** という名前の次のポリシーを作成します。このポリシーは、後のステップでロールに追加します。JSON ポリシーの作成の詳細については、「[IAM ポリシーの作成](#)」を参照してください。

このチュートリアルに適応できるその他のポリシーについては、以下のページを参照してください。

- [IAM プリンシパルへのアクセスの制御](#)

- [Amazon EC2: プログラムおよびコンソールでユーザーがタグ付けされた EC2 インスタンスの起動や停止を許可する](#)
- [EC2: 一致するプリンシパルタグとリソースタグに基づいてインスタンスを開始または停止する](#)
- [EC2: タグに基づくインスタンスの開始または停止](#)
- [IAM: 特定のタグを持つロールを引き受ける](#)

ABAC ポリシー: プリンシパルとリソースタグが一致する場合にのみ Access Secrets Manager リソースにアクセスする

次のポリシーでは、リソースの作成、読み取り、編集、削除をプリンシパルに許可しますが、それらのリソースにプリンシパルと同じキーと値のペアがタグ付けされている場合に限りです。プリンシパルがリソースを作成するときに、プリンシパルのタグに一致する値を持つ access-project、access-team、および cost-center タグを追加する必要があります。このポリシーでは、オプションの Name または OwnedBy タグの追加も許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllActionsSecretsManagerSameProjectSameTeam",
      "Effect": "Allow",
      "Action": "secretsmanager:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/access-project": "${aws:PrincipalTag/access-project}",
          "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}",
          "aws:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "access-project",
            "access-team",
            "cost-center",
            "Name",
            "OwnedBy"
          ]
        }
      },
      "StringEqualsIfExists": {
```

```

        "aws:RequestTag/access-project": "${aws:PrincipalTag/access-project}",
        "aws:RequestTag/access-team": "${aws:PrincipalTag/access-team}",
        "aws:RequestTag/cost-center": "${aws:PrincipalTag/cost-center}"
    }
}
},
{
    "Sid": "AllResourcesSecretsManagerNoTags",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetRandomPassword",
        "secretsmanager:ListSecrets"
    ],
    "Resource": "*"
},
{
    "Sid": "ReadSecretsManagerSameTeam",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:Describe*",
        "secretsmanager:Get*",
        "secretsmanager:List*"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}"
        }
    }
},
{
    "Sid": "DenyUntagSecretsManagerReservedTags",
    "Effect": "Deny",
    "Action": "secretsmanager:UntagResource",
    "Resource": "*",
    "Condition": {
        "ForAnyValue:StringLike": {
            "aws:TagKeys": "access-*"
        }
    }
},
{
    "Sid": "DenyPermissionsManagement",
    "Effect": "Deny",

```



```
    "Action": "secretsmanager:*Policy",
    "Resource": "*"
  }
]
```

## このポリシーで行うこと

- AllActionsSecretsManagerSameProjectSameTeam ステートメントは、リソースタグがプリンシパルタグと一致する場合のみ、関連するすべてのリソースでこのサービスのすべてのアクションを許可します。"Action": "secretsmanager:\*" をポリシーに追加することで、ポリシーは Secrets Manager が大きくなるにつれて大きくなります。Secrets Manager が新しい API オペレーションを追加する場合、そのアクションをステートメントに追加する必要はありません。ステートメントは、3つの条件ブロックを使用して ABAC を実装します。リクエストは、3つのブロックすべてが true を返す場合にのみ許可されます。
- 指定されたタグキーがリソースに存在し、その値がプリンシパルのタグと一致する場合、このステートメントの最初の条件ブロックは true を返します。このブロックは、一致しないタグ、またはリソースタグ付けをサポートしていないアクションに対して false を返します。このブロックで許可されていないアクションについては、「[AWS Secrets Manager のアクション、リソース、および条件キー](#)」を参照してください。このページは、[\[Secret\] リソースタイプ](#)で実行されるアクションが secretsmanager:ResourceTag/tag-key 条件キーをサポートしていることを示しています。一部の [Secrets Manager アクション](#)では、GetRandomPassword や ListSecrets など、そのリソースタイプをサポートしません。これらのアクションを許可するには、追加のステートメントを作成する必要があります。
- 2番目の条件ブロックは、リクエストで渡されたすべてのタグキーが指定されたリストに含まれている場合に true を返します。これは、StringEquals 条件演算子とともに ForAllValues を使用して行われます。キーまたはキーのセットのサブセットが渡されない場合、条件は true を返します。これにより、リクエストでタグを渡すことを許可しない Get\* オペレーションが可能になります。リクエストにリストにないタグキーが含まれている場合、条件は false を返します。リクエストで渡されるすべてのタグキーは、このリストのメンバーと一致する必要があります。詳細については、「[複数値のコンテキストキー](#)」を参照してください。
- 3番目の条件ブロックは、リクエストでタグの受け渡しをサポートされている場合、3つすべてのタグが存在する場合、およびプリンシパルタグの値に一致する場合に true を返します。このブロックは、リクエストがタグの受け渡しをサポートしていない場合も true を返します。これは、条件演算子の [...IfExists](#) のおかげです。ブロックは、それをサポートするアクション中に渡されたタグがない場合、またはタグのキーと値が一致しない場合、false を返します。

- AllResourcesSecretsManagerNoTags ステートメントでは、最初のステートメントでは許可されていない GetRandomPassword および ListSecrets アクションを許可します。
- ReadSecretsManagerSameTeam ステートメントは、プリンシパルがリソースと同じアクセスチームタグでタグ付けされている場合、読み取り専用オペレーションを許可します。これは、プロジェクトまたはコストセンタータグに関係なく許可されます。
- DenyUntagSecretsManagerReservedTags ステートメントは、Secrets Manager から「access-」で始まるキーを持つタグを削除するリクエストを拒否します。これらのタグはリソースへのアクセスを制御するために使用されるため、タグを削除するとアクセス許可が削除される可能性があります。
- DenyPermissionsManagement ステートメントは、Secrets Manager リソースベースのポリシーを作成、編集、または削除することを拒否します。これらのポリシーは、シークレットのアクセス許可を変更するために使用できます。

#### Important

このポリシーでは、サービスに対するすべてのアクションを許可する戦略が使用されますが、アクセス許可を変更するアクションは明示的に拒否されます。アクションを拒否すると、プリンシパルがそのアクションの実行を許可する他のポリシーよりも優先されます。これにより、意図しない結果が生じる可能性があります。ベストプラクティスとして、明示的な拒否は、そのアクションを許可する状況がない場合にのみ使用します。それ以外の場合、個々のアクションのリストを許可し、不要なアクションはデフォルトで拒否されます。

## ステップ 3: ロールを作成する

以下の IAM ロールを作成し、前のステップで作成した **access-same-project-team** ポリシーをアタッチします。IAM ロールの作成の詳細については、「[IAM ユーザーにアクセス許可を委任するロールの作成](#)」を参照してください。IAM ユーザーとロールの代わりにフェデレーションを使用する場合は、「[IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)」を参照してください。

## ABAC ロール

ジョブ関数	ロール名	ロールタグ	ロールの説明
プロジェクト Pegasus エンジニアリング	access-peg-engineering	access-project = peg  access-team = eng  cost-center = 987654	エンジニアがすべてのエンジニアリングリソースを読み取り、Pegasus エンジニアリングリソースを作成および管理できるようにします。
プロジェクト Pegasus 品質保証	access-peg-quality-assurance	access-project = peg  access-team = qas  cost-center = 987654	QA チームがすべての QA リソースを読み取り、すべての Pegasus QA リソースを作成および管理できるようにします。
プロジェクト Unicorn エンジニアリング	access-uni-engineering	access-project = uni  access-team = eng  cost-center = 123456	エンジニアがすべてのエンジニアリングリソースを読み取り、Unicorn のエンジニアリングリソースを作成および管理できるようにします。
プロジェクト Unicorn 品質保証	access-uni-quality-assurance	access-project = uni	QA チームがすべての QA リソースを読み取り、すべての Unicorn QA リソースを作成

ジョブ関数	ロール名	ロールタグ	ロールの説明
		access-team = qas	および管理できるようにします。
		cost-center = 123456	

## ステップ 4: シークレットの作成をテストする

ロールにアタッチされたアクセス許可ポリシーにより、従業員はシークレットを作成できます。これは、シークレットがプロジェクト、チーム、およびコストセンターでタグ付けされている場合のみ許可されます。ユーザーとしてサインインし、正しいロールを引き受け、Secrets Manager でアクティビティをテストすることで、アクセス許可が想定どおりに機能していることを確認します。

必要なタグがある場合とない場合のシークレットの作成をテストするには

1. IAM のユーザー、ロール、ポリシーを確認できるように、メインブラウザウィンドウで、管理者ユーザーとしてサインインしたままにします。テストには、ブラウザの匿名ウィンドウまたは別のブラウザを使用します。そこに、access-Arn timer-peg-eng IAM ユーザーをとしてサインインし、<https://console.aws.amazon.com/secretsmanager/> で Secrets Manager コンソールを開きます。
2. access-uni-engineering ロールへの切り替えを試みます。

このオペレーションは、access-Arn timer-peg-eng ユーザーおよび access-uni-engineering ロールで access-project および cost-center タグ値が一致しないため失敗します。

AWS Management Consoleでのロールの切り替えの詳細については、「[ロールの切り替え \(コンソール\)](#)」を参照してください。

3. access-peg-engineering ロールに切り替えます。
4. 以下の情報を使用して新しいシークレットを保存します。シークレットを保存する方法については、「AWS Secrets Manager ユーザーガイド」の「[基本的なシークレットの作成](#)」を参照してください。

**⚠ Important**

Secrets Manager は、Secrets Manager と連携する追加の AWS サービスに対する権限がないというアラートを表示します。例えば、Amazon RDS データベースの認証情報を作成するには、RDS インスタンス、RDS クラスター、Amazon Redshift クラスターを記述するアクセス許可が必要です。このチュートリアルではこれらの特定の AWS サービスを使用していないため、これらのアラートは無視してかまいません。

1. [Select secret type (シークレットタイプの選択)] セクションで、[Other type of secrets (他の種類のシークレット)] を選択します。2 つのテキストボックスに、「test-access-key」と「test-access-secret」と入力します。
2. [Secret name (シークレット名)] フィールドに「test-access-peg-eng」と入力します。
3. 次の表から異なるタグの組み合わせを追加し、想定される動作を確認します。
4. [Store (保存)] を選択して、シークレットの作成を試みます。ストレージに障害が発生した場合は、前の Secrets Manager コンソールページに戻り、以下のテーブルから次のタグセットを使用します。最後のタグセットは許可され、シークレットを正常に作成します。

**test-access-peg-eng** ロールの ABAC タグの組み合わせ

access-project タグ値	access-team タグ値	cost-center タグ値	追加のタグ	予想される動作
(none)	(none)	(none)	(none)	access-project タグの値が peg のロールの値と一致しないため、拒否されました。
uni	eng	987654	(none)	access-project タグの値が peg のロールの値と一致しないため、拒否されました。
peg	qas	987654	(none)	access-team タグの値が eng のロールの値と一致しないため、拒否されました。

access-project タグ値	access-team タグ値	cost-center タグ値	追加のタグ	予想される動作
peg	eng	123456	(none)	cost-center タグの値が 987654 のロールの値と一致しないため、拒否されました。
peg	eng	987654	owner = Jane	3 つの必須タグがすべて存在し、その値がロールの値と一致しても、ポリシーで追加のタグ owner が許可されないため、拒否されます。
peg	eng	987654	Name = Jane	3 つの必須タグがすべて存在し、その値がロールの値と一致しているため許可されます。オプションの Name タグを含めることもできます。

- サインアウトし、以下の各ロールとタグ値について、この手順の最初の 3 つのステップを繰り返します。この手順の 4 番目のステップでは、欠落しているタグ、オプションのタグ、許可されていないタグ、無効なタグ値のセットをテストします。次に、必要なタグを使用して、次のタグと名前を持つシークレットを作成します。

#### ABAC ロールとタグ

[User name] (ユーザー名)	ロール名	シークレット名	シークレットタグ
access-Mary-peg-qas	access-peg-quality-assurance	test-access-peg-qas	access-project = peg  access-team = qas  cost-center = 987654

[User name] (ユーザー名)	ロール名	シークレット名	シークレットタグ
access-Saanvi-uni-eng	access-uni-engineering	test-access-uni-eng	access-project = uni access-team = eng cost-center = 123456
access-Carlos-uni-qas	access-uni-quality-assurance	test-access-uni-qas	access-project = uni access-team = qas cost-center = 123456

## ステップ 5: シークレットの表示をテストする

各ロールにアタッチしたポリシーにより、従業員はプロジェクトに関係なく、チーム名でタグ付けされたシークレットを表示できます。Secrets Manager でロールをテストして、アクセス許可が想定どおりに動作していることを確認します。

必要なタグがある場合とない場合のシークレットの表示をテストするには

1. 次のいずれかの IAM ユーザーとしてサインインします。

- access-Arn timer-peg-eng
- access-Mary-peg-qas
- access-Saanvi-uni-eng
- access-Carlos-uni-qas

2. 一致するロールに切り替えます。

- access-peg-engineering



- access-peg-quality-assurance
- access-uni-engineering
- access-uni-quality-assurance

AWS Management Console でのロールの切り替えの詳細については、「[ロールの切り替え \(コンソール\)](#)」を参照してください。

3. 左側のナビゲーションペインで、メニューアイコンを選択してメニューを展開し、[Secrets (シークレット)]を選択します。
4. 現在のロールに関係なく、テーブルに 4 つのシークレットすべてが表示されます。これは、access-same-project-team という名前のポリシーですべてのリソースに対して secretsmanager:ListSecrets アクションが許可されるためです。
5. いずれかのシークレットの名前を選択します。
6. シークレットの詳細ページで、ロールのタグによって、ページのコンテンツを表示できるかが決まります。ロールの名前とシークレットの名前を比較します。同じチーム名を共有する場合、access-team タグは一致します。一致しない場合、アクセスは拒否されます。

#### 各ロールの ABAC シークレット表示動作

ロール名	シークレット名	予想される動作
access-peg-engineering	test-access-peg-eng	許可
	test-access-peg-qas	拒否
	test-access-uni-eng	許可
	test-access-uni-qas	拒否
access-peg-quality-assurance	test-access-peg-eng	拒否
	test-access-peg-qas	許可
	test-access-uni-eng	拒否
	test-access-uni-qas	許可
access-uni-engineering	test-access-peg-eng	許可

ロール名	シークレット名	予想される動作
access-uni-quality-assurance	test-access-peg-qas	拒否
	test-access-uni-eng	許可
	test-access-uni-qas	拒否
	test-access-peg-eng	拒否
	test-access-peg-qas	許可
	test-access-uni-eng	拒否
	test-access-uni-qas	許可

7. ページの上部にあるパンくずリストから、[Secrets (シークレット)] を選択してシークレットのリストに戻ります。異なるロールを使用して、この手順のステップを繰り返し、各シークレットを表示できるかどうかをテストします。

## ステップ 6: スケーラビリティをテストする

ロールベースのアクセスコントロール (RBAC) よりも属性ベースのアクセスコントロール (ABAC) を使用する重要な理由は、スケーラビリティです。会社が新しいプロジェクト、チーム、または人員を AWS に追加する場合、ABAC 主導のポリシーを更新する必要はありません。例えば、Example Company が [Centaur] という名前の新しいプロジェクトに資金を提供しているとします。Saanvi Sarkar というエンジニアが [Centaur] のリードエンジニアとなり、[ユニコーン] プロジェクトに取り組んでいます。また、Saanvi は Peg プロジェクトの作業も確認します。また、Nikhil Jayashankar をはじめ、[Centaur] プロジェクトのみに取り組む新しく採用されたエンジニアもいます。

新しいプロジェクトを AWS に追加するには

1. IAM 管理者ユーザーとしてサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>)。
2. 左側のナビゲーションペインで、[ロール] を選択し、access-cen-engineering という名前の IAM ロールを追加します。access-same-project-team アクセス許可ポリシーをロールにアタッチし、次のロールタグを追加します。

- access-project = cen

- access-team = eng
  - cost-center = 101010
3. 左側のナビゲーションペインで、[ユーザー] を選択します。
  4. access-Nikhil-cen-eng という名前の新しいユーザーを追加し、access-assume-role という名前のポリシーをアタッチして、次のユーザータグを追加します。
    - access-project = cen
    - access-team = eng
    - cost-center = 101010
  5. 「[ステップ 4: シークレットの作成をテストする](#)」 および 「[ステップ 5: シークレットの表示をテストする](#)」 の手順を使用します。別のブラウザウィンドウで、Nikhil が [Centaur] エンジニアリングシークレットのみを作成できること、および Nikhil がすべてのエンジニアリングシークレットを表示できることをテストします。
  6. 管理者としてサインインしたメインブラウザウィンドウで、ユーザー access-Saanvi-uni-eng を選択します。
  7. [Permissions (アクセス許可)] タブで、[access-assume-role] アクセス許可ポリシーを削除します。
  8. access-assume-specific-roles という名前の次のインラインポリシーを追加します。ユーザーへのインラインポリシーの追加の詳細については、「[ユーザーまたはロールのインラインポリシーを埋め込むには \(コンソール\)](#)」を参照してください。

#### ABAC ポリシー: 特定のロールのみ継承する

このポリシーでは、Saanvi が Pegasus および Centaur プロジェクトのエンジニアリングロールを引き受けることを許可します。IAM は複数値タグをサポートしていないため、このカスタムポリシーを作成する必要があります。Saanvi のユーザーに access-project = peg および access-project = cen のタグを付けることはできません。さらに、AWS 認証モデルが両方の値に一致することはできません。詳細については、「[IAM および AWS STS でのタグ付けの規則](#)」を参照してください。代わりに、引き受けることができる 2 つのロールを手動で指定する必要があります。

このポリシーを使用するには、イタリック体のプレースホルダーテキストをお客様のアカウント情報と置き換えます。

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Sid": "TutorialAssumeSpecificRoles",
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": [
      "arn:aws:iam::account-ID-without-hyphens:role/access-peg-
engineering",
      "arn:aws:iam::account-ID-without-hyphens:role/access-cen-
engineering"
    ]
  }
]
```

9. 「[ステップ 4: シークレットの作成をテストする](#)」および「[ステップ 5: シークレットの表示をテストする](#)」の手順を使用します。別のブラウザウィンドウで、Saanvi が両方のロールを引き受けることができることを確認します。ロールのタグに応じて、プロジェクト、チーム、およびコストセンターのみのシークレットを作成できることを確認します。また、作成したばかりのシークレットを含め、エンジニアリングチームが所有するすべてのシークレットに関する詳細を表示できることを確認します。

## ステップ 7: シークレットの更新と削除をテストする

ロールにアタッチされた `access-same-project-team` ポリシーにより、従業員はプロジェクト、チーム、コストセンターでタグ付けされたシークレットを更新および削除できます。Secrets Manager でロールをテストして、アクセス許可が想定どおりに動作していることを確認します。

必要なタグの有無にかかわらず、シークレットの更新と削除をテストするには

1. 次のいずれかの IAM ユーザーとしてサインインします。
  - `access-Arn timer-peg-eng`
  - `access-Mary-peg-qas`
  - `access-Saanvi-uni-eng`
  - `access-Carlos-uni-qas`
  - `access-Nikhil-cen-eng`
2. 一致するロールに切り替えます。

- access-peg-engineering
- access-peg-quality-assurance
- access-uni-engineering
- access-peg-quality-assurance
- access-cen-engineering

AWS Management Console でのロールの切り替えの詳細については、「[ロールの切り替え \(コンソール\)](#)」を参照してください。

3. ロールごとに、シークレットの説明を更新し、次のシークレットを削除してみてください。詳細については、「AWS Secrets Manager ユーザーガイド」の「[シークレットの変更](#)」および「[シークレットの削除と復元](#)」を参照してください。

各ロールの ABAC シークレットの更新と削除動作

ロール名	シークレット名	予想される動作
access-peg-engineering	test-access-peg-eng	許可
	test-access-uni-eng	拒否
	test-access-uni-qas	拒否
access-peg-quality-assurance	test-access-peg-qas	許可
	test-access-uni-eng	拒否
access-uni-engineering	test-access-uni-eng	許可
	test-access-uni-qas	拒否
access-peg-quality-assurance	test-access-uni-qas	許可

## [概要]

これで、属性ベースのアクセスコントロール (ABAC) のタグを使用するために必要なすべてのステップが正常に完了しました。タグ付け戦略を定義する方法を学びました。この戦略をプリンシパルと

リソースに適用しました。Secrets Manager の戦略を適用するポリシーを作成して適用しました。また、ABAC は、新しいプロジェクトやチームメンバーを追加すると簡単にスケールできることも学びました。その結果、テストロールを使用して IAM コンソールにサインインし、AWS で ABAC のタグを使用する方法を体験できます。

#### Note

特定の条件下でのみアクションを許可するポリシーを追加しました。より広範なアクセス許可を持つユーザーまたはロールに別のポリシーを適用する場合、アクションはタグ付けを必要とするだけではないことがあります。例えば、AdministratorAccess AWS 管理ポリシーを使用してユーザーに完全な管理アクセス許可を付与しても、これらのポリシーではそのアクセスが制限されません。複数のポリシーが関係する場合のアクセス許可の決定方法の詳細については、「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。

## 関連リソース

関連情報については、以下のリソースを参照してください。

- [AWS の ABAC とは](#)
- [AWS グローバル条件コンテキストキー](#)
- [IAM ユーザーの作成 \(コンソール\)](#)
- [IAM ユーザーにアクセス許可を委任するロールの作成](#)
- [IAM リソースのタグ付け](#)
- [タグを使用した AWS リソースへのアクセスの制御](#)
- [ロールの切り替え \(コンソール\)](#)
- [IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)

アカウントのタグをモニタリングする方法については、「[サーバーレスワークフローおよび Amazon CloudWatch Events を使用して AWS リソースでタグの変更を監視する](#)」を参照してください。

## IAM チュートリアル: ABAC で SAML セッションタグを使用する

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。AWS では、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたは

ロール) を含む IAM リソースと、AWS リソースにアタッチできます。エンティティが AWS へのリクエストを行うために使用されると、エンティティはプリンシパルになり、プリンシパルにはタグが含まれます。

ロールを引き受けるとき、またはユーザーをフェデレートするときに [セッションタグ](#) を渡すこともできます。その後、タグ条件キーを使用して、タグに基づいてプリンシパルにアクセス許可を付与するポリシーを定義できます。タグを使用して AWS リソースへのアクセスを制御すると、AWS ポリシーの変更を減らしてチームとリソースを拡張できます。ABAC ポリシーは、個々のリソースをリストする従来の AWS ポリシーよりも柔軟性があります。ABAC の詳細と、従来のポリシーに対する利点については、「[AWS の ABAC とは](#)」を参照してください。

会社で SAML ベースの ID プロバイダー (IdP) を使用して企業ユーザー ID を管理する場合は、SAML 属性を使用して、AWS のきめ細かなアクセス制御を行うことができます。属性には、コストセンター ID、ユーザーの E メールアドレス、部門分類、およびプロジェクト割り当てを含めることができます。これらの属性をセッションタグとして渡すと、これらのセッションタグに基づいて AWS へのアクセスを制御できます。

SAML 属性をセッションプリンシパルに渡して [ABAC チュートリアル](#) を完了するには、このトピックに含まれる変更を使用して「[IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#)」のタスクを実行します。

## 前提条件

ABAC で SAML セッションタグを使用するステップを実行するには、次のものがが必要です。

- 特定の属性を持つテストユーザーを作成できる SAML ベースの IdP へのアクセス。
- 管理者アクセス許可を持つユーザーとしてサインインできること。
- AWS Management Console での IAM ユーザー、ロール、ポリシーの作成と編集の経験。ただし、IAM 管理プロセスの記憶にサポートが必要な場合は、ABAC チュートリアルでは、ステップバイステップの手順を参照できるリンクを提供します。
- IAM で SAML ベースの IdP を設定した経験があります。詳細および詳細 IAM ドキュメントへのリンクを表示するには、「[AssumeRoleWithSAML を使用したセッションタグの受け渡し](#)」を参照してください。

## ステップ 1: テストユーザーを作成する

[ステップ 1: テストユーザーを作成する](#) の手順に従います。ID はプロバイダーで定義されるため、従業員の IAM ユーザーを追加する必要はありません。



## ステップ 2: ABAC ポリシーを作成する

[ステップ 2: ABAC ポリシーを作成する](#) の手順に従って、IAM で指定した管理ポリシーを作成します。

## ステップ 3: SAML ロールを作成して設定する

SAML の ABAC チュートリアルを使用する場合は、追加のステップを実行し、ロールを作成します。SAML IdP を設定し、AWS Management Console アクセスを有効にする必要があります。詳細については、「[ステップ 3: ロールを作成する](#)」を参照してください。

### ステップ 3A: SAML ロールを作成する

SAML ID プロバイダーと、ステップ 1 で作成した test-session-tags ユーザーを信頼する 1 つのロールを作成します。ABAC チュートリアルでは、異なるロールタグを持つ個別のロールを使用します。SAML IdP からセッションタグを渡すため、必要なロールは 1 つだけです。SAML ベースのロールを作成する方法については、「[SAML 2.0 フェデレーション用のロールを作成する \(コンソール\)](#)」を参照してください。

ロールに access-session-tags という名前を付けます。access-same-project-team アクセス許可ポリシーをロールにアタッチします。ロールの信頼ポリシーを編集して、次のポリシーを使用します。ロールの信頼関係を編集する方法の詳細については、「[ロールの修正 \(コンソール\)](#)」を参照してください。

次のロール信頼ポリシーでは、SAML ID プロバイダーと test-session-tags ユーザーがロールを引き受けることができます。ロールを引き受けるときは、指定した 3 つのセッションタグを渡す必要があります。sts:TagSession アクションは、セッションタグを渡すことを許可するために必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSamlIdentityAssumeRole",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRoleWithSAML",
        "sts:TagSession"
      ],
      "Principal": {"Federated": "arn:aws:iam::123456789012:saml-provider/ExampleCorpProvider"},
    }
  ]
}
```

```
    "Condition": {
      "StringLike": {
        "aws:RequestTag/cost-center": "*",
        "aws:RequestTag/access-project": "*",
        "aws:RequestTag/access-team": [
          "eng",
          "qas"
        ]
      },
      "StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}
    }
  }
}
```

この AllowSamlIdentityAssumeRole ステートメントにより、エンジニアリングチームおよび品質保証チームのメンバーは、Example Corporation IdP から AWS にフェデレーションするときに、このロールを引き受けることができます。ExampleCorpProvider SAML プロバイダーは IAM で定義されています。管理者は、必要な 3 つのセッションタグを渡すように SAML アサーションをすでに設定しています。アサーションは追加のタグを渡すことができますが、これら 3 つは存在する必要があります。ID の属性は、*cost-center* タグと *access-project* タグに対して任意の値を持つことができます。ただし、*access-team* 属性値は、ID がエンジニアリングチームまたは品質保証チームにあることを示す *eng* か *qas* に一致する必要があります。

### ステップ 3B: SAML IdP を設定する

*cost-center*、*access-project*、*access-team* の各属性をセッションタグとして渡すように SAML IdP を設定します。詳細については、「[AssumeRoleWithSAML を使用したセッションタグの受け渡し](#)」を参照してください。

これらの属性をセッションタグとして渡すには、SAML アサーションに以下の要素を含めます。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:cost-center">
  <AttributeValue>987654</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-project">
  <AttributeValue>peg</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-team">
  <AttributeValue>eng</AttributeValue>
</Attribute>
```

## ステップ 3C: コンソールアクセスを有効にする

フェデレーティッド SAML ユーザーのコンソールアクセスを有効にします。詳細については、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする](#)」を参照してください。

## ステップ 4: シークレットの作成をテストする

access-session-tags ロールを使用して AWS Management Console にフェデレートします。詳細については、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする](#)」を参照してください。次に、[ステップ 4: シークレットの作成をテストする](#) の手順に従ってシークレットを作成します。属性を持つ異なる SAML ID を使用して、ABAC チュートリアルで示されているタグと一致させます。詳細については、「[ステップ 4: シークレットの作成をテストする](#)」を参照してください。

## ステップ 5: シークレットの表示をテストする

[ステップ 5: シークレットの表示をテストする](#) の手順に従って、前のステップで作成したシークレットを表示します。属性を持つ異なる SAML ID を使用して、ABAC チュートリアルで示されているタグと一致させます。

## ステップ 6: スケーラビリティをテストする

[ステップ 6: スケーラビリティをテストする](#) の指示に従って、スケーラビリティをテストします。これを行うには、SAML ベースの IdP に次の属性を持つ新しい ID を追加します。

- cost-center = 101010
- access-project = cen
- access-team = eng

## ステップ 7: シークレットの更新と削除をテストする

「[ステップ 7: シークレットの更新と削除をテストする](#)」の手順に従って、シークレットを更新および削除します。属性を持つ異なる SAML ID を使用して、ABAC チュートリアルで示されているタグと一致させます。

**⚠ Important**

課金されないように、作成したシークレットをすべて削除します。Secrets Manager の料金設定の詳細については、「[AWS Secrets Manager 料金設定](#)」を参照してください。

**概要**

これで、アクセス許可の管理に SAML セッションタグとリソースタグを使用するために必要なすべてのステップが正しく完了しました。

**📌 Note**

特定の条件下でのみアクションを許可するポリシーを追加しました。より広範なアクセス許可を持つユーザーまたはロールに別のポリシーを適用する場合、アクションはタグ付けを必要とするだけではないことがあります。例えば、AdministratorAccess AWS 管理ポリシーを使用してユーザーに完全な管理アクセス許可を付与しても、これらのポリシーではそのアクセスが制限されません。複数のポリシーが関係する場合のアクセス許可の決定方法の詳細については、「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。

## IAM チュートリアル: ユーザーに自分の認証情報および MFA 設定を許可する

ユーザーが [セキュリティ認証情報] ページで自分の多要素認証 (MFA) デバイスと認証情報を管理することを許可できます。AWS Management Console を使用して、認証情報 (アクセスキー、パスワード、デジタル署名用証明書、SSH パブリックキー) を設定したり、不要な認証情報を削除または非アクティブ化したり、ユーザーの MFA デバイスを有効にしたりできます。これは少数のユーザーにとっては便利ですが、ユーザーの数が増えるとすぐに、このタスクは時間がかかるようになる可能性があります。このチュートリアルでは、このようなベストプラクティスを管理者に負担を与えずに実現する方法を説明します。

このチュートリアルでは、AWS サービスへのアクセスをユーザーに許可する方法を示します。ただし、ユーザーが MFA を使用してサインインした場合に限ります。MFA デバイスでサインインしていない場合、ユーザーは他のサービスにアクセスできません。

このワークフローには 3 つの基本的な手順が含まれます。

## ステップ 1: MFA サインインを強制するポリシーを作成する

いくつかの IAM アクションを除く、すべてのアクションを禁止するカスタマー管理ポリシーを作成します。これらの例外により、ユーザーは [セキュリティ認証情報] ページで自分の認証情報を変更したり、MFA デバイスを管理したりすることができます。該当ページへのアクセスの詳細については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

## ステップ 2: テストユーザーグループにポリシーをアタッチする

メンバーが MFA でサインインすると、すべての Amazon EC2 アクションにフルアクセスできるグループを作成します。このようなユーザーグループを作成するには、AmazonEC2FullAccess という AWS 管理ポリシーと最初の手順で作成したカスタマー管理ポリシーの両方をアタッチします。

## ステップ 3: ユーザーアクセスをテストする

テストユーザーとしてサインインし、ユーザーが MFA デバイスを作成するまで Amazon EC2 へのアクセスがブロックされていることを確認します。ユーザーは、そのデバイスを使用してサインインできます。

## 前提条件

このチュートリアルステップを実行するには、以下を持っている必要があります:

- 管理者アクセス許可を持つ IAM ユーザーとしてサインインできる AWS アカウント。
- アカウント ID 番号。ステップ 1 のポリシーに入力します。

アカウント ID 番号を確認するには、ページ上部のナビゲーションバーで [サポート]、[サポートセンター] の順に選択します。アカウント ID 番号は、このページの [サポート] メニューの下で確認できます。

- [仮想 \(ソフトウェアベース\) MFA デバイス](#)、[FIDO セキュリティキー](#)、または [ハードウェアベース MFA デバイス](#)。
- ユーザーグループのメンバーであるテスト IAM ユーザーは次のとおりです。

ユーザーの作成		ユーザーグループアカウントを作成して設定する		
[User name] (ユーザー名)	その他の手順	ユーザーグループ名	メンバーとしてユーザーを追加する	その他の手順
MFAUser	[コンソールアクセスを有効にする - オプション] のオプションのみを選択し、パスワードを割り当てます。	EC2MFA	MFAUser	このユーザーグループへのポリシーのタッチや、アクセス許可の付与は行わないでください。

## ステップ 1: MFA サインインを強制するポリシーを作成する

まず、IAM ユーザー各自の認証情報と MFA デバイスの管理に必要な権限を除いて、すべてのアクセス権限を拒否する IAM カスタマー管理ポリシーを作成します。

1. 管理者認証情報を使用してユーザーとして AWS マネジメントコンソールにサインインします。IAM のベストプラクティスに準拠し、AWS アカウントのルートユーザー認証情報ではサインインしないでください。

### Important

IAM [ベストプラクティス](#)では、長期的な認証情報を持つIAMユーザーを使用するのではなく、一時的な認証情報を使用して AWS にアクセスするために、IDプロバイダーとのフェデレーションを使用することを人間ユーザーに求めることを推奨します。

2. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
3. ナビゲーションペインで ポリシーを選択してから ポリシーの作成を選択します。
4. [JSON] タブを選択し、以下の JSON ポリシードキュメントからテキストをコピーします。AWS: MFA で認証された IAM ユーザーが [\[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)
5. このポリシーに関するテキストを [JSON] ボックスに貼り付けます。ポリシーの検証中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[次へ] を選択します。

**Note**

いつでも [Visual エディタ] と [JSON] オプションを切り替えることができます。ただし、上記のポリシーには NotAction 要素が含まれていますが、これはビジュアルエディタではサポートされていません。このポリシーについては、[Visual editor (ビジュアルエディタ)] タブに通知が表示されます。[JSON] に戻り、このポリシーの操作を続行します。

このポリシー例では、初めて AWS Management Console にサインインする際のパスワードのリセットをユーザーに許可していません。新しいユーザーがサインインしてパスワードをリセットするまで、そのユーザーにアクセス許可を付与しないことを推奨しています。

6. [確認および作成] ページで、ポリシー名として「**Force\_MFA**」と入力します。ポリシーの説明の [タグ] 領域に **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.** と入力すると、オプションでタグのキーと値のペアをカスタマー管理ポリシーに追加できます。ポリシーによって割り当てられたアクセス許可を確認し、[ポリシーの作成] を選択して作業を保存します。

新しいポリシーが管理ポリシーの一覧に表示され、アタッチの準備ができます。

## ステップ 2: テストユーザーグループにポリシーをアタッチする

次に、MFA で保護されたアクセス許可を付与するために使用されるテスト IAM ユーザーグループに、2 つのポリシーをアタッチします。

1. ナビゲーションペインで、[ユーザーグループ] を選択します。
2. 検索ボックスに「**EC2MFA**」と入力し、リストのグループ名 (チェックボックスではありません) を選択します。
3. [アクセス許可] タブを選択し、[アクセス許可の追加] を選択してから、[ポリシーの添付] を選択します。
4. [Attach permission policies to EC2MFA group] (許可ポリシーを EC2MFA グループにアタッチする) ページの検索ボックスに、**EC2Full** と入力します。次に、リストで、AmazonEC2FullAccess の横にあるチェックボックスを選択します。変更はまだ保存しないでください。

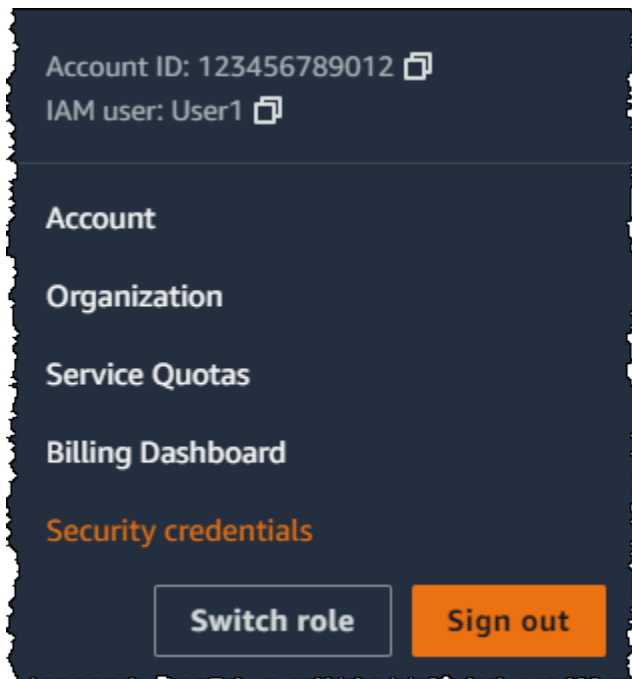


5. 検索ボックスに「**Force**」と入力し、リストの [Force\_MFA] の横にあるチェックボックスをオンにします。
6. [ポリシーのアタッチ] を選択します。

### ステップ 3: ユーザーアクセスをテストする

チュートリアルはこの部分では、テストユーザーとしてサインインし、ポリシーが意図したとおりに動作することを確認します。

1. 前のセクションで割り当てたパスワードを使用し、**MFAUser** として AWS アカウントにサインインします。URL: `https://<alias or account ID number>.signin.aws.amazon.com/console` を使用します。
2. [EC2] を選択して Amazon EC2 コンソールを開き、ユーザーには一切の操作を行うアクセス許可がないことを確認します。
3. 右上のナビゲーションバーで MFAUser ユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



4. ここで MFA デバイスを追加します。[Multi-Factor Authentication(MFA) (多要素認証 (MFA))] セクションで、[Assign MFA device (MFA デバイスを割り当てる)] を選択します。

**Note**

`iam:DeleteVirtualMFADevice` を実行する権限がないというエラーが表示されることがあります。これは、誰かが以前にこのユーザーに仮想 MFA デバイスの割り当てを開始し、プロセスをキャンセルした場合に発生する可能性があります。続行するには、ユーザーまたは他の管理者がユーザーの既存の割り当てられていない仮想化 MFA デバイスを削除する必要があります。詳細については、「[iam:DeleteVirtualMFADevice を実行する権限がありません](#)」を参照してください。

- このチュートリアルでは、携帯電話で Google Authenticator アプリなどの仮想 (ソフトウェアベース) MFA デバイスを使用します。[Authenticator app] (認証アプリケーション) を選択し、[Next] (次へ) をクリックします。

IAM が QR コードを含む仮想 MFA デバイスの設定情報を生成して表示します。図は、QR コードに対応していないデバイスでの手動入力に利用できるシークレット設定キーを示しています。

- 仮想 MFA アプリを開きます。(仮想 MFA デバイスをホストするために使用できるアプリのリストについては、「[仮想 MFA アプリケーション](#)」を参照) 仮想 MFA アプリが複数のアカウント (複数の仮想 MFA デバイス) をサポートしている場合は、新しいアカウント (新しい仮想 MFA デバイス) を作成するオプションを選択します。
- MFA アプリが QR コードをサポートしているかどうかを確認してから、次のいずれかを実行します。
  - ウィザードから、[Show QR code (QR コードの表示)] を選択します。QR コードをスキャンするアプリを使用します。例えば、カメラアイコンまたは [Scan code] (スキャンコード) に似たオプションを選択し、デバイスのカメラを使用してコードをスキャンします。
  - [Set up device] (デバイスの設定) ウィザードで [Show secret key] (シークレットキーを表示) を選択し、MFA アプリケーションにシークレットキーを入力します。

これで仮想 MFA デバイスはワンタイムパスワードの生成を開始します。

- [Set up device] (デバイスの設定) ウィザードの [Enter the code from your authenticator app.] (認証アプリケーションからコードを入力) ボックスに、現在仮想 MFA デバイスに表示されているワンタイムパスワードを入力します。[Register MFA] (MFA の登録) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーと正常に関連付けられません。ただし、MFA デバイスは同期しません。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。その場合は、[デバイスの再同期](#)ができます。

これで仮想 MFA デバイスを AWS で使用できます。

9. コンソールからサインアウトし、再度 **MFAUser** としてサインインします。今回は AWS により、携帯電話から MFA コードを取得するよう求められます。コードを取得したら、それをボックスに入力し、[Submit (送信)] を選択します。
10. [EC2] を選択し、再度 Amazon EC2 コンソールを開きます。今回は、すべての情報を表示して、必要なアクションを実行することができます。このユーザーとして他のコンソールに移動すると、アクセス拒否メッセージが表示されます。その理由は、このチュートリアルポリシーでは Amazon EC2 へのアクセスのみが許可されるためです。

## 関連リソース

詳細については、以下のトピックを参照してください。

- [AWS での多要素認証 \(MFA\) の使用](#)
- [AWS でのユーザーの MFA デバイスの有効化](#)
- [IAM のサインインページでの MFA デバイスの使用](#)

## IAM ID (ユーザー、ユーザーグループ、ロール)

### Tip

AWS へのサインインに問題がある場合、正しいサインインページが表示されていることを確認します。

- AWS アカウントのルートユーザー (アカウント所有者) としてサインインする場合は、AWS アカウント の作成時に設定した認証情報を使用します。
- IAM ユーザーとしてサインインするには、アカウント管理者が AWS へサインインするために提供した認証情報を使用します。
- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM アイデンティティセンターユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「[AWS アクセスポータルにサインインする](#)」を参照してください。

サインインチュートリアルについては、「AWS サインイン ユーザーガイド」の「[AWS へのサインイン方法](#)」を参照してください。

### Note

サポートをリクエストする必要がある場合は、このページの [Feedback] (フィードバック) リンクを使用しないでください。入力したフィードバックは、AWS サポートではなく AWS ドキュメントチームが受け取ります。代わりに、このページの上部にある [Contact Us] (お問い合わせ) リンクを選択します。そこには、必要なサポートを受けるのに役立つリソースへのリンクがあります。

AWS アカウントのルートユーザー またはアカウントの管理ユーザーは、IAM ID を作成できます。IAM ID は、AWS アカウント へのアクセスを提供します。IAM ユーザーグループとは、1つのユニットとして管理される IAM ユーザーの集まりです。IAM ID とは、人間のユーザーまたはプログラムによるワークロードを表し、認証を受けて AWS でアクションを実行する権限を持ちます。各

IAM ID は、1 つ以上のポリシーに関連付けることができます。ポリシーは、ユーザー、ロール、またはユーザーグループのメンバーが実行できるアクション、AWS リソース、および条件を決定します。

## AWS アカウント ルートユーザー

AWS アカウント を最初に作成する場合は、アカウントのすべての AWS のサービス とリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。

### Important

日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

## IAM ユーザー

IAM ユーザー は、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つ AWS アカウント内のアイデンティティです。可能であれば、[ベストプラクティス](#)では、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。アクセスキーを作成する前に、[長期的なアクセスキーの代替案](#)を確認してください。アクセスキーを必要とする特定のユースケースがある場合は、必要に応じてアクセスキーを更新することをお勧めします。詳細については、「[長期的な認証情報を必要とするユースケースのためにアクセスキーを必要な時に更新する](#)」を参照してください。IAM ユーザーを AWS アカウント に追加するには、「[AWS アカウント での IAM ユーザーの作成](#)」を参照してください。

### Note

セキュリティ上の[ベストプラクティス](#)として、IAM ユーザーを作成するのではなく、ID フェデレーションを通じてリソースへのアクセスを提供することをお勧めします。IAM ユーザーが必要な特定の状況についての情報は、「[IAM ユーザー \(ロールの代わりに\) を作成する場合](#)」を参照してください。

## IAM ユーザーグループ

[IAM グループ](#) は、IAM ユーザーの集団を指定するアイデンティティです。サインインにグループを使用することはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMPublishers という名前でグループを作成し、公開するワークロードが一般的に必要なとするようなアクセス許可を、そのグループに付与することができます。

## IAM ロール

[IAM ロール](#) は、特定の許可を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、詳細のユーザーに関連付けられていません。[ロールを切り替える](#) ことによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWSAPI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「[IAM ロールを使用する](#)」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で使用されます。

- フェデレーションユーザーアクセス - フェデレーションアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーションアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「[サードパーティーアイデンティティプロバイダー向けロールの作成](#)」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可セット](#)」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスでのロールとリソースベースのポリシーの違いの詳細については、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。
- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でア

アプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。
- サービスリンクロール - サービスリンクロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWSAPI 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する](#)」を参照してください。

## IAM での一時認証情報

人間のユーザーとワークロードの両方に、一時的な認証情報を使用することが[ベストプラクティス](#)です。一時的な認証情報は主に IAM ロールとともに使用されますが、他の用途もあります。標準的な IAM ユーザーよりも限定的なアクセス許可を含む一時的な認証情報を要求することができます。こ



れにより、より限定的な認証情報によって許可されていないタスクを誤って実行することを防止できます。一時的な認証情報の利点は、一定期間が経過すると自動的に失効することです。認証情報が有効である期間は認証情報を制御できます。

## IAM Identity Center ユーザーを使用する場合とは？

AWS リソースにアクセスする人間のユーザー全員は、IAM Identity Center を使用することをお勧めします。IAM Identity Center は、IAM ユーザーからの AWS リソースへのアクセスを大幅に改善します。IAM Identity Center により以下が提供されます。

- アイデンティティと割り当ての集約
- AWS 組織全体のアカウントへのアクセス
- 既存の ID プロバイダーへの接続
- 一時的な認証情報
- 多要素認証 (MFA)
- エンドユーザーによるセルフサービスの MFA 設定
- 管理目的での MFA の使用の実施
- すべての AWS アカウント 権限へのシングルサインオン

詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center](#)」(IAM Identity Center とは) を参照してください。

## IAM ユーザーの作成が適している場合 (ロールではなく)

IAM ユーザーは、フェデレーションユーザーでサポートされていないユースケースにのみ使用することをお勧めします。ユースケースには次のようなものがあります。

- IAM ロールを使用できないワークロード – AWS へのアクセスが必要な場所から、ワークロードを実行する場合があります。状況によっては、IAM ロールを使用して WordPress プラグインなどに対して一時的な認証情報を提供することができない場合もあります。このような状況では、そのワークロードに対して IAM ユーザーの長期的なアクセスキーを使用して、AWS への認証を行います。
- サードパーティー AWS クライアント – IAM Identity Center を使用したアクセスがサポートされていないツール (AWS でホストされていないサードパーティー AWS クライアントまたはベンダーなど) を使用している場合は、IAM ユーザーの長期的なアクセスキーを使用します。

- AWS CodeCommit アクセス — CodeCommit を使用してコードを保存している場合、CodeCommit の SSH キーまたはサービス固有の認証情報を持つ IAM ユーザーを使用して、リポジトリへの認証を行うことができます。通常の認証に IAM Identity Center のユーザーを使用することに加えて、これを行うことをお勧めします。IAM Identity Center のユーザーとは、お客様の AWS アカウント またはクラウドアプリケーションにアクセスする必要がある従業員のことです。IAM ユーザーを設定せずに CodeCommit リポジトリへのアクセス許可をユーザーに付与するには、`git-remote-codecommit` ユーティリティを設定します。IAM および CodeCommit の詳細については、「[CodeCommit での IAM の使用: Git 認証情報、SSH キー、および AWS アクセスキー](#)」を参照してください。`git-remote-codecommit` ユーティリティの設定についての詳細は、「AWS CodeCommit ユーザーガイド」の「[認証情報のローテーションを使用した AWS CodeCommit リポジトリへの接続](#)」を参照してください。
- Amazon Keyspaces (Apache Cassandra 向け) へのアクセス – IAM Identity Center 内のユーザーを使用できない状況 (Cassandra との互換性をテストする場合など) では、サービス専用の認証情報を持つ IAM ユーザーを使用して Amazon Keyspaces で認証できます。IAM Identity Center のユーザーとは、お客様の AWS アカウント またはクラウドアプリケーションにアクセスする必要がある従業員のことです。一時的な認証情報を使用して Amazon Keyspaces に接続することもできます。詳細については、「Amazon Keyspaces (Apache Cassandra 向け) デベロッパーガイド」の「[Using temporary credentials to connect to Amazon Keyspaces using an IAM role and the SigV4 plugin](#)」(一時的な認証情報を使用して IAM ロールと SigV4 プラグインを使用して Amazon Keyspaces に接続する) を参照してください。
- 緊急アクセス — ID プロバイダーにアクセスできない状況で、AWS アカウント のアクションを実行する必要がある場合。緊急アクセスの IAM ユーザーを設定することは、レジリエンス計画の一部となります。緊急時のユーザー認証情報は、多要素認証 (MFA) を使用して厳重に管理し、安全性を確保することをお勧めします。

## IAM ロールの作成が適している場合 (ユーザーではなく)

次のような状況では、IAM ロールを作成します。

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行するアプリケーションを作成しており、そのアプリケーションが AWS にリクエストを送信します。

IAM ユーザーを作成してユーザーの認証情報をアプリケーションに渡したり、認証情報をアプリケーションに埋め込んだりしないでください。代わりに、EC2 インスタンスにアタッチする IAM ロールを作成し、インスタンスで実行されているアプリケーションに一時的なセキュリティ認証情報を与えます。アプリケーションは AWS でこれらの認証情報を使用すると、ロールにアタッチされたポリシーによって許可されているすべてのオペレーションを実行できます。詳細に

については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください

モバイルフォン上で実行するアプリを作成します。このアプリは AWS にリクエストを送信します。

IAM ユーザーを作成してユーザーのアクセスキーをアプリケーションと共に配信しないでください。代わりに、Login with Amazon、Amazon Cognito、Facebook、または Google などの ID プロバイダーを使用してユーザーを認証し、IAM ロールをユーザーにマッピングします。アプリケーションはロールを使用して、ロールにアタッチされたポリシーで指定されたアクセス権限を持つ一時的なセキュリティ認証情報を取得できます。詳細については、次を参照してください:

- [Amazon Cognito ユーザーガイド](#)
- [OIDC フェデレーション](#)

企業内のユーザーが企業ネットワークで認証された後、再びサインインすることなく AWS を使用できるようにします。つまり、ユーザーに AWS へのフェデレーションを許可します。

IAM ユーザーを作成しないでください。企業の ID システムと AWS 間にフェデレーション関係を設定します。これには 2 つの方法があります。

- 企業の ID システムが SAML 2.0 と互換性がある場合は、企業の認証システムと AWS 間に信頼を確立できます。詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。
- 企業のユーザー ID を一時的な AWS セキュリティ認証情報を提供する IAM ロールに変換するカスタムプロキシサーバーを作成して使用します。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

## AWS アカウントのルートユーザー の認証情報と IAM ユーザーの認証情報を比較する

ルートユーザーはアカウント所有者であり、AWS アカウント の作成時に作成されます。IAM ユーザーや AWS IAM Identity Center ユーザーなどを含む他のタイプのユーザーは、ルートユーザーまたはアカウントの管理者によって作成されます。すべての AWS ユーザーはセキュリティ認証情報を持っています。

### ルートユーザーの認証情報

アカウント所有者の認証情報によって、アカウント内のすべてのリソースへのフルアクセスが許可されます。[IAM ポリシー](#)を使用して、リソースへのルートユーザーアクセスを明示的に拒否することはできません。AWS Organizations [サービスコントロールポリシー](#) (SCP) を使用できるのは、メンバーアカウントのルートユーザーの許可を制限する場合のみです。このため、日常的な AWS タス

クに使用するための管理者許可を持つユーザーを IAM Identity Center で作成することをお勧めします。その後、ルートユーザーの認証情報を保護し、ルートユーザーとしてサインインする必要がある少数のアカウント/サービス管理タスクのみを実行するためにそれを使用します。これらのタスクのリストについては、「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。IAM Identity Center で日常的に使用する管理者を設定する方法については、IAM Identity Center ユーザーガイドの「[はじめに](#)」を参照してください。

## IAM 認証情報

IAM ユーザーは AWS で作成されるエンティティであり、IAM ユーザーを使用して AWS リソースとやり取りする人またはサービスを表します。これらのユーザーは、特定のカスタム権限を持つ AWS アカウント内のアイデンティティです。たとえば、IAM ユーザーを作成し、IAM Identity Center にディレクトリを作成するアクセス許可を付与できます。IAM ユーザーは、AWS Management Console を使用して、またはプログラムで AWS CLI または AWS API を使用して AWS にアクセスするために使用できる長期的な認証情報を持っています。IAM ユーザーが AWS Management Console にサインインする手順については、AWS サインインユーザーガイドの「[IAM ユーザーとして AWS Management Console にサインインするには](#)」を参照してください。

一般的に、IAM ユーザーにはユーザー名やパスワードなどの長期的な認証情報があるため、作成しないことをお勧めします。代わりに、人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用を要求します。一時的な資格情報を提供する IAM ロールを引き受けることで、人間のユーザーのアイデンティティプロバイダーを使用した AWS アカウントへのフェデレーションアクセスが可能になります。一元的なアクセス管理を行うには、[IAM Identity Center](#) を使用して、ご自分のアカウントへのアクセスと、それらのアカウント内でのアクセス許可を管理することをお勧めします。ユーザー ID を、IAM Identity Center を使用して管理することも、外部 ID プロバイダーから、IAM Identity Center 内のユーザー ID に付与するアクセス許可を管理することもできます。詳細については、IAM Identity Center ユーザーガイドの「[IAM Identity Center とは](#)」を参照してください。

## AWS アカウントのルートユーザー

Amazon Web Services (AWS) アカウントを初めてを作成する場合は、このアカウントのすべての AWS サービスとリソースに対して完全なアクセス権限を持つシングルサインインアイデンティティで始めます。このアイデンティティは AWS アカウントルートユーザーと呼ばれ、アカウントの作成に使用したメールアドレスとパスワードでのサインインによりアクセスされます。

### Important

日常的なタスクにはルートユーザーを使用せず、[AWS アカウントのルートユーザーのベストプラクティス](#)に従うことを強くお勧めします。ルートユーザーの認証情報は保護し、ルー

トユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

以下のトピックでは、ルートユーザーに関連する管理タスクについて詳しく説明します。

## タスク

- [AWS アカウントのルートユーザー の多要素認証を有効にする \(コンソール\)](#)
- [AWS アカウントのルートユーザー のパスワードを変更する](#)
- [紛失または忘れたルートユーザーのパスワードのリセット](#)
- [ルートユーザーのアクセスキーの作成](#)
- [ルートユーザーのアクセスキーの削除](#)
- [ルートユーザー認証情報が必要なタスク](#)
- [ルートユーザーに関する問題のトラブルシューティング](#)
- [関連情報](#)

## AWS アカウントのルートユーザー の多要素認証を有効にする (コンソール)

多要素認証 (MFA) は、セキュリティを強化するためのシンプルで効果的なメカニズムです。1 つ目の要因であるパスワードは、ユーザーが記憶する秘密であり、知識要因とも呼ばれます。その他の要因としては、所有要因 (セキュリティキーなど、ユーザーが持っているもの) や継承要因 (生体認証スキャンなど、ユーザー自身のもの) があります。セキュリティを向上させるには、多要素認証 (MFA) を設定して AWS リソースを保護することを強くお勧めします。

AWS アカウントのルートユーザー および IAM ユーザーでは、MFA を使用することができます。ルートユーザーの MFA を有効化すると、ルートユーザーの認証情報のみが影響を受けます。IAM ユーザーの MFA を有効にする方法の詳細については、「[AWS でのユーザーの MFA デバイスの有効化](#)」を参照してください。

ルートユーザーの MFA を有効にする前に、[アカウント設定と連絡先情報を確認および更新](#)して、Eメールと電話番号にアクセスできることを確認してください。MFA デバイスが紛失したり、盗難にあったり、機能しなくなったりしても、そのメールアドレスと電話番号を使用してアイデンティティを確認することで、ルートユーザーとしてサインインできます。代替の認証要素を使用してログイン



する方法については、「[MFA デバイスの紛失および故障時の対応](#)」を参照してください。この機能を無効にするには、「[AWS Support](#)」にお問い合わせください。

## トピック

- [ルートユーザーが使用できる MFA タイプ](#)
- [AWS アカウントのルートユーザー のパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)
- [AWS アカウントのルートユーザー \(コンソール\) の仮想 MFA デバイスを有効にします](#)
- [AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする \(コンソール\)](#)

## ルートユーザーが使用できる MFA タイプ

ルートユーザーに対し AWS でサポートされている MFA タイプは、パスキーとセキュリティキー、仮想認証アプリケーション、およびハードウェア TOTP トークンです。

### パスキーとセキュリティキー

AWS Identity and Access Management は MFA のパスキーとセキュリティキーをサポートします。FIDO 標準に基づいて、パスキーではパブリックキー暗号化が使用され、パスワードよりも安全性の高い、強力でフィッシング耐性のある認証が提供されます。AWS は、デバイスバインドパスキー (セキュリティキー) と同期パスキーの 2 種類のパスキーをサポートしています。

- セキュリティキー: これらは YubiKey などの物理デバイスで、認証の 2 番目の要素として使用されます。
- 同期パスキー: これらは、Google、Apple、Microsoft アカウントなどのプロバイダーの認証情報マネージャー、および 1Password、Dashlane、Bitwarden などのサードパーティーサービスを 2 番目の要素として使用します。

Apple MacBooks の Touch ID や PC の Windows Hello 顔認識などの組み込みの生体認証機能を使用して、認証情報マネージャーのロックを解除し、AWS にサインインできます。パスキーは、指紋、顔、またはデバイス PIN を使用して、選択したプロバイダーで作成されます。デバイス間でパスキーを同期することで、AWS へのサインインが容易になり、使いやすさと回復性が向上します。

FIDO 仕様と互換性のあるすべての [FIDO 認定製品](#) のリストが、FIDO アライアンスから提供されています。1 つのパスキーまたはセキュリティキーで、複数のルートユーザーアカウントと IAM ユーザーをサポートします。パスキーとセキュリティキーの有効化の詳細については、「[AWS アカ](#)

[トのルートユーザー のパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)」を参照してください。

## 仮想認証アプリケーション

仮想認証アプリケーションは、電話やその他のデバイスで実行され、物理デバイスをエミュレートします。仮想認証アプリは、[タイムベースドワンタイムパスワード \(TOTP\) アルゴリズム](#)を実装しており、単一デバイスで複数のトークンをサポートします。ユーザーは、サインイン中にプロンプトが表示されたら、デバイスから有効なコードを入力する必要があります。ユーザーに割り当てられた各トークンは一意であることが必要です。ユーザーは、別のユーザーのトークンからコードを入力して認証を受けることはできません。

ハードウェアの購入承認の待機中、またはハードウェアの到着を待つ間に、仮想 MFA デバイスを使用することをお勧めします。仮想 MFA デバイスとして使用可能なサポートされているアプリケーションのリストについては、「[多要素認証 \(MFA\)](#)」を参照してください。AWS で仮想 MFA デバイスを設定する手順については、「[AWS アカウントのルートユーザー \(コンソール\) の仮想 MFA デバイスを有効にします](#)」を参照してください。

## ハードウェア TOTP トークン

ハードウェアデバイスでは、[タイムベースドワンタイムパスワード \(TOTP\) アルゴリズム](#)に基づいて 6 桁の数値コードが生成されます。サインイン時に、ユーザーはデバイスから取得した有効なコードを 2 番目のウェブページに入力する必要があります。ユーザーに割り当てられた各 MFA デバイスは一意であることが必要です。ユーザーは、別のユーザーのデバイスからコードを入力して認証することはできません。サポートされているハードウェア MFA デバイスの詳細については、「[多要素認証 \(MFA\)](#)」を参照してください。AWS でハードウェア TOTP デバイスを設定する手順については、「[AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする \(コンソール\)](#)」を参照してください。

物理的な MFA デバイスを使用する場合は、ハードウェア TOTP デバイスの代わりに FIDO セキュリティキーを使用することをお勧めします。FIDO セキュリティキーのメリットは、バッテリーが不要でフィッシング耐性があるという点です。さらには、セキュリティを強化するために、1 台のデバイスで複数のルートユーザーと IAM ユーザーをサポートしています。

## AWS アカウントのルートユーザー のパスキーまたはセキュリティキーを有効にする (コンソール)

ルートユーザーのパスキーは、AWS CLI API や AWS API からではなく、AWS Management Console からのみ設定して有効にできます。



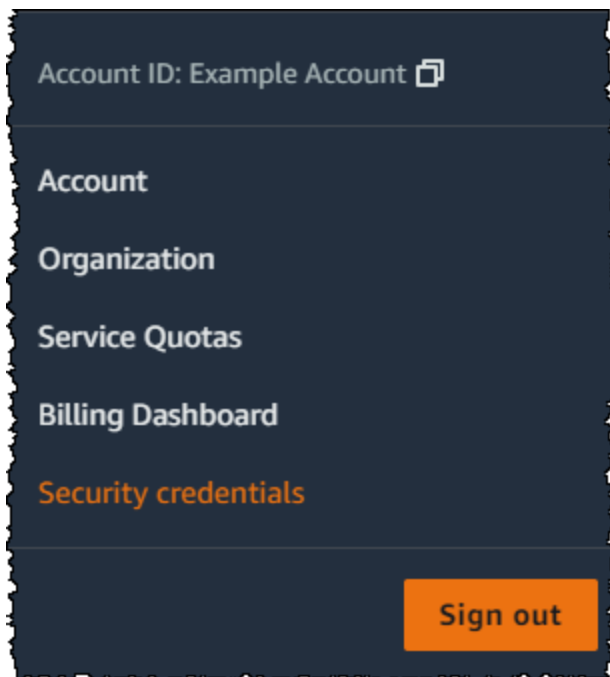
## ルートユーザーのパスキーまたはセキュリティキーを有効にするには (コンソール)

1. Root user (ルートユーザー) を選択して AWS アカウント の E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

### Note

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[IAM ユーザーのサインイン] ページが表示された場合、ページ下部付近で [ルートユーザーの電子メールを使用してサインインする] を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

2. ナビゲーションバーの右側で、使用するアカウント名を選択してから、[Security credentials] (セキュリティ認証情報) を選択します。必要に応じて、[Continue to Security credentials] (セキュリティ認証情報に進む) を選択します。



3. ルートユーザーの [セキュリティ認証情報] ページの [多要素認証 (MFA)] で、[MFA デバイスの割り当て] を選択します。
4. [MFA デバイス名] ページで、[デバイス名] を入力し、[パスキーまたはセキュリティキー] を選択し、[次へ] を選択します。

5. [デバイスの設定] で、パスキーを設定します。顔や指紋などの生体認証データを使用、デバイス PIN を使用、またはコンピュータの USB ポートに FIDO セキュリティキーを挿入してタップすることで、パスキーを作成します。
6. ブラウザの指示に従って、パスキープロバイダーを選択するか、デバイス全体で使用するパスキーを保存する場所を選択します。
7. Continue (続行) をクリックします。

これで、AWS で使用するパスキーが登録されました。次回にルートユーザーの認証情報を使用してサインインするときは、パスキーで認証してサインインプロセスを完了する必要があります。

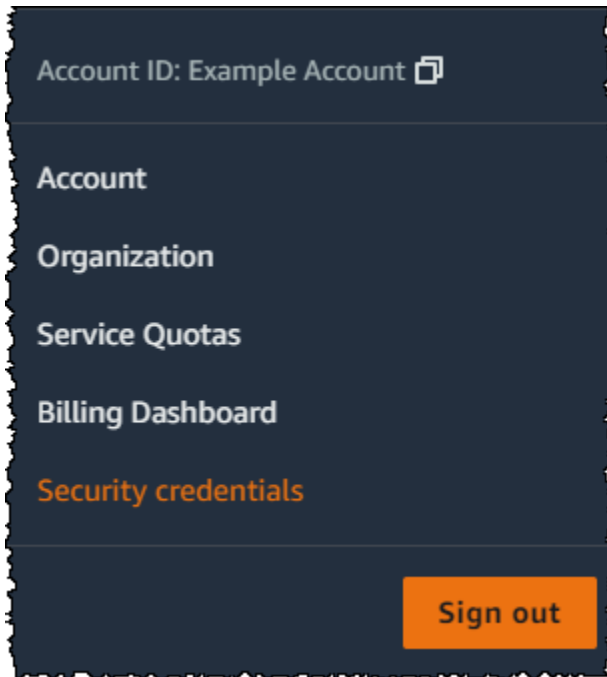
FIDO セキュリティキーに関する問題のトラブルシューティングについては、「[FIDO セキュリティキーのトラブルシューティング](#)」を参照してください。

## AWS アカウントのルートユーザー (コンソール) の仮想 MFA デバイスを有効にします

AWS Management Console を使用して、仮想 MFA デバイスをルートユーザーが使用できるように設定して有効化することができます。AWS アカウントの MFA デバイスを有効にするには、ルートユーザー認証情報を使用して AWS にサインインしている必要があります。

仮想 MFA デバイスを設定および有効化してルートユーザーで使用するには (コンソール)

1. AWS Management Console にサインインします。
2. ナビゲーションバーの右側で、使用するアカウント名を選択してから、[Security credentials] (セキュリティ認証情報) を選択します。必要に応じて、[Continue to Security credentials] (セキュリティ認証情報に進む) を選択します。



3. [Multi-Factor Authentication(MFA)] (多要素認証 (MFA)) セクションで、[Assign MFA device] (MFA デバイスの割り当て) を選択します。
4. ウィザードで [デバイス名] を入力し、[認証アプリ]、[次へ] の順に選択します。

IAM が QR コードを含む仮想 MFA デバイスの設定情報を生成して表示します。図は、QR コードに対応していないデバイスでの手動入力に利用できるシークレット設定キーを示しています。

5. デバイスで仮想 MFA アプリを開きます。

仮想 MFA アプリが複数の仮想 MFA デバイスまたはアカウントをサポートしている場合は、新しい仮想 MFA デバイスまたはアカウントを作成するオプションを選択します。

6. QR コードをスキャンするアプリを使用してアプリを設定するのが最も簡単な方法です。QR コードに対応していない場合には、設定情報を手入力します。IAM によって生成された QR コードやシークレット設定キーはお客様の AWS アカウント に紐付けられており、別のアカウントでは使用できません。ただし、元の MFA デバイスが利用できなくなった場合に、これらの情報を再利用してアカウントの新しい MFA デバイスを設定できます。

- QR コードを使用して仮想 MFA デバイスを設定するには、ウィザードで [Show QR code (QR コードの表示)] を選択します。その後、アプリの指示に従ってコードをスキャンします。例えば、カメラアイコンや [Scan account barcode] のようなコマンドを選択し、デバイスのカメラを使用してコードを QR スキャンする場合があります。
- [Set up device] (デバイスの設定) ウィザードで、[Show secret key] (シークレットキーの表示) を選択し、その後、MFA アプリにシークレットキーを入力します。

**⚠ Important**

QR コードまたはシークレット設定キーの安全なバックアップを作成するか、アカウント用に複数の MFA デバイスを有効にしていることを確認します。AWS アカウントのルートユーザー および IAM ユーザーに対し、「[現在サポートされている MFA タイプ](#)」の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。例えば、仮想 MFA デバイスがホストされているスマートフォンを紛失した場合などは、仮想 MFA デバイスが使用できなくなる可能性があります。そのような場合で、ユーザーにアタッチされた他の MFA デバイスや、[ルートユーザー MFA デバイスの回復](#) を使用してもアカウントにサインインできない場合は、アカウントへのサインインが不能になるので、[カスタマー サービスに連絡](#)して MFA によるアカウントの保護を解除する必要があります。

デバイスは 6 桁の数字の生成を開始します。

7. ウィザードの [MFA Code 1] (MFA コード 1) ボックスに、仮想 MFA デバイスに現在表示されているワンタイムパスワードを入力します。デバイスが新しいワンタイムパスワードを生成するまで待ちます (最長 30 秒)。生成されたら [MFA code 2 (MFA コード 2)] ボックスに 2 つ目のワンタイムパスワードを入力します。[Add MFA] (MFA を追加) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けられませんが、その MFA デバイスは同期されません。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。その場合は、[デバイスの再同期](#)ができます。

デバイスを AWS で使用する準備が整いました。AWS Management Consoleでの MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

## AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする (コンソール)

ルートユーザーの物理的な MFA デバイスは、AWS CLI または AWS API からではなく、AWS Management Console からのみ設定して有効にすることができます。

**Note**

他のテキスト (例: MFA を使用してサインイン や 認証デバイスのトラブルシューティング) が表示される場合があります。ただし、提供されている機能は同じです。いずれの場合も、認証の代替要因を使用してアカウントの E メールアドレスと電話番号を確認できない場合は、MFA 設定の非アクティブ化について [AWS Support](#) にお問い合わせください。

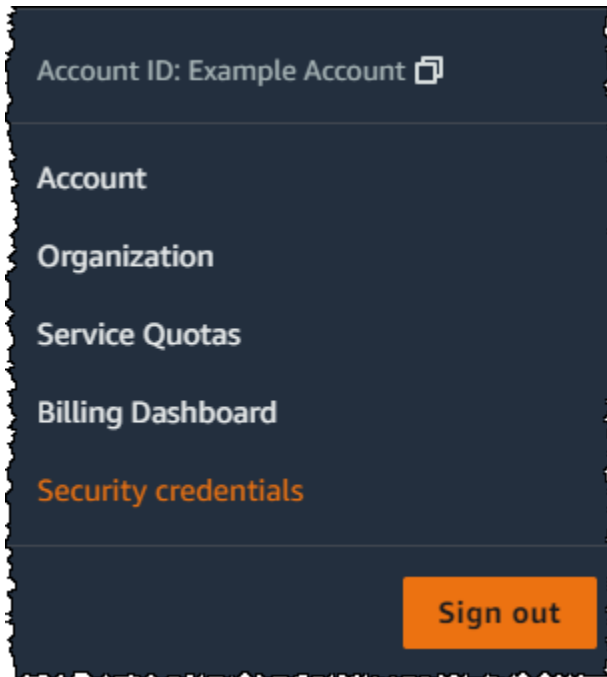
## ルートユーザーの MFA デバイスを有効化するには (コンソール)

1. Root user (ルートユーザー) を選択して AWS アカウント の E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

**Note**

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[IAM ユーザーのサインイン] ページが表示された場合、ページ下部付近で [ルートユーザーの電子メールを使用してサインインする] を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

2. ナビゲーションバーの右側で、使用するアカウント名を選択し、次に [Security Credentials] (セキュリティ認証情報) を選択します。必要に応じて、[Continue to Security credentials] (セキュリティ認証情報に進む) を選択します。



3. [Multi-factor authentication (MFA)] セクションを展開します。
4. [Assign MFA device] (MFA デバイスの割り当て) を選択します。
5. ウィザード内で [Device name] (デバイス名) に入力した後、[Hardware TOTP token] (ハードウェア TOTP トークン)、[Next] (次へ) の順に選択します。
6. [シリアル番号] ボックスに MFA デバイス背面に表示されているシリアル番号を入力します。
7. MFA デバイスに表示されている 6 桁の数字を [MFA code 1 (MFA コード 1)] に入力します。デバイス前面のボタンを押して数字を表示する場合があります。



8. デバイスがコードを更新するまで 30 秒ほど待ち、更新後に表示された 6 桁の数字を [MFA code 2 (MFA コード 2)] に入力します。デバイス前面のボタンを再度押して新しい数字を表示する場合があります。
9. [Add MFA] (MFA を追加) を選択します。これで、MFA デバイスと AWS アカウント の関連付けが完了しました。

#### **⚠ Important**

認証コードを生成した後すぐに、リクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常

に関連付けられますが、その MFA デバイスは同期されなくなります。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。その場合は、[デバイスの再同期](#)ができます。

次回、ルートユーザー認証情報を使ってサインインするときに、MFA デバイスのコードを入力する必要があります。

## AWS アカウントのルートユーザー のパスワードを変更する

[\[セキュリティ 認証情報\]](#) または [\[アカウント\]](#) ページにアクセスすると、メールアドレスとパスワードを変更できます。AWS サインインページで [\[Forgot password\]?](#) (パスワードを忘れた場合) を選択して、パスワードをリセットすることもできます。

ルートユーザーのパスワードを変更するには、IAM ユーザーではなく、AWS アカウントのルートユーザーとしてサインインする必要があります。ルートユーザーパスワードを忘れた場合にそれをリセットする方法については、「[紛失または忘れたルートユーザーのパスワードのリセット](#)」を参照してください。

パスワードを保護するには、次のベストプラクティスに従う必要があります。

- パスワードを定期的に変更します。
- パスワードを知っているユーザーがアカウントにアクセスできるので、パスワードはプライベートに保ちます。
- AWS では、他のサイトで使用しているものとは異なるパスワードを使用します。
- 安易に想像できるパスワードは使用しないでください。例えば、secret、password、amazon、123456 などのパスワードです。また、辞書に載っている単語や氏名、メールアドレスまたはその他の簡単に入手できる個人情報なども含みます。

### AWS Management Console

ルートユーザーのパスワードを変更するには


#### 最小アクセス許可

以下の手順を実行するには、少なくとも次の IAM アクセス許可が必要です。




- 追加の AWS アカウント (IAM) アクセス許可を必要としない AWS Identity and Access Management ルートユーザーとしてサインインする必要があります。IAM ユーザーまたはロールとしてこれらの手順を実行することはできません。

1. AWS アカウントのメールアドレスとパスワードを使用し、AWS アカウントのルートユーザーとして「[AWS Management Console](#)」にサインインします。
2. コンソールの右上隅のアカウント名またはアカウント番号を選択してから [Account] (アカウント) を選択します。
3. [アカウント] ページで、[アカウント設定] の横の [編集] を選択します。セキュリティ上の理由から、再認証を求められます。

 Note

[編集] オプションが表示されない場合は、アカウントのルートユーザーとしてログインしていない可能性があります。IAM ユーザーまたはロールとしてサインインしている間は、アカウント設定を変更できません。

4. [アカウント設定の更新] ページで、[パスワード]、[編集] の順に選択します。
5. [パスワードの更新] ページで、[現在のパスワード]、[新しいパスワード]、[新しいパスワードの確認] の各フィールドに入力します。

 Important

必ず強力なパスワードを選択してください。IAM ユーザー用のアカウントパスワードポリシーを設定することはできますが、このポリシーはルートユーザーには適用されません。

AWS ではパスワードが以下の条件を満たす必要があります：

- 8～128 文字で構成されていること。
- 英字の大文字と小文字、数字、および ! @ # \$ % ^ & \* ( ) < > [ ] { } | \_ + = の記号を含める必要があります。
- AWS アカウント アカウント名またはメールアドレスと同じでないこと。

**Note**

AWS ではサインインプロセスの改善を進めています。これらの改善の 1 つは、より安全なパスワードポリシーをアカウントに強制することです。AWS アカウントをアップグレードした場合は、上記のパスワードポリシーを満たすことを要求されます。AWS が依然アカウントをアップグレードしていない場合、AWS はこのポリシーをまだ施行していません。ただし、より安全なパスワードのガイドラインに従うことを強くお勧めします。

6. [Save changes] (変更の保存) をクリックします。

## AWS CLI or AWS SDK

このタスクは AWS CLI 内でも AWS SDK のいずれかによる API 操作でもサポートされません。このタスクは、AWS Management Console を使用してのみ実行できます。

## 紛失または忘れたルートユーザーのパスワードのリセット

AWS アカウント を初めて作成したときに、E メールアドレスとパスワードを指定しました。これらは、AWS アカウントのルートユーザー 認証情報です。パスワード ルートユーザーパスワードを忘れた場合は、AWS Management Console からパスワードをリセットできます。

ルートユーザーパスワードをリセットするには:

1. AWS アカウント のメールアドレスを使用してルートユーザーとして [AWS Management Console](#) へのサインインを開始し、[Next] (次へ) を選択します。

**Note**

IAM ユーザー認証情報を使用して [AWS Management Console](#) にサインインしている場合、ルートユーザーパスワードをリセットするには、まずサインアウトする必要があります。アカウント固有の IAM ユーザーのサインインページが表示された場合は、ページの下部付近にある ルートアカウントの認証情報を使用してサインインする を選択します。必要に応じて、アカウントの E メールアドレスを指定し、[次へ] を選択して [ルートuser sign in (ルートユーザーサインイン)] ページにアクセスします。

## 2. [パスワードを忘れた場合] を選択します。

### Note

IAM ユーザーの場合、このオプションは使用できません。[パスワードを忘れた場合] オプションは、ルートユーザーアカウントでのみ使用できます。IAM ユーザーの場合、忘れたパスワードをリセットするには管理者に依頼必要があります。詳細については、「[AWS アカウントのIAMユーザーパスワードを忘れてしまいました](#)」を参照してください。AWS アクセスポータルからサインインする場合は、「[IAM アイデンティティセンターのユーザーパスワードのリセット](#)」を参照してください。

3. アカウントに関連付けられている E メールアドレスを入力します。次に、CAPTCHA のテキストを入力し、[続行] を選択します。
4. AWS アカウントに関連付けられている E メールを Amazon Web Services からのメッセージで確認します。メールは @verify.signin.aws で終わるアドレスから届きます。Eメールの指示にしたがって操作します。アカウントに E メールが表示されない場合は、スパムフォルダを確認してください。メールにアクセスできなくなった場合は、AWS サインイン ユーザーガイドの「[自分の AWS アカウントのメールにアクセスできない](#)」を参照してください。

## ルートユーザーのアクセスキーの作成

### Warning

ルートユーザーのアクセスキーペアを作成しないことを強くお勧めします。[ルートユーザーが必要とするタスクはごくわずか](#)であり、これらのタスクは通常、頻繁に実行されないため、AWS Management Console にサインインしてルートユーザーのタスクを実行することをお勧めします。アクセスキーを作成する前に、[長期的なアクセスキーの代替案](#)を確認してください。

お勧めしませんが、AWS Command Line Interface (AWS CLI) または、ルートユーザーの認証情報を使用した AWS SDK のいずれかの API 操作を使用してコマンドを実行できるように、ルートユーザーのアクセスキーを作成できます。アクセスキーを作成するときは、アクセスキー ID とシークレットアクセスキーをセットとして作成します。アクセスキーの作成時に、AWS では、アクセスキーのシークレットアクセスキーの部分を表示およびダウンロードする機会が 1 回限り与えられます。これをダウンロードしない場合や紛失した場合は、アクセスキーを削除して新しいアクセスキー

を作成できます。ルートユーザーアクセスキーは、コンソール、AWS CLI、または AWS API で作成できます。

新しく作成したアクセスキーのステータスは active になります。これは、CLI および API コール用にこのアクセスキーを使用できる状態です。ルートユーザー に最大 2 つのアクセスキーを割り当てることができます。

使用していないアクセスキーは非アクティブ化する必要があります。アクセスキーが非アクティブになると、そのアクセスキーを API 呼び出しに使用することはできなくなります。非アクティブになったキーも、引き続き制限に対してカウントされます。アクセスキーはいつでも作成したり削除したりすることができます。ただし、一度アクセスキーを削除した場合、永久に削除されて、回復することはできません。

## AWS Management Console

AWS アカウントのルートユーザー のアクセスキーを作成するには

### 最小アクセス許可

以下の手順を実行するには、少なくとも次の IAM アクセス許可が必要です。

- 追加の AWS アカウント (IAM) アクセス許可を必要としない AWS Identity and Access Management ルートユーザーとしてサインインする必要があります。IAM ユーザーまたはロールとしてこれらの手順を実行することはできません。

1. AWS アカウント のメールアドレスとパスワードを使用し、AWS アカウントのルートユーザー として [AWS Management Console の始め方](#) にサインインしてください。
2. コンソールの右上隅で、アカウント名またはアカウント番号を選択し、続いて [セキュリティ認証情報] を選択します。
3. [Access keys] (アクセスキー) セクションで、[Create access key] (アクセスキーを作成) を選択します。このオプションを使用できない場合は、既に最大数のアクセスキーがあります。新しいキーを作成する前に、既存のアクセスキーのいずれかを削除する必要があります。詳細については、「[IAM オブジェクトクォータ](#)」を参照してください。
4. [ルートユーザーアクセスキーの代替方法] ページで、セキュリティに関する推奨事項を確認してください。続行するには、チェックボックスを選択し、[アクセスキーの作成] を選択します。
5. [アクセスキーの取得] ページに、[アクセスキー] の ID が表示されます。

6. [シークレットアクセスキー] で [表示] を選択し、ブラウザのウィンドウからアクセス キー ID とシークレットキーをコピーし、どこか別の安全な場所に貼り付けます。また、[.csv ファイルをダウンロード] を選択して、アクセスキー ID とシークレットキーが含まれる rootkey.csv という名前のファイルをダウンロードすることもできます。そのファイルをどこか安全な場所に保管します。
7. [Done] (完了) をクリックします。アクセスキーが不要になったら、悪用されることを防ぐために、[そのキーを削除するか](#)、少なくとも無効化することを検討してください。

## AWS CLI & SDKs

ルートユーザーのアクセスキーを作成するには

### Note

ルートユーザーとして次のコマンドまたは API 操作を実行するには、アクティブなアクセスキーペアが 1 つ必要です。アクセスキーがない場合は、AWS Management Console を使用して最初のアクセスキーを作成します。次いで、AWS CLI でその 1 つ目のアクセスキーから得た認証情報を使用して、2 つ目のアクセスキーを作成するか、アクセスキーを削除します。

- AWS CLI: [aws iam create-access-key](#)

### Example

```
$ aws iam create-access-key
{
  "AccessKey": {
    "UserName": "MyUserName",
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "Status": "Active",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "CreateDate": "2021-04-08T19:30:16+00:00"
  }
}
```

- AWS API: IAM API リファレンスの [CreateAccessKey](#)。

## ルートユーザーのアクセスキーの削除

AWS Management Console、AWS CLI または AWS API を使用して、ルートユーザーアクセスキーを削除することはできません。

### AWS Management Console

ルートユーザーのアクセスキーを削除するには

#### 最小アクセス許可

以下の手順を実行するには、少なくとも次の IAM アクセス許可が必要です。

- 追加の AWS アカウント (IAM) アクセス許可を必要としない AWS Identity and Access Management ルートユーザーとしてサインインする必要があります。IAM ユーザーまたはロールとしてこれらの手順を実行することはできません。

1. AWS アカウント のメールアドレスとパスワードを使用し、AWS アカウントのルートユーザーとして [AWS Management Console の始め方](#) にサインインしてください。
2. コンソールの右上隅で、アカウント名またはアカウント番号を選択し、続いて [セキュリティ認証情報] を選択します。
3. [アクセスキー] セクションで、削除するアクセスキーを選択して、[アクション]、[削除] の順に選択します。

#### Note

または、アクセスキーを完全に削除する代わりに [非アクティブ化] することもできます。これにより、キー ID またはシークレットキーを変更することなく、後でそのキーの使用を再開できます。非アクティブになっているアクセスキーを AWS API へのリクエストで使用しようとしても、その試みはすべて失敗して、アクセス拒否の状態になります。

4. [<access key ID> を削除] ダイアログボックスで、[非アクティブ化] を選択し、アクセスキー ID を入力して削除を確認したら、[削除] を選択します。

## AWS CLI & SDKs

ルートユーザーのアクセスキーを削除するには

### 最小アクセス許可

以下の手順を実行するには、少なくとも次の IAM アクセス許可が必要です。

- 追加の AWS アカウント (IAM) アクセス許可を必要としない AWS Identity and Access Management ルートユーザーとしてサインインする必要があります。IAM ユーザーまたはロールとしてこれらのステップを実行することはできません。

- AWS CLI: [aws iam delete-access-key](#)

### Example

```
$ aws iam delete-access-key \  
  --access-key-id AKIAIOSFODNN7EXAMPLE
```

このコマンドが成功した場合、出力は生成されません。

- AWS API: [DeleteAccessKey](#)

## ルートユーザー認証情報が必要なタスク

### Important

AWS へのサインインに問題がある場合 ユーザーのタイプに応じて正しい [AWS サインインページ](#)が表示されていることを確認します。AWS アカウントのルートユーザー (アカウント所有者) の場合は、AWS アカウント の作成時に設定した認証情報を使用して AWS にサインインできます。IAM ユーザーの場合、アカウント管理者から AWS へのサインインに使用可能な認証情報をもらうことができます。サポートをリクエストする必要がある場合、このページのフィードバックリンクは使用しないでください。このフォームは AWS サポートではなく、AWS Support ドキュメントチームに送信されます。代わりに、「[お問い合わせ](#)」ページで [AWS アカウントにまだログインできない] を選択してから、表示されているサポートオプションのいずれかを選択します。



日常的なタスクを実行したり、AWS リソースにアクセスしたりするには、[AWS IAM Identity Center で管理者ユーザーを設定](#)することをお勧めします。ただし、アカウントのルートユーザーとしてサインインする場合にのみ、以下に示すタスクを実行できます。

## アカウント管理タスク

- [アカウントの設定を変更します。](#)これには、アカウント名、E メールアドレス、ルートユーザーパスワード、およびルートユーザーアクセスキーが含まれます。連絡先情報、支払い通貨設定、AWS リージョンなどの他のアカウント設定では、ルートユーザー認証情報は不要です。
- [IAM ユーザーアクセス許可を更新します。](#)唯一の IAM 管理者が自身のアクセス許可を誤って取り消した場合は、ルートユーザーとしてサインインしてポリシーを編集し、アクセス許可を復元できます。
- [AWS アカウント を閉じます。](#)

詳細については、次のトピックを参照してください。

- [自分の AWS アカウントを別のエンティティに所有権を割り当てる方法について教えてください。](#)
- [AWS アカウントを閉じる方法について教えてください。](#)
- [スタンドアロン AWS アカウントを閉じる。](#)

## 請求タスク

- [請求情報とコスト管理コンソールへの IAM アクセスをアクティブにする](#)
- 一部の請求タスクはルートユーザーに限定されます。詳細については、AWS Billingユーザーガイドの「[AWS アカウントの管理](#)」を参照してください。
- 特定の税金請求書を表示します。[aws-portal:ViewBilling](#) 許可を持つ IAM ユーザーは、AWS Europe から VAT 請求書を表示およびダウンロードすることができますが、AWS Inc または Amazon Internet Services Private Limited (AISPL) からはできません。

## AWS GovCloud (US) タスク

- [AWS GovCloud \(US\) にサインアップします。](#)
- AWS GovCloud (US) アカウントに AWS Support からのルートユーザーアクセスキーを要求します。

## Amazon EC2 タスク

- リザーブドインスタンスマーケットプレイスに[出品者として登録します](#)。

## AWS KMS タスク

- AWS Key Management Service キーが管理不能になった場合、管理者は AWS Support に連絡して再び管理可能にできます。ただし、AWS Support はルートユーザーのプライマリ電話番号に応答して、チケット OTP の確認による認証をします。

## Amazon Mechanical Turk タスク

- [AWS アカウントを MTurk リクエストアカウントにリンクします](#)。

## Amazon Simple Storage Service Tasks

- [MFA \(多要素認証\)に対応するように Amazon S3 バケットを設定します](#)。
- [すべてのプリンシパルを拒否する Amazon S3 バケットを編集または削除します](#)。

## Amazon Simple Queue Service Task

- [すべてのプリンシパルを拒否する Amazon SQS リソースポリシーを編集または削除します](#)。

## ルートユーザーに関する問題のトラブルシューティング

ここに記載する情報は、AWS アカウントのルートユーザーに関する問題のトラブルシューティングに役立ちます。

アカウントルートユーザーとしてサインインしたときに実行できると期待されるタスクを実行できない

アカウントのルートユーザーとしてサインインしたときにタスクを完了できない場合、そのアカウントが AWS Organizations 内の組織のメンバーである可能性があります。組織の管理者がサービスコントロールポリシー (SCP) でアカウントのアクセス許可を制限した場合、ルートユーザーを含めたすべてのユーザーが影響を受けます。詳細については、AWS Organizations ユーザーガイドの「[サービスコントロールポリシー](#)」を参照してください。

## AWS アカウント のルートユーザーパスワードを忘れてしまった

ルートユーザーであり、AWS アカウント アカウントのパスワードを紛失または忘れた場合は、パスワードをリセットできます。AWS アカウント の作成に使用した E メールアドレスを把握し、E メールアカウントへのアクセス権限を持っている必要があります。詳細については、「[紛失または忘れたルートユーザーのパスワードのリセット](#)」を参照してください。

## AWS アカウント アカウントの E メールにアクセスできない

AWS アカウント を作成する際に E メールアドレスとパスワードを指定します。これらは、AWS アカウントのルートユーザー の認証情報です。AWS アカウント に関連付けられているメールアドレスが不明な場合、@signin.aws または @verify.signin.aws から、AWS アカウント を開くために使用された可能性のある、組織のメールアドレスとの間で送受信されたメッセージを検索します。

E メールアドレスがわかっても、E メールにアクセスできなくなった場合は、まず次のいずれかのオプションを使用して、E メールへのアクセスを回復します。

- E メールアドレスのドメインを所有している場合は、削除した E メールアドレスを復元できます。または、E メールアカウントにキャッチオールを設定することもできます。「キャッチオール」は、メールサーバーに存在しなくなった E メールアドレスに送信されたすべてのメッセージをキャッチし、別のメールアドレスにリダイレクトします。
- アカウントの E メールアドレスが企業 E メールシステムの一部である場合は、IT システム管理者に連絡することをお勧めします。管理者は、E メールへのアクセス許可の回復を支援できる可能性があります。

それでも AWS アカウント にサインインできない場合、[\[Contact us\]](#) (お問い合わせ) で代替サポートオプションを見つけることができます。

## 関連情報

以下の記事では、ルートユーザーの操作に関するさらに詳細な情報を提供しています。

- [AWS アカウント とそのリソースを保護するためのベストプラクティスにはどのようなものがありますか?](#)
- [ルートユーザーが使用されたことを通知する EventBridge イベントルールを作成する方法とは?](#)
- [AWS アカウントのルートユーザー アクティビティを監視して通知する](#)
- [IAM ルートユーザーのアクティビティを監視する](#)

# IAM ユーザー

## ⚠ Important

IAM [ベストプラクティス](#)では、長期的な認証情報を持つIAMユーザーを使用するのではなく、一時的な認証情報を使用して AWS にアクセスするために、IDプロバイダーとのフェデレーションを使用することを人間ユーザーに求めることを推奨します。

AWS Identity and Access Management (IAM) のユーザーとは、AWS で作成したエンティティのことです。IAM ユーザーは、AWS とやり取りするために IAM ユーザーを使用する人間のユーザーまたはワークロードを表します。AWS のユーザーは名前と認証情報で構成されます。

管理者アクセス許可を持つ IAM ユーザーが AWS アカウントのルートユーザー ということではありません。ルートユーザーの詳細については、「[AWS アカウントのルートユーザー](#)」を参照してください。

## AWS が IAM ユーザーを識別する方法

IAM ユーザーを作成すると、IAM はそのユーザーを識別するための以下の手段を作成します。

- IAM ユーザーの「フレンドリ名」。IAM ユーザーを作成したときに指定した名前です (Richard、Anaya など)。これは、AWS Management Console に表示される名前です。
- IAM ユーザーの Amazon リソースネーム (ARN)。AWS 全体で IAM ユーザーを一意に識別する必要がある場合は、ARN を使用します。例えば、ARN を使用して、Amazon S3 バケットの IAM ポリシーの Principal として IAM ユーザーを指定できます。IAM ユーザーの ARN は次の例のように表示されます。

```
arn:aws:iam::account-ID-without-hyphens:user/Richard
```

- IAM ユーザー用の一意の識別子。この ID が返されるのは、API、Tools for Windows PowerShell または AWS CLI を使用して IAM ユーザーを作成した場合だけです。この ID はコンソールには表示されません。

これらの ID の詳細については[IAM ID](#)を参照してください。

## IAM ユーザーと認証情報

AWS には、IAM ユーザーの認証情報に応じてさまざまな方法でアクセスできます。

- [コンソールパスワード](#): AWS Management Console などのインタラクティブセッションにサインインするときに IAM ユーザーが入力するパスワード。IAM ユーザーのパスワード (コンソールアクセス) を無効化すると、サインイン認証情報を使用して AWS Management Console にサインインできなくなります。権限を変更したり、引き受けたロールを使用してコンソールにアクセスできないようにしたりすることはありません。
- [アクセスキー](#): AWS へのプログラムによる呼び出しに使用されます。ただし、IAM ユーザーのアクセスキーを作成する前に検討すべきより安全な代替手段があります。詳細については、「AWS 全般のリファレンス」の「[長期的なアクセスキーの考慮事項と代替方法](#)」を参照してください。IAM ユーザーがアクティブなアクセスキーを持っている場合、それらのキーは引き続き機能し、AWS CLI、Tools for Windows PowerShell、AWS API、または AWS Console Mobile Application 用のツールを介したアクセスを許可します。
- [CodeCommit で使用する SSH キー](#): CodeCommit での認証に使用可能な OpenSSH 形式の SSH パブリックキー。
- [サーバー証明書](#): AWS の一部のサービスでの認証に使用可能な SSL/TLS 証明書。AWS Certificate Manager (ACM) を使用してサーバー証明書のプロビジョニング、管理、デプロイを行うことをお勧めします。ACM でサポートされていないリージョンで HTTPS 接続をサポートする必要があるときにのみ、IAM を使用してください。ACM をサポートするリージョンについては、「AWS 全般のリファレンス」の「[AWS Certificate Manager エンドポイントとクォータ](#)」を参照してください。

IAM ユーザーに適した認証情報を選択できます。AWS Management Console を使用して IAM ユーザーを作成するときは、少なくともコンソールパスワードまたはアクセスキーを含める選択をする必要があります。デフォルトでは、AWS CLI または AWS API を使用して作成された新しい IAM ユーザーには、どのような種類の認証情報も提供されていません。ユースケースに基づいて、IAM ユーザーに合った種類の認証情報を作成する必要があります。

パスワード、アクセスキー、および多要素認証 (MFA) デバイスの管理には、以下のオプションがあります。

- [IAM ユーザーのパスワードを管理します。](#)。AWS Management Console へのアクセスを許可するパスワードを作成および変更します。最低限のパスワードの複雑さを強制するパスワードポリシーを設定します。自分のパスワードの変更をユーザーに許可します。
- [IAM ユーザーのアクセスキーを管理します。](#) アカウントのリソースにプログラムでアクセスするためのアクセスキーを作成および更新します。
- [IAM ユーザーの多要素認証 \(MFA\) を有効化する。](#) [ベストプラクティス](#)として、アカウントのすべての IAM ユーザーに対して、多要素認証を要求することをお勧めします。MFA を使用する場合、

ユーザーは 2 つの形式の ID を指定する必要があります。まず、ユーザー ID の一部である認証情報 (パスワードまたはアクセスキー) を指定します。さらに、一時的な数値コードを指定します。この数値コードは、ハードウェアデバイスか、スマートフォンまたはタブレットのアプリケーションで生成されます。

- [使用されていないパスワードおよびアクセスキーを見つけます](#)。アカウントのパスワードまたはアクセスキーを持つユーザー、またはアカウント内の IAM ユーザーであれば、だれでも AWS リソースにアクセスできます。セキュリティ保護のための[ベストプラクティス](#)は、ユーザーがパスワードやアクセスキーを使用しなくなったら、それらを削除することです。
- [アカウントの認証情報レポートをダウンロードします](#)。アカウント内のすべての IAM ユーザーと、ユーザーの各種認証情報 (パスワード、アクセスキー、MFA デバイスなど) のステータスが示された認証情報レポートを生成し、ダウンロードできます。パスワードおよびアクセスキーについては、パスワードやアクセスキーが最近いつ使用されたかが、認証情報レポートに表示されます。

## IAM ユーザーおよびアクセス許可

デフォルトでは、新しい IAM ユーザーには、何かを実施するための[アクセス許可](#)がありません。AWS オペレーションの実行や AWS リソースへのアクセスを承認されていません。個々の IAM ユーザーを持つ利点は、アクセス許可を各ユーザーに個別に割り当てることができることです。管理アクセス許可を複数のユーザーに割り当てることができます。これらのユーザーは、AWS リソースを管理するだけでなく、他の IAM ユーザーを作成して管理することもできます。ただし、通常、ユーザーのアクセス許可はユーザーの作業に必要なタスク (AWS アクションまたはオペレーション) とリソースだけに制限します。

Diego というユーザーを想定します。IAM ユーザー Diego を作成する際には、そのパスワードを作成し、特定の Amazon EC2 インスタンスの開始と Amazon RDS データベース内のテーブルの情報の読み取り (GET) を行うことができるアクセス許可をアタッチします。ユーザーを作成して初期認証情報とアクセス許可を付与する手順については、「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。既存のユーザーのアクセス許可を変更する手順については、「[IAM ユーザーのアクセス許可の変更](#)」を参照してください。ユーザーのパスワードやアクセスキーを変更する手順については、「[AWS でのユーザーパスワードの管理](#)」と「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

IAM ユーザーにアクセス許可の境界を追加することもできます。アクセス許可の境界は、AWS 管理ポリシーを使用してアイデンティティベースのポリシーで IAM ユーザーまたはロールに付与できるアクセス許可の上限を設定できる高度な機能です。ポリシーのタイプと用途の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。



## IAM ユーザーとアカウント

各 IAM ユーザーが関連付けられる AWS アカウントは 1 つだけです。IAM ユーザーは AWS アカウント内で定義されているため、AWS のファイルに対する支払方法を持つ必要はありません。アカウント内の IAM ユーザーが実行したすべての AWS アクティビティは、お客様のアカウントに請求されます。

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

## IAM ユーザーとサービスアカウント

IAM ユーザーは、関連付けられた認証情報とアクセス権限を持つ、IAM のリソースです。IAM ユーザーは、認証情報を使用して AWS リクエストを行う人物またはアプリケーションを表すことができます。これは通常、サービスアカウントと呼ばれます。IAM ユーザーの長期的な認証情報を使用することを選択した場合は、アプリケーションコードに直接アクセスキーを埋め込まないでください。AWS SDK と AWS Command Line Interface では、既知のロケーションにアクセスキーを置くことができるため、コードで保持する必要はありません。詳細については、「AWS 全般のリファレンス」の「[IAM ユーザーアクセスキーを適切に管理する](#)」を参照してください。または、ベストプラクティスとして、[長期的なアクセスキーの代わりに一時的なセキュリティ認証情報 \(IAM ロール\) を使用できます](#)。

## AWS アカウントでの IAM ユーザーの作成

 [Follow us on Twitter](#)

### Important

IAM [ベストプラクティス](#)では、長期的な認証情報を持つ IAM ユーザーを使用するのではなく、一時的な認証情報を使用して AWS にアクセスするために、ID プロバイダーとのフェデレーションを使用することを人間ユーザーに求めることを推奨します。

### Note

ウェブサイトで Amazon 製品を販売するための Product Advertising API に関する情報を探していてこのページを見つけた場合は、「[Product Advertising API 5.0 ドキュメント](#)」を参照してください。



このページに IAM コンソールから到達した場合は、サインイン済みであっても、アカウントに IAM ユーザーが含まれていない可能性があります。AWS アカウントのルートユーザー、ロール、一時的な認証情報のいずれかを使用してサインインします。これらの IAM 認証情報については、「[IAM ID \(ユーザー、ユーザーグループ、ロール\)](#)」を参照してください。

ユーザーを作成して、このユーザーに作業タスクの実行を許可するステップは以下のとおりです。

1. AWS Management Console、AWS CLI、Tools for Windows PowerShell で、または AWS API オペレーションを使用してユーザーを作成します。AWS Management Console でユーザーを作成する場合は、選択内容に基づいてステップ 1 ~ 4 が自動的に処理されます。ユーザーをプログラムにより作成した場合、各ステップを別個に実行する必要があります。
2. ユーザーに必要なアクセスのタイプに応じてユーザーの認証情報を作成します。
  - [コンソールアクセスを有効にする - オプション]: ユーザーが AWS Management Console にアクセスする必要がある場合は、[該当ユーザーのパスワードを作成します](#)。ユーザーのコンソールアクセスを無効にすると、ユーザー名とパスワードを使用して AWS Management Console にサインインできなくなります。アクセス許可を変更したり、想定されたロールを使用してコンソールにアクセスすることを妨げたりすることはありません。

#### Tip

ユーザーが必要とする認証情報のみを作成します。例えば、AWS Management Console を介したアクセスのみ必要なユーザーには、アクセスキーを作成しないでください。

3. ユーザーを 1 つ以上のグループに追加することで、必要なタスクを実行するアクセス権限を付与します。アクセス許可ポリシーをユーザーに直接アタッチして、アクセス許可を付与することもできます。ただし、ユーザーをグループに配置し、アクセス権限の管理は、グループにアタッチされているポリシーを通して行うことをお勧めします。[アクセス許可の境界](#)を使用してユーザーのアクセス許可を制限することもできます。ただし、これは一般的ではありません。
4. (オプション) タグをアタッチして、メタデータをユーザーに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
5. 必要なサインイン情報をユーザーに提供します。この情報には、ユーザーがアカウントのサインインページで認証情報として入力するパスワードやコンソールの URL が含まれます。詳細については、「[IAM ユーザーが AWS にサインインする方法](#)」を参照してください。

6. (オプション) ユーザーの [多要素認証 \(MFA\)](#) を設定します。MFA では、ユーザーが AWS Management Console にサインインするたびに 1 回限り使用のコードを入力することが求められます。
7. (オプション) ユーザーに自分の認証情報を管理する権限を与えることができます。(デフォルト設定では、自身の認証情報を管理する権限は、ユーザーにはありません) 詳細については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」を参照してください。

ユーザーの作成に必要なアクセス許可の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

## トピック

- [IAM ユーザーの作成 \(コンソール\)](#)
- [IAM ユーザーの作成 \(AWS CLI\)](#)
- [IAM ユーザーの作成 \(AWS API\)](#)

## IAM ユーザーの作成 (コンソール)

IAM ユーザーは AWS Management Console で作成できます。

### IAM ユーザーを作成するには (コンソール)

1. AWS サインインユーザーガイドの「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従ってください。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. ナビゲーションペインで [ユーザー]、[ユーザーを追加] の順に選択します。
4. [ユーザーの詳細を指定] ページの [ユーザーの詳細] の [ユーザー名] に、新しいユーザーの名前を入力します。これは、AWS のサインイン名です。

#### Note

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。ユーザー名は、最大 64 文字の英数字、プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (\_)、ハイフン (-) を組み合わせて指定します。名前はアカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、TESTUSER というユーザーと testuser というユーザーを作成することはできません。ユーザー名をポリシーま

または ARN の一部として使用する場合、名前の大文字と小文字が区別されます。サインイン中など、コンソールにユーザー名が表示される場合、大文字と小文字は区別されません。

5. [ユーザーアクセスを — AWS Management Console へ提供するオプション] を選択します。これにより、新しいユーザーの AWS Management Console サインイン認証情報が生成されます。

個人にコンソールアクセスを提供しているかどうかを尋ねられます。IAM ではなく IAM Identity Center でユーザーを作成することをお勧めします。

- IAM Identity Center でのユーザーの作成に切り替えるには、[Identity Center でユーザーを指定] を選択します。

IAM Identity Center を有効にしていない場合は、このオプションを選択するとコンソールのサービスページが表示され、サービスを有効にできます。この手順の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[IAM Identity Center で一般的なタスクを開始する](#)」を参照してください。

IAM ID Center を有効にしている場合、このオプションを選択すると、IAM ID Center の [ユーザーの詳細を指定] ページに移動します。この手順の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[ユーザーの追加](#)」を参照してください

- IAM Identity Center を使用できない場合は、[IAM ユーザーを作成する] を選択し、以下の手順を続けてください。
  - a. [コンソールのパスワード] で、以下のいずれかを選択します。
    - [自動生成パスワード] – ユーザーは、[アカウントのパスワードポリシー](#)の要件を満たすランダムに生成されたパスワードを取得します。[パスワードの取得] ページに到達すると、パスワードを表示またはダウンロードできます。
    - [カスタムパスワード] – ボックスに入力したパスワードがユーザーに割り当てられます。
  - b. (オプション) [ユーザーは次回のサインイン時に新しいパスワードを作成する必要がある (推奨)] がデフォルトで選択されており、ユーザーに初回サインイン時にパスワードを強制的に変更させることができます。

**Note**

管理者が「[ユーザーにパスワードの変更を許可](#)」のアカウントのパスワードポリシー設定」を有効にしている場合、このチェックボックスには何の効果もありません

ん。それ以外の場合は、[IAMUserChangePassword](#) という名前の AWS マネージドポリシーが自動的に新しいユーザーにアタッチされます。ポリシーは、ユーザーに対して、各自のパスワードを変更するためのアクセス許可を付与します。

6. [次へ] を選択します。
7. [アクセス許可の設定] ページで、このユーザーにアクセス許可を割り当てる方法を指定します。以下の 3 つのオプションのいずれかを選択します。
  - [ユーザーをグループに追加する] – アクセス許可ポリシーがすでに付与されている 1 つ以上のグループにユーザーを割り当てる場合は、このオプションを選択します。IAM は、アカウント内のグループおよびアタッチされているポリシーを一覧表示します。1 つ以上の既存のグループを選択するか、[グループの作成] を選択して新しいグループを作成する必要があります。詳細については、「[IAM ユーザーのアクセス許可の変更](#)」を参照してください。
  - [アクセス許可のコピー] – グループメンバーシップ、アタッチされている管理ポリシー、埋め込まれているインラインポリシー、および既存の[アクセス許可の境界](#)のすべてを既存のユーザーから新しいユーザーにコピーする場合は、このオプションを選択します。IAM は、アカウント内のユーザーを一覧表示します。アクセス許可が新しいユーザーのニーズに最も近いユーザーを選択します。
  - [ポリシーを直接アタッチする] – アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーを一覧表示する場合は、このオプションを選択します。ユーザーにアタッチするポリシーを選択します。または、[ポリシーの作成] を選択して、新しいブラウザタブを開き、新しいポリシーを作成します。詳細については、「[IAM ポリシーの作成](#)」のステップ 4 を参照してください。ポリシーを作成したら、そのタブを閉じて元のタブに戻り、ユーザーにポリシーを追加します。

 Tip

可能な限り、ポリシーをグループにアタッチし、ユーザーを適切なグループのメンバーにします。

8. (オプション) [アクセス許可の境界](#)を設定します。これはアドバンスド機能です。

[アクセス許可の境界] セクションを開き、[アクセス許可の境界を使用してアクセス許可の上限を設定する] を選択します。IAM は、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーを一覧表示します。アクセス許可の境界として使用するポリシーを選択するか、[ポリシーの作成] を選択して新しいブラウザタブを開き、新しいポリシーを作成します。詳細について

は、「[IAM ポリシーの作成](#)」のステップ 4 を参照してください。ポリシーを作成したら、そのタブを閉じて元のタブに戻り、アクセス許可の境界として使用するポリシーを選択します。

9. [次へ] を選択します。
10. (オプション) [レビューと作成] ページの [タグ] で [新しいタグを追加] を選択し、ユーザーにキーと値のペアとしてタグをアタッチし、メタデータを追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
11. この時点までに行ったすべての選択を確認します。続行する準備ができたなら、[ユーザーの作成] を選択します。
12. [パスワードの取得] ページで、ユーザーに割り当てられたパスワードを取得します。
  - パスワードの横にある [表示] を選択すると、ユーザーのパスワードが表示され、手動で記録できます。
  - [csv のダウンロード] を選択すると、ユーザーのサインイン認証情報が .csv ファイルとしてダウンロードされ、安全な場所に保存できます。
13. [メールのサインイン方法] を選択します。ローカルメールクライアントに下書きが表示され、カスタマイズしてユーザーに送信できます。メールのテンプレートは、各ユーザーの以下の詳細が含まれています。
  - [ユーザーネーム]
  - アカウントのサインインページへの URL。次の例を使用して、適切なアカウント ID またはアカウントエイリアスと置き換えます。

```
https://AWS-account-ID or alias.signin.aws.amazon.com/console
```

#### Important

ユーザーのパスワードは、生成されたメールには記載されていません。パスワードは、組織のセキュリティガイドラインに従った方法でユーザーに提供する必要があります。

14. ユーザーがプログラムによるアクセス用にアクセスキーも必要とする場合は、「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

## IAM ユーザーの作成 (AWS CLI)

IAM ユーザーは AWS CLI で作成できます。

## IAM ユーザーを作成するには (AWS CLI)

1. ユーザーを作成します。
    - [aws iam create-user](#)
  2. (オプション) ユーザーに AWS Management Console へのアクセス権を付与します。これにはパスワードが必要です。また、[アカウントのサインインページの URL](#) をユーザーに提供する必要があります。
    - [aws iam create-login-profile](#)
  3. (オプション) ユーザーにプログラムによりアクセス権を付与します。これにはアクセスキーが必要です。
    - [aws iam create-access-key](#)
    - Tools for Windows PowerShell: [New-IAMAccessKey](#)
    - IAM API: [CreateAccessKey](#)
- ⚠ Important**

シークレットアクセスキーを表示またはダウンロードできるのはこのときだけです。AWS API を使用する前に、ユーザーにこの情報を提供する必要があります。ユーザーの新しいアクセスキー ID とシークレットアクセスキーは、安全な場所に保存してください。このステップを行った後に、シークレットキーに再度アクセスすることはできません。
4. ユーザーを 1 つまたは複数のグループに追加します。指定したグループには、ユーザーに対して適切なアクセス権限を付与するポリシーがアタッチされている必要があります。
    - [aws iam add-user-to-group](#)
  5. (オプション) ユーザーのアクセス権限を定義するポリシーをユーザーにアタッチします。注意: ユーザーのアクセス許可の管理は、ユーザーをグループに追加してグループにポリシーをアタッチすることで (ユーザーに直接アタッチするのではなく) 行うことをお勧めします。
    - [aws iam attach-user-policy](#)
  6. (オプション) タグをアタッチして、カスタム属性をユーザーに追加します。詳細については、「[IAM ユーザーのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

7. (オプション) ユーザーに自身のセキュリティ認証情報を管理する権限を与えることができます。詳細については、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

## IAM ユーザーの作成 (AWS API)

IAM ユーザーは AWS API で作成できます。

(AWS API) から IAM ユーザーを作成するには

1. ユーザーを作成します。
  - [CreateUser](#)
2. (オプション) ユーザーに AWS Management Console へのアクセス権を付与します。これにはパスワードが必要です。また、[アカウントのサインインページの URL](#) をユーザーに提供する必要があります。
  - [CreateLoginProfile](#)
3. (オプション) ユーザーにプログラムによりアクセス権を付与します。これにはアクセスキーが必要です。
  - [CreateAccessKey](#)

### Important

シークレットアクセスキーを表示またはダウンロードできるのはこのときだけです。AWS API を使用する前に、ユーザーにこの情報を提供する必要があります。ユーザーの新しいアクセスキー ID とシークレットアクセスキーは、安全な場所に保存してください。このステップを行った後に、シークレットキーに再度アクセスすることはできません。

4. ユーザーを 1 つまたは複数のグループに追加します。指定したグループには、ユーザーに対して適切なアクセス権限を付与するポリシーがアタッチされている必要があります。
  - [AddUserToGroup](#)
5. (オプション) ユーザーのアクセス権限を定義するポリシーをユーザーにアタッチします。注意: ユーザーのアクセス許可の管理は、ユーザーをグループに追加してグループにポリシーをアタッチすることで (ユーザーに直接アタッチするのではなく) 行うことをお勧めします。



- [AttachUserPolicy](#)
6. (オプション) タグをアタッチして、カスタム属性をユーザーに追加します。詳細については、「[IAM ユーザーのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。
  7. (オプション) ユーザーに自身のセキュリティ認証情報を管理する権限を与えることができます。詳細については、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

## AWS Management Console への IAM ユーザーアクセスのコントロール

AWS Management Console を使用して AWS アカウント にサインインするアクセス許可を持つ IAM ユーザーは、AWS リソースにアクセスできます。以下の一覧は、AWS Management Console を通じて AWS アカウント リソースへのアクセス権限を IAM ユーザーに付与する方法を示しています。また、AWS ウェブサイトを通じて、AWS アカウントの他の機能に IAM ユーザーがアクセスする方法も示しています。

### Note

IAM の使用は無料です。

### AWS Management Console

AWS Management Console にアクセスする必要がある各 IAM ユーザーのパスワードを作成します。ユーザーは、IAM 対応の AWS アカウント サインインページから、コンソールにアクセスします。サインインページへのサインインについての詳細は、「AWS サインイン ユーザーガイド」の「[AWS にサインインする方法](#)」を参照してください。パスワード作成の詳細については、[AWS でのユーザーパスワードの管理](#)を参照してください。

パスワードを削除することで、AWS Management Console への IAM ユーザーアクセスを無効にできます。これにより、サインイン認証情報を使用して AWS Management Console にサインインすることができなくなります。権限を変更したり、引き受けたロールを使用してコンソールにアクセスできないようにしたりすることはありません。ユーザーがアクティブなアクセスキーを持っている場合、それらのキーは引き続き機能し、AWS CLI、Tools for Windows PowerShell、AWS API、または AWS Console Mobile Application 用のツールを介したアクセスを許可します。

## Amazon EC2 インスタンス、Amazon S3 バケットなどの AWS リソース

IAM ユーザーがパスワードを持っている場合も、AWS リソースにアクセスするにはアクセス許可が必要です。お客様が IAM ユーザーを作成した際、デフォルトではそのユーザーにアクセス許可はありません。必要なアクセス許可を IAM ユーザーに付与するには、ユーザーにポリシーをアタッチします。多数の IAM ユーザーが同じリソースで同じタスクを実行する場合、これらの IAM ユーザーをグループに割り当てることができます。次に、そのグループにアクセス許可を割り当てることができます。IAM ユーザーおよびグループの作成の詳細については、「[IAM ID \(ユーザー、ユーザーグループ、ロール\)](#)」を参照してください。ポリシーを使用するアクセス許可設定の詳細については、[AWS リソースのアクセス管理](#)を参照してください。

### AWS ディスカッションフォーラム

[AWS ディスカッションフォーラム](#)への投稿は、どなたでもお読みいただけます。ユーザーが質問やコメントを AWS ディスカッションフォーラムに投稿する場合は、自分のユーザーネームを使用することができます。ユーザーが初めて AWS ディスカッションフォーラムに投稿する場合、ユーザーはニックネームおよび E メールアドレスを入力するように求められます。AWS ディスカッションフォーラムでそのニックネームを使用できるのは、そのユーザーのみです。

### AWS アカウント の請求および使用情報

ユーザーに、AWS アカウント の請求および使用情報へのアクセスを許可することができます。詳細については、[AWS Billing ユーザーガイド](#) の「請求情報へのアクセスのコントロール」を参照してください。

### AWS アカウント プロファイル情報

ユーザーが AWS アカウント のプロフィール情報にアクセスすることはできません。

### AWS アカウント セキュリティ認証情報

ユーザーが AWS アカウント のセキュリティ認証情報にアクセスすることはできません。

#### Note

IAM ポリシーによるアクセスコントロールは、インターフェイスとは関係ありません。たとえば、AWS Management Console にアクセスするパスワードをユーザーに提供できません。AWS Management Console 内におけるユーザーのアクションを、ユーザー (またはユーザーが属するグループ) のポリシーで制御することができます。または、AWS への API コールを実行するための AWS アクセスキーをユーザーに提供することもできます。これらのア

クセスキーを認証に使用するライブラリまたはクライアントを通じてユーザーが呼び出すことのできるアクションをポリシーで制御することもできます。

## IAM ユーザーが AWS にサインインする方法

IAM ユーザーとして AWS Management Console にサインインするには、ユーザー ID またはアカウントエイリアスを入力する必要があります。管理者が [コンソールで IAM ユーザーを作成しました](#) にサインインすると、ユーザー名とアカウント ID またはアカウントエイリアスを含むアカウントサインインページの URL など、サインイン認証情報が送信されたはずですが、

```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

### ヒント

ウェブブラウザでアカウントのサインインページのブックマークを作成するには、アカウントのサインイン URL を手動でブックマークエントリに入力する必要があります。ウェブブラウザのブックマーク機能は使用しないでください。リダイレクトによってサインイン URL が不明確になるからです。

また、次の一般サインインエンドポイントでサインインして、アカウント ID またはアカウントエイリアスを手動で入力することもできます。

```
https://console.aws.amazon.com/
```

利便性のため、AWS サインインページはブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。次回、ユーザーが任意のページに移動したとき、AWS Management Console コンソールは Cookie を使用して、ユーザーをアカウントのサインインページにリダイレクトします。

管理者が IAM ユーザーIDに関連付けられているポリシーで指定した AWS リソースにのみアクセスできます。ユーザーがコンソールで操作するには、AWS リソースのリスト表示や作成など、コンソールが実行するアクションの実行権限が必要です。詳細については、[AWS リソースのアクセス管理](#) および [IAM アイデンティティベースのポリシーの例](#) を参照してください。

**Note**

組織に既存のアイデンティティシステムがある場合は、Single Sign-On (SSO) オプションを作成することができます。SSO を使用すると、ユーザーは IAM ユーザーアイデンティティを持たなくてもアカウントの AWS Management Console にアクセスできます。SSO では、ユーザーが組織のサイトにサインインしたり、AWS に個別にサインインする必要もなくなります。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

## CloudTrail でのログ記録サインインの詳細

CloudTrail を有効化してログにサインインイベントを記録する場合、CloudTrail がイベントを記録する場所を選択するしくみを認識しておく必要があります。

- ユーザーがコンソールに直接サインインすると、選択されたサービスのコンソールがリージョンをサポートするかどうかによって、グローバルまたはリージョンのサインインエンドポイントにリダイレクトされます。例えば、メインコンソールのホームページはリージョンをサポートするため、次の URL にサインインした場合:

```
https://alias.signin.aws.amazon.com/console
```

https://us-east-2.signin.aws.amazon.com などのリージョンのサインインエンドポイントにリダイレクトされ、そのリージョンのログでリージョンの CloudTrail ログエントリとなります。

一方、Amazon S3 コンソールはリージョンをサポートしないため、次の URL にサインインした場合:

```
https://alias.signin.aws.amazon.com/console/s3
```

AWS により https://signin.aws.amazon.com にあるグローバルのサインインエンドポイントにリダイレクトされ、グローバルの CloudTrail ログエントリとなります。

- リージョン対応のメインコンソールのホームページで次のような URL 構文を使ってサインインすることにより、手動で特定のリージョンのサインインエンドポイントをリクエストすることができます。

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```

AWS により ap-southeast-1 リージョンサインインエンドポイントにリダイレクトされ、そのリージョンの CloudTrail ログイベントとなります。

CloudTrail と IAM の詳細については、「[CloudTrail による IAM イベントのログ記録](#)」を参照してください。

ユーザーが、アカウントの操作のためにプログラムによるアクセスが必要な場合は、各ユーザーにアクセスキーペア (アクセスキー ID とシークレットアクセスキー) を作成できます。ただし、ユーザーのアクセスキーを作成する前に検討すべきより安全な代替手段があります。詳細については、「AWS 全般のリファレンス」の「[長期的なアクセスキーの考慮事項と代替方法](#)」を参照してください。

## IAM のサインインページでの MFA デバイスの使用

[多要素認証 \(MFA\)](#) デバイスで構成されているユーザーは、MFA デバイスを使用して AWS Management Console にサインインする必要があります。ユーザーがサインイン認証情報を入力した後、AWS はそのユーザーのアカウントを調べ、そのユーザーに MFA が必要かどうかを確認します。以下のトピックでは、MFA が必要なときにユーザーがサインインを完了する方法について説明します。

### トピック

- [複数の MFA デバイスが有効化された状態でログインする](#)
- [FIDO セキュリティキーでのサインイン](#)
- [仮想 MFA デバイスでのサインイン](#)
- [ハードウェア TOTP トークンを使用したサインイン](#)

### 複数の MFA デバイスが有効化された状態でログインする

ユーザーが AWS アカウント で複数の MFA デバイスを有効にした状態で、そのアカウントのルートユーザーまたは IAM ユーザーとして AWS Management Console にログインする場合でも、サインインに必要な MFA デバイスは 1 台のみです。パスワードによる認証を行ったユーザーは、使用する MFA デバイスタイプを選択し認証を完了します。その後、ユーザーは、選択したタイプのデバイスを使用した認証を求められます。

## FIDO セキュリティキーでのサインイン

MFA がユーザーに必要な場合は、2 番目のサインインページが表示されます。ユーザーは FIDO セキュリティキーをタップする必要があります。

### Note

Google Chrome をお使いの方は、[Verify your identity with amazon.com] (amazon.comで本人確認をしてください) と要求するポップアップが表示されますが、いずれのオプションも選択しないようにしてください。セキュリティキーをタップするだけでよいのです。

他の MFA デバイスとは異なり、FIDO のセキュリティキーは同期しなくなります。管理者は、FIDO セキュリティキーを紛失したり壊れたりした場合、そのキーを無効にすることができます。詳細については、「[MFA デバイスの無効化 \(コンソール\)](#)」を参照してください。

WebAuthn をサポートするブラウザおよび、AWS がサポートする FIDO 対応デバイスについては、「[パスキーとセキュリティキーを使用するためのサポートされる設定](#)」を参照してください。

## 仮想 MFA デバイスでのサインイン

MFA がユーザーに必要な場合は、2 番目のサインインページが表示されます。[MFA code (MFA コード)] ボックスに、ユーザーは MFA アプリケーションから提供される数値コードを入力する必要があります。

MFA コードが正しい場合、ユーザーは AWS Management Console にアクセスできます。このコードが間違っていると、ユーザーは別のコードで再試行できます。

仮想 MFA デバイスが同期しなくなることがあります。ユーザーが AWS Management Console へのサインインを何度か試みて失敗した場合、仮想 MFA デバイスを同期するよう求められます。ユーザーは画面の指示に従って仮想 MFA デバイスを同期できます。お客様の AWS アカウントのユーザーに代わってデバイスを同期する方法については、「[仮想デバイスとハードウェア MFA デバイスの再同期](#)」を参照してください。

## ハードウェア TOTP トークンを使用したサインイン

MFA がユーザーに必要な場合は、2 番目のサインインページが表示されます。ユーザーは、[MFA code] (MFA コード) ボックスに、ハードウェア TOTP トークンから提供される数値コードを入力する必要があります。



MFA コードが正しい場合、ユーザーは AWS Management Console にアクセスできます。このコードが間違っていると、ユーザーは別のコードで再試行できます。

ハードウェア TOTP トークンは、同期から外れることはありません。AWS Management Console へのサインインが、複数回連続して失敗したユーザーに対しては、MFA トークンデバイスと同期するように求められます。ユーザーは画面の指示に従って MFA トークンデバイスを同期できます。お客様の AWS アカウント のユーザーに代わってデバイスを同期する方法については、「[仮想デバイスとハードウェア MFA デバイスの再同期](#)」を参照してください。

## IAM ユーザーの管理

### Note

[ベストプラクティス](#)として、一時的な認証情報の使用により AWS にアクセスするには、人間のユーザーに対して ID プロバイダーとのフェデレーションの使用を必須とすることをお勧めします。ベストプラクティスに従えば、IAM ユーザーやグループを管理することにはなりません。管理するのではなく、ユーザーとグループは AWS の外部で管理され、フェデレーション ID として AWS リソースにアクセスできます。フェデレーション ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、Identity Center ディレクトリのユーザー、または ID ソースから提供された認証情報を使用して AWS のサービスにアクセスするユーザーです。フェデレーション ID は、ID プロバイダーが定義したグループを使用します。AWS IAM Identity Center を使用している場合は、IAM Identity Center でのユーザーとグループの作成に関する情報について、AWS IAM Identity Center ユーザーガイドの「[Manage identities in IAM Identity Center](#)」(IAM Identity Center での ID の管理) を参照してください。

Amazon Web Services には、AWS アカウント の IAM ユーザーを管理するための複数のツールが用意されています。アカウントまたは特定のグループの IAM ユーザーを一覧表示したり、ユーザーが属するすべてのユーザーグループを一覧表示したりできます。また、IAM ユーザーの名前変更やパスの変更をしたりできます。IAM ユーザーではなくフェデレーション ID の使用に移行している場合は、AWS アカウントから IAM ユーザーを削除したり、ユーザーを非アクティブ化したりできます。

IAM ユーザーの管理ポリシーの追加、変更、または削除の詳細については、「[IAM ユーザーのアクセス許可の変更](#)」を参照してください。IAM ユーザーのインラインポリシーの管理の詳細については、「[IAM ID のアクセス許可の追加および削除](#)」、「[IAM ポリシーの編集](#)」、および「[IAM ポリシーを削除する](#)」を参照してください。ベストプラクティスとして、インラインポリシーではなく管理ポリシーを使用します。AWS 管理ポリシーでは、多くの一般的なユースケースでアクセス許可を付



与します。AWS 管理ポリシーは、すべての AWS のユーザーが使用できるため、特定のユースケースに対して最小特権のアクセス許可が付与されない場合があることに留意してください。そのため、ユースケースに応じた[顧客管理ポリシー](#)を定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「[AWS マネージドポリシー](#)」を参照してください。特定のジョブ機能用に設計された AWS 管理ポリシーの詳細については、[AWS ジョブ機能の管理ポリシー](#)を参照してください。

IAM ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。

#### Tip

[IAM Access Analyzer](#) では、IAM ロールが使用するサービスとアクションを分析し、使用可能なきめ細かなポリシーを生成できます。生成された各ポリシーをテストしたら、ポリシーを本番環境にデプロイできます。これにより、ワークロードに必要なアクセス許可のみが付与されます。ポリシー生成の詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

IAM ユーザーパスワードの管理に関する詳細については、「[IAM ユーザーのパスワードの管理](#)」を参照してください。

#### トピック

- [ユーザーアクセスの表示](#)
- [IAM ユーザーの一覧表示](#)
- [IAM ユーザーの名前の変更](#)
- [IAM ユーザーの削除](#)
- [IAM ユーザーの非アクティブ化](#)

#### ユーザーアクセスの表示

ユーザーを削除する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## IAM ユーザーの一覧表示

IAM ユーザーを一覧表示するには、AWS アカウント または特定の IAM ユーザーグループで、ユーザーが属するすべてのユーザーグループを一覧表示します。ユーザーの一覧表示に必要な権限の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

アカウントのすべてのユーザーを一覧表示するには

- [AWS Management Console](#): ナビゲーションペインで [ユーザー] を選択します。コンソールには、AWS アカウント 内のユーザーが表示されます。
- AWS CLI: [aws iam list-users](#)
- AWS API: [ListUsers](#)

特定のユーザーグループのユーザーを一覧表示するには

- [AWS Management Console](#): ナビゲーションペインで [ユーザーグループ] を選択し、ユーザーグループ名を選択してから、[ユーザー] タブを選択します。
- AWS CLI: [aws iam get-group](#)
- AWS API: [GetGroup](#)

ユーザーが所属しているすべてのユーザーグループを一覧表示するには

- [AWS Management Console](#): ナビゲーションペインで [ユーザー] を選択し、ユーザー名を選択してから、[グループ] タブを選択します。
- AWS CLI: [aws iam list-groups-for-user](#)
- AWS API: [ListGroupForUser](#)

## IAM ユーザーの名前の変更

ユーザーの名前またはパスを変更するには、AWS CLI、Tools for Windows PowerShell または AWS API を使用する必要があります。コンソールに、ユーザーの名前を変更するためのオプションはありません。ユーザーの名前の変更に必要なアクセス許可の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

ユーザーの名前またはパスを変更すると、以下のことが起こります。

- ユーザーにアタッチされているポリシーは、新しい名前のユーザーにそのままアタッチされています。
- ユーザーは名前が変わるだけで、所属するユーザーグループは変わりません。
- ユーザーの一意の ID は変更されません。一意の ID の詳細については、「[一意の識別子](#)」を参照してください。
- ユーザーをプリンシパル (このユーザーにはアクセスが許可されます) として参照しているリソースやロールのポリシーは、新しい名前またはパスを使用するように自動的に更新されます。例えば、Amazon S3 内のキューベースのポリシー、または Amazon SQS 内のリソースベースのポリシーは、新しい名前またはパスを使用するように自動的に更新されます。

IAM は、ユーザーをリソースとして参照しているポリシーを、自動的に更新しません。新しい名前またはパスを使用するには、手動で更新する必要があります。例えば、Richard というユーザーに自身の認証情報の管理を許可するポリシーがアタッチされているとします。名前を Richard から Rich に変更する場合、管理者はそのポリシーを更新して、次のリソースを

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Richard
```

次のように変更する必要があります。

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Rich
```

パスを変更する場合も同様です。管理者は、ユーザーの新しいパスを反映するようにポリシーを更新する必要があります。

ユーザーの名前を変更するには

- AWS CLI: [aws iam update-user](#)
- AWS API: [UpdateUser](#)

## IAM ユーザーの削除

ユーザーが退職した場合、お客様の AWS アカウント から IAM ユーザーを削除することがあります。ユーザーが一時的に不在の場合は、[IAM ユーザーの非アクティブ化](#)の説明に従って、アカウントから削除する代わりに、そのユーザーのアクセスを非アクティブ化できます。

トピック

- [IAM ユーザーの削除 \(コンソール\)](#)
- [IAM ユーザーの削除 \(AWS CLI\)](#)

## IAM ユーザーの削除 (コンソール)

AWS Management Console を使用して IAM ユーザーを削除すると、IAM によって以下の情報が自動的に削除されます。

- ユーザー
- すべてのグループメンバシップ (つまり、ユーザーは、メンバーとして所属していたすべての IAM ユーザーグループから削除されます)
- ユーザーのパスワード
- ユーザーのアクセスキー
- ユーザーに組み込まれていたすべてのインラインポリシー (ユーザーグループのアクセス権限を通じてユーザーに適用されているポリシーは影響を受けません)

### Note

IAM は、ユーザーを削除すると、ユーザーにアタッチされた管理ポリシーをすべて削除しますが、管理ポリシーは削除しません。

- [関連する MFA デバイス](#)

## IAM ユーザーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザー] を選択し、ユーザー名または行そのものではなく、削除するユーザー名の横にあるチェックボックスをオンにします。
3. ページの上部で、[削除] を選択します。
4. 確認ダイアログボックスで、テキスト入力フィールドにユーザー名を入力し、ユーザーの削除を確認します。[削除] を選択します。

## IAM ユーザーの削除 (AWS CLI)

AWS CLI を使用してユーザーを削除する場合、AWS Management Console の場合とは異なり、ユーザーにアタッチされている項目を削除する必要があります。この手順では、そのプロセスについて説明します。

アカウントからユーザーを削除するには (AWS CLI)

1. ユーザーのパスワード (使用していた場合) を削除します。

[aws iam delete-login-profile](#)

2. ユーザーのアクセスキーを削除します (使用していた場合)。

[aws iam list-access-keys](#) (ユーザーのアクセスキーを一覧表示するには) および [aws iam delete-access-key](#)

3. ユーザーの署名証明書を削除します。認証情報を削除すると、完全に削除され、復元できないことに注意してください。

[aws iam list-signing-certificates](#) (ユーザーの署名証明書を一覧表示するには) および [aws iam delete-signing-certificate](#)

4. ユーザーの SSH パブリックキーを削除します (使用していた場合)。

[aws iam list-ssh-public-keys](#) (ユーザーの SSH パブリックキーを一覧表示するには) および [aws iam delete-ssh-public-key](#)

5. ユーザーの Git 認証情報を削除します。

[aws iam list-service-specific-credentials](#) (ユーザーの Git 認証情報を一覧表示するには) および [aws iam delete-service-specific-credential](#)

6. ユーザーの多要素認証 (MFA) デバイスを無効にします (使用していた場合)。

[aws iam list-mfa-devices](#) (ユーザーの MFA デバイスを一覧表示するには)、[aws iam deactivate-mfa-device](#) (デバイスを無効にするには)、[aws iam delete-virtual-mfa-device](#) (仮想 MFA デバイスを完全に削除するには)

7. ユーザーのインラインポリシーを削除します。

[aws iam list-user-policies](#) (ユーザーのインラインポリシーを一覧表示するには) および [aws iam delete-user-policy](#) (ポリシーをさくじょするには)

8. ユーザーにアタッチされているすべての管理ポリシーをデタッチします。

[aws iam list-attached-user-policies](#) (ユーザーにアタッチされているポリシーを一覧表示するには) および [aws iam detach-user-policy](#) (ポリシーをデタッチするには)

- すべてのユーザーグループからユーザーを削除します。

[aws iam list-groups-for-user](#) (ユーザーが属するユーザーグループを一覧表示するには) および [aws iam remove-user-from-group](#)

- ユーザーを削除。

[aws iam delete-user](#)

## IAM ユーザーの非アクティブ化

IAM ユーザーが一時的に会社から離れるときは、そのユーザーを非アクティブ化する必要がある場合があります。IAM ユーザー認証情報を現状のままにしておき、AWS のアクセスをブロックすることができます。

ユーザーを非アクティブ化するには、AWS へのユーザーアクセスを拒否するポリシーを作成してアタッチします。ユーザーアクセスは後で復元できます。

ユーザーにアタッチしてアクセスを拒否できるポリシーを 2 例紹介します。

次のポリシーには期限を設けていません。ユーザーのアクセスを復元するには、ポリシーを削除する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

次のポリシーには、2024 年 12 月 24 日午後 11 時 59 分 (UTC) にポリシーを開始し、2025 年 2 月 28 日午後 11 時 59 分 (UTC) にポリシーを終了するという条件があります。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2024-12-24T23:59:59Z"},
      "DateLessThan": {"aws:CurrentTime": "2025-02-28T23:59:59Z"}
    }
  }
]
```

## IAM ユーザーのアクセス許可の変更

AWS アカウント 内の IAM ユーザーのアクセス許可を変更するには、グループメンバーシップを変更するか、既存のユーザーからアクセス許可をコピーするか、ユーザーに直接ポリシーをアタッチするか、または[アクセス許可の境界](#)を設定することができます。アクセス許可の境界では、ユーザーに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

ユーザーのアクセス権限の変更に必要なアクセス権限の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

### トピック

- [ユーザーアクセスの表示](#)
- [ユーザーのアクセスアクティビティに基づくポリシーの生成](#)
- [ユーザーへのアクセス許可の追加 \(コンソール\)](#)
- [ユーザーのアクセス許可の変更 \(コンソール\)](#)
- [ユーザーからのアクセス許可ポリシーの削除 \(コンソール\)](#)
- [ユーザーからのアクセス許可の境界の削除 \(コンソール\)](#)
- [ユーザーのアクセス許可の追加と削除 \(AWS CLI または AWS API\)](#)

## ユーザーアクセスの表示

ユーザーのアクセス許可を変更する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) か



ら削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## ユーザーのアクセスアクティビティに基づくポリシーの生成

IAM エンティティ (ユーザーまたはロール) に必要な権限を超えるアクセス許可を付与することがあります。付与するアクセス権限を調整するために、エンティティのアクセスアクティビティに基づく IAM ポリシーを生成できます。IAM Access Analyzer は AWS CloudTrail ログを確認し、指定した日付範囲内のロールが使用したアクセス許可を含むポリシーテンプレートを生成します。テンプレートを使用して、きめ細かなアクセス権限で管理ポリシーを作成し、それを IAM エンティティにアタッチできます。これにより、特定のユースケースでロールが AWS リソースとインタラクションするために必要なアクセス権限のみを付与します。詳細については、「[アクセスアクティビティに基づいてポリシーを生成する](#)」を参照してください。

## ユーザーへのアクセス許可の追加 (コンソール)

IAM では 3 つの方法でユーザーにアクセス許可ポリシーを追加できます。

- ユーザーをグループに追加する – IAM ユーザーをグループのメンバーにします。グループのポリシーがユーザーにアタッチされます。
- 既存のユーザーからアクセス許可をコピーする – すべてのグループメンバーシップ、アタッチされた管理ポリシー、インラインポリシー、および既存のアクセス許可の境界をソースユーザーからコピーします。
- ポリシーをユーザーに直接アタッチする – 管理ポリシーをユーザーに直接アタッチします。アクセス許可の管理を簡単にするために、ポリシーをグループに割り当ててから、ユーザーを適切なグループのメンバーにします。

### Important

ユーザーにアクセス許可の境界が設定されている場合は、アクセス許可の境界で許可されている以上のアクセス許可をユーザーに追加することはできません。

## ユーザーをグループに追加することによるアクセス権限の追加

ユーザーをグループに追加した結果は、すぐにユーザーに反映されます。

ユーザーをグループに追加することでアクセス許可をユーザーに追加するには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. コンソールの [グループ] 列で、ユーザーの現在のグループメンバーシップを確認します。必要に応じて、次の手順を実行して、その列をユーザーテーブルに追加します。

1. 右端のテーブルの上で、設定のシンボル



を選択します。

2. [Manage Columns (列の管理)] ダイアログボックスで、[グループ] 列を選択します。必要に応じて、ユーザーテーブルに表示しない列見出しのチェックボックスをオフにすることもできます。
3. [Close (閉じる)] を選択して、ユーザーのリストに戻ります。

[グループ] 列に、ユーザーが属するグループが表示されます。この列には、グループ名を 2 つまで表示できます。ユーザーが 3 つ以上のグループのメンバーである場合は、最初の 2 つのグループ (アルファベット順) が表示されます。その他のグループメンバーシップは数のみ表示されます。例えば、ユーザーが、グループ A、グループ B、グループ C、グループ D に所属している場合、フィールドには [Group A, Group B + 2 more (グループ A、グループ B、他 2 グループ)] という値が表示されます。ユーザーが所属している合計グループ数を表示するには、[Group count (グループ数)] 列をユーザーのテーブルに追加します。

4. 変更するアクセス許可を持つユーザーの名前を選択します。
5. [Permissions (アクセス許可)] タブを選択してから、[アクセス許可の追加] を選択します。[Add user to group] を選択します。
6. ユーザーが参加する各グループのチェックボックスをオンにします。リストには、各グループの名前と、そのグループのメンバーとなった場合にユーザーが受け取るポリシーが表示されます。
7. (オプション) 既存のグループの選択に加えて、[グループの作成] を選択して新しいグループを定義することができます。
  - a. 新しいタブで、[ユーザーグループ名] に新しいグループの名前を入力します。

**Note**

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。グループ名は、最大 128 文字の英数字、プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、ハイフン (-) を組み合わせて指定します。名前はアカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、TESTGROUP というグループと testgroup というグループを作成することはできません。

- b. グループにアタッチする管理ポリシーのチェックボックスを 1 つ以上オンにします。[ポリシーの作成] を選択して、新しい管理ポリシーを作成することもできます。その場合は、新しいポリシーが完成したらこのブラウザタブまたはウィンドウに戻り、[Refresh (更新)] を選択した後、グループにアタッチする新しいポリシーを選択します。詳細については、「[IAM ポリシーの作成](#)」を参照してください。
  - c. [ユーザーグループの作成] を選択します。
  - d. 元のタブに戻り、グループのリストを更新します。次に、新しいグループのチェックボックスをオンにします。
8. [次へ] を選択して、ユーザーに追加するグループメンバーシップのリストを表示します。次に、[アクセス許可の追加] を選択します。

### 別のユーザーにコピーすることによるアクセス権限の追加

アクセス許可をコピーした結果は、すぐにユーザーに反映されます。

別のユーザーからアクセス許可をコピーしてアクセス許可をユーザーに追加するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択して、変更するアクセス許可を持つユーザーの名前を選択し、[Permissions (アクセス許可)] タブを選択します。
3. [Add permissions (アクセス許可の追加)]、[Copy permissions from existing user (既存のユーザーからアクセス許可をコピー)] の順に選択します。リストには、使用可能なユーザーと、そのグループメンバーシップ、アタッチされたポリシーが表示されます。グループやポリシーの完全なリストが 1 行に収まらない場合は、[and *n* more (さらに *n* 個追加)] リンクを選択できます。これを行うこと、新しいブラウザタブを開き、ポリシー ([Permissions (アクセス許可)] タブ)、グループ ([グループ] タブ) の詳細なリストが表示されます。

4. コピーするアクセス権限を持つユーザーの横のラジオボタンをオンにします。
5. [次へ] を選択して、ユーザーに加える変更のリストを表示します。次に、[アクセス許可の追加] を選択します。

ポリシーをユーザーに直接アタッチすることによるアクセス権限の追加

ポリシーをアタッチした結果は、すぐにユーザーに反映されます。

管理ポリシーを直接アタッチすることでユーザーにアクセス許可を追加するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択して、変更するアクセス許可を持つユーザーの名前を選択し、[Permissions (アクセス許可)] タブを選択します。
3. [アクセス許可の追加]、[ポリシーを直接アタッチする] の順に選択します。
4. ユーザーにアタッチする管理ポリシーのチェックボックスを 1 つ以上オンにします。[ポリシーの作成] を選択して、新しい管理ポリシーを作成することもできます。その場合は、新しいポリシーが完成したらこのブラウザタブまたはウィンドウに戻ります。[Refresh (更新)] を選択した後、ユーザーにアタッチする新しいポリシーのチェックボックスをオンにします。詳細については、「[IAM ポリシーの作成](#)」を参照してください。
5. [次へ] を選択して、ユーザーにアタッチするポリシーのリストを表示します。次に、[アクセス許可の追加] を選択します。

ユーザーのアクセス許可の境界の設定

アクセス許可の境界を設定した結果は、すぐにユーザーに反映されます。

ユーザーのアクセス許可の境界を設定するには

1. AWS Management Console にサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. アクセス許可の境界を変更する対象のユーザーの名前を選択します。
4. [Permissions] (許可) タブを選択します。必要に応じて、[アクセス許可の境界] セクションを開き、[境界の設定] を選択します。
5. アクセス許可の境界として使用するポリシーを選択します。

6. [Set boundary (境界の設定)] を選択します。

## ユーザーのアクセス許可の変更 (コンソール)

IAM では、次の方法でユーザーに関連付けられているアクセス許可を変更できます。

- アクセス許可ポリシーの編集 – ユーザーのインラインポリシーまたはユーザーのグループのインラインポリシーを編集するか、ユーザーにアタッチされている管理ポリシーを直接またはグループを介して編集します。ユーザーにアクセス許可の境界が設定されている場合は、アクセス許可の境界で使用されているポリシーで許可される以上のアクセス許可をユーザーに付与することはできません。
- アクセス許可の境界の変更 – ユーザーのアクセス許可の境界として使用されているポリシーを変更します。この変更に伴って、ユーザーに許可されるアクセス許可の上限が拡張または縮小される場合があります。

### ユーザーにアタッチされているアクセス許可ポリシーの編集

アクセス許可を変更した結果は、すぐにユーザーに反映されます。

### ユーザーにアタッチされている管理ポリシーを編集する

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. アクセス許可ポリシーを変更する対象のユーザーの名前を選択します。
4. [Permissions] (許可) タブを選択します。必要に応じて、[Permissions policies (アクセス許可ポリシー)] セクションを開きます。
5. ポリシーの詳細を表示するために編集するポリシーの名前を選択します。[ポリシーの用途] タブを選択し、ポリシーを編集した場合に影響を受ける可能性がある他のエンティティを確認します。
6. [Permissions (アクセス許可)] タブを選択し、ポリシーで付与されるアクセス許可を確認します。[ポリシーの編集] を選択します。
7. ポリシーを編集し、[ポリシー検証](#)に関する推奨事項を解決します。詳細については、「[IAM ポリシーの編集](#)」を参照してください。
8. [ポリシーの確認] を選択し、ポリシーの概要を確認します。次に、[変更の保存] を選択します。

## ユーザーのアクセス許可の境界の変更

アクセス許可の境界を変更した結果は、すぐにユーザーに反映されます。

ユーザーのアクセス許可の境界を設定するために使用されているポリシーを変更するには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. アクセス許可の境界を変更する対象のユーザーの名前を選択します。
4. [Permissions] (許可) タブを選択します。必要に応じて、[Permissions boundary (アクセス許可の境界)] セクションを開き、[Change boundary (境界の変更)] を選択します。
5. アクセス許可の境界として使用するポリシーを選択します。
6. [Set boundary (境界の設定)] を選択します。

## ユーザーからのアクセス許可ポリシーの削除 (コンソール)

ポリシーを削除した結果は、すぐにユーザーに反映されます。

IAM ユーザーのアクセス許可を取り消すには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. アクセス許可の境界を削除する対象のユーザーの名前を選択します。
4. [Permissions] (許可) タブを選択します。
5. 既存のポリシーを削除してアクセス許可を取り消す場合は、[タイプ] を表示して、ユーザーがそのポリシーを取得する方法を理解してから、[削除] を選択してポリシーを削除します。
  - グループメンバーシップのためにポリシーが適用されている場合、[削除] を選択して、ユーザーをそのグループから削除します。1つのグループには複数のポリシーがアタッチされている場合があります。グループからユーザーを削除すると、そのユーザーは、グループメンバーシップを通じて受け取ったすべてのポリシーへのアクセスを失います。
  - ポリシーがユーザーに直接アタッチされた管理ポリシーの場合、[削除] を選択して、ユーザーへのポリシーのアタッチを解除します。これは、そのポリシー自体やそのポリシーがアタッチされている他のエンティティに影響を与えません。

- ポリシーがインライン埋め込みポリシーである場合は、[X] を選択すると、IAM からポリシーが削除されます。ユーザーに直接アタッチされたインラインポリシーは、そのユーザーにのみ存在します。

## ユーザーからのアクセス許可の境界の削除 (コンソール)

アクセス許可の境界を削除した結果は、すぐにユーザーに反映されます。

ユーザーからアクセス許可の境界を削除するには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. アクセス許可の境界を削除する対象のユーザーの名前を選択します。
4. [Permissions] (許可) タブを選択します。必要に応じて、[Permissions boundary (アクセス許可の境界)] セクションを開き、[Remove boundary (境界の削除)] を選択します。
5. [境界の削除] を選択してアクセス許可の境界を削除することを確定します。

## ユーザーのアクセス許可の追加と削除 (AWS CLI または AWS API)

アクセス許可をプログラムにより追加または削除するには、グループメンバーシップの追加または削除、管理ポリシーのアタッチまたはデタッチ、インラインポリシーの追加または削除のいずれかを行う必要があります。詳細については、次のトピックを参照してください。

- [IAM グループへのユーザーの追加と削除](#)
- [IAM ID のアクセス許可の追加および削除](#)

## AWS でのユーザーパスワードの管理

アカウント内の IAM ユーザーのパスワードを管理できます。IAM ユーザーが AWS Management Console にアクセスするには、パスワードが必要です。ユーザーが、AWS Tools for Windows PowerShell、AWS CLI SDK または API を使用してプログラムで AWS リソースにアクセスする場合、パスワードは必要ありません。このような環境では、IAM ユーザーに [アクセスキー](#) を割り当てることができます。ただし、アクセスキーに代わるより安全な代替手段が他にもあり、最初に検討することをお勧めします。詳細については、「[AWS セキュリティ認証情報](#)」を参照してください。



## 内容

- [IAM ユーザー用のアカウントパスワードポリシーの設定](#)
- [IAM ユーザーのパスワードの管理](#)
- [IAM ユーザーに自分のパスワードを変更する権限を付与する](#)
- [IAM ユーザーが自分のパスワードを変更する方法](#)

## IAM ユーザー用のアカウントパスワードポリシーの設定

IAM ユーザーのパスワードの複雑度の要件や必須のローテーション期間を指定するためのカスタムパスワードポリシーを AWS アカウント に設定できます。カスタムパスワードポリシーを設定しない場合、IAM ユーザーのパスワードはデフォルトの AWS パスワードポリシーの要件を満たす必要があります。詳細については、「[カスタムパスワードポリシーのオプション](#)」を参照してください。

### トピック

- [パスワードポリシーの設定に関するルール](#)
- [パスワードポリシーを設定するために必要なアクセス許可](#)
- [デフォルトのパスワードポリシー](#)
- [カスタムパスワードポリシーのオプション](#)
- [パスワードポリシーの設定 \(コンソール\)](#)
- [パスワードポリシーの設定 \(AWS CLI\)](#)
- [パスワードポリシーの設定 \(AWS API\)](#)

### パスワードポリシーの設定に関するルール

IAM パスワードポリシーは、AWS アカウントのルートユーザー パスワードまたは IAM ユーザー アクセスキーには適用されません。パスワードの有効期限が切れた場合、IAM ユーザーは AWS Management Console にサインインできなくなりますが、引き続きアクセスキーを使用できます。

パスワードポリシーを作成または変更する場合、パスワードポリシーの設定の多くは、ユーザーが次回パスワードを変更するときに適用されます。ただし、一部の設定はすぐに適用されます。例:

- 最小文字数と文字タイプの要件が変更されると、これらの設定はユーザーが次回パスワードを変更するときに適用されます。既存のパスワードが更新されたパスワードポリシーに従っていない場合でも、ユーザーは既存のパスワードの変更を強制されません。

- パスワードの有効期限を設定した場合、有効期限は直ちに適用されます。例えば、パスワードの有効期限を 90 日に設定したとします。この場合、既存のパスワードが作成されてから 90 日を超える期間が経過しているすべての IAM ユーザーのパスワードが失効します。これらのユーザーは、次回サインインするときにパスワードを変更する必要があります。

サインインが指定された回数失敗したユーザーをアカウントからロックアウトするために、「ロックアウトポリシー」を作成することはできません。セキュリティを強化するために、強力なパスワードポリシーと Multi-Factor Authentication (MFA) を組み合わせることをお勧めします。MFA の詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

パスワードポリシーを設定するために必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) がアカウントのパスワードポリシーを表示または編集することを許可するには、アクセス許可を設定する必要があります。ポリシーには、IAM ポリシーに次のパスワードポリシーアクションを含めることができます。

- `iam:GetAccountPasswordPolicy` – エンティティが各自のアカウントのパスワードポリシーを表示することを許可します
- `iam>DeleteAccountPasswordPolicy` – エンティティが各自のアカウントのカスタムパスワードポリシーを削除し、デフォルトのパスワードポリシーに戻すことを許可します
- `iam:UpdateAccountPasswordPolicy` – エンティティが各自のアカウントのカスタムパスワードポリシーを作成または変更することを許可します

次のポリシーは、アカウントのパスワードポリシーを表示および編集するためのフルアクセスを許可します。この例の JSON ポリシードキュメントを使用して IAM ポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessPasswordPolicy",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam>DeleteAccountPasswordPolicy",
        "iam:UpdateAccountPasswordPolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

```
    }  
  ]  
}
```

IAM ユーザーが各自のパスワードを変更するために必要なアクセス許可の詳細については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」をご参照ください。

## デフォルトのパスワードポリシー

管理者がカスタムパスワードポリシーを設定しない場合、IAM ユーザーパスワードはデフォルトの AWS パスワードポリシーの要件を満たす必要があります。

デフォルトのパスワードポリシーでは、次の条件が適用されます。

- パスワードの文字数制限: 8 ~ 128 文字
- 大文字、小文字、数字、英数字以外の文字 (! @ # \$ % ^ & \* ( ) \_ + - = [ ] { } | ' ) のうち、最低 3 つの文字タイプの組み合わせ
- AWS アカウント 名または E メールアドレスと同じでないこと
- 有効期限のないパスワード

## カスタムパスワードポリシーのオプション

アカウントのカスタムパスワードポリシーを設定する場合、次の条件を指定できます。

- パスワードの最小文字数 – 6 ~ 128 文字の間で指定できます。
- パスワードの強度 – 次のいずれかのチェックボックスをオンにして、IAM ユーザーパスワードの強度を定義できます。
  - ラテンアルファベットの大文字 (A-Z) が少なくとも 1 つ必要
  - ラテンアルファベットの小文字 (a-z) が少なくとも 1 つ必要
  - 少なくとも 1 つの数字が必要
  - 少なくとも 1 つの英数字以外の文字 ! @ # \$ % ^ & \* ( ) \_ + - = [ ] { } | ' が必要
- パスワードの有効期限をオンにする – IAM ユーザーのパスワードが設定されてから有効な期間を 1 ~ 1,095 日の間で選択して指定できます。例えば、90 日間の有効期限を指定すると、すぐにすべてのユーザーに影響します。変更後、90 日を超える古いパスワードのユーザーがコンソールにログインする場合は、新しいパスワードを設定する必要があります。75 日から 89 日が経過したパスワードのユーザーは、パスワードの有効期限について AWS Management Console から警告を受

け取ります。IAM ユーザーは、アクセス許可があれば、いつでもパスワードを変更できます。新しいパスワードを設定すると、そのパスワードの有効期間が始まります。IAM ユーザーは一度に 1 つだけ有効なパスワードを持つことができます。

- [Password expiration requires administrator reset] (パスワードの失効に管理者のリセットを必要とする) – パスワードが失効した後に IAM ユーザーが AWS Management Console を使用して各自のパスワードを更新できないようにするには、このオプションを選択します。このオプションを選択する前に、AWS アカウントで複数のユーザーが IAM ユーザーパスワードをリセットするための管理アクセス許可を持っていることを確認します。管理者ユーザーの `iam:UpdateLoginProfile` 権限は IAM ユーザーパスワードをリセットできます。IAM ユーザーの `iam:ChangePassword` 権限キーとアクティブアクセスキーは、IAM ユーザーコンソールのパスワードをプログラムでリセットできます。このチェックボックスをオフにした場合、パスワードが失効している IAM ユーザーは、AWS Management Console にアクセスする前に新しいパスワードを設定する必要があります。
- [Allow users to change their own password] (ユーザーが各自のパスワードを変更することを許可) – お客様のアカウント内のすべての IAM ユーザーが各自のパスワードを変更することを許可できます。これにより、そのユーザーだけに対する `iam:ChangePassword` アクションと `iam:GetAccountPasswordPolicy` アクションへのアクセスがユーザーに付与されます。このオプションでは、各ユーザーにアクセス許可ポリシーがアタッチされません。あるいは、IAM ではすべてのユーザーにアカウントレベルでアクセス許可が適用されます。あるいは、一部のユーザーのみが、自分自身のパスワードを管理できるように許可できます。これを行うには、このチェックボックスをオフにします。パスワードを管理できるユーザーを限定するポリシーの使用方法については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」を参照してください。
- パスワードの再利用を禁止 – 指定した数の以前のパスワードを IAM ユーザーが再利用することを禁止できます。再利用できない以前のパスワードの数を 1~24 個の間で指定できます。

## パスワードポリシーの設定 (コンソール)

AWS Management Console を使用して、カスタムパスワードポリシーを作成、変更、および削除できます。

### カスタムパスワードポリシーを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [アカウント設定] を選択します。

3. [Password policy] (パスワードポリシー) セクションで、[Edit] (編集) を選択します。
4. カスタムパスワードポリシーを使用するには、[Custom] (カスタム) を選択します。
5. パスワードポリシーに適用するオプションを選択し、[Save Changes] (変更の保存) を選択します。
6. [Set custom] (カスタムを設定) を選択して、カスタムパスワードポリシーを設定することを確認します。

#### カスタムパスワードポリシーを変更するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [アカウント設定] を選択します。
3. [Password policy] (パスワードポリシー) セクションで、[Edit] (編集) を選択します。
4. パスワードポリシーに適用するオプションを選択し、[Save Changes] (変更の保存) を選択します。
5. [Set custom] (カスタムを設定) を選択して、カスタムパスワードポリシーを設定することを確認します。

#### カスタムパスワードポリシーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [アカウント設定] を選択します。
3. [Password policy] (パスワードポリシー) セクションで、[Edit] (編集) を選択します。
4. [IAM default] (IAM のデフォルト) を選択してカスタムパスワードポリシーを削除し、[Save changes] (変更を保存) を選択します。
5. [Set default] (デフォルトを設定) を選択して、IAM デフォルトパスワードポリシーを設定することを確認します。

#### パスワードポリシーの設定 (AWS CLI)

AWS Command Line Interface を使用して、パスワードポリシーを設定できます。

AWS CLI からアカウントのカスタムパスワードポリシーを管理するには

以下のコマンドを実行します。

- カスタムパスワードポリシーを作成または変更するには: [aws iam update-account-password-policy](#)
- パスワードポリシーを表示するには: [aws iam get-account-password-policy](#)
- カスタムパスワードポリシーを削除するには: [aws iam delete-account-password-policy](#)

## パスワードポリシーの設定 (AWS API)

AWS API オペレーションを使用して、パスワードポリシーを設定できます。

AWS API からアカウントのカスタムパスワードポリシーを管理するには

以下のオペレーションを呼び出します。

- カスタムパスワードポリシーを作成または変更するには: [UpdateAccountPasswordPolicy](#)
- パスワードポリシーを表示するには: [GetAccountPasswordPolicy](#)
- カスタムパスワードポリシーを削除するには: [DeleteAccountPasswordPolicy](#)

## IAM ユーザーのパスワードの管理

AWS Management Console を使用して AWS リソースを操作する IAM ユーザーには、サインインのためのパスワードが必要です。AWS アカウントの IAM ユーザーのパスワードを作成、変更、削除できます。

ユーザーにパスワードを割り当てると、ユーザーは、以下のような、アカウント用のサインイン URL を使用して AWS Management Console にサインインできます。

```
https://12-digit-AWS-account-ID or alias.signin.aws.amazon.com/console
```

IAM ユーザーの AWS Management Console へのサインインの詳細については、AWS サインインユーザーガイドの「[AWS にサインインする方法](#)」を参照してください。

ユーザーは、パスワードが割り当てられている場合でも、AWS リソースへのアクセスにはやはりアクセス許可が必要です。デフォルトでは、ユーザーには何も権限が与えられていません。ユーザーに必要な権限を与えるには、ユーザーまたはユーザーが属しているグループにポリシーを割り当てます。ユーザーおよびグループの作成の詳細については、[IAM ID \(ユーザー、ユーザーグループ、ロー](#)

ル)を参照してください。ポリシーを使用するアクセス許可設定の詳細については、[IAM ユーザーのアクセス許可の変更](#)を参照してください。

ユーザーに自分のパスワードを変更する権限を付与できます。詳細については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」を参照してください。ユーザーがアカウントのサインインページにアクセスする方法の詳細については、AWS サインイン ユーザーガイドの「[AWS へサインインする方法](#)」を参照してください。

## トピック

- [IAM ユーザーパスワードの作成、変更、削除 \(コンソール\)](#)
- [IAM ユーザーパスワードの作成、変更、削除 \(AWS CLI\)](#)
- [IAM ユーザーパスワードの作成、変更、削除 \(AWS API\)](#)

## IAM ユーザーパスワードの作成、変更、削除 (コンソール)

AWS Management Console を使用して IAM ユーザーのパスワードを管理できます。

ユーザーが組織を離れる際、または今後 AWS へのアクセスが不要な場合には、使用されていた認証情報を検索し、それらが無効になっていることを確認する必要があります。必要のなくなった認証情報は削除することが理想的です。それらは必要に応じて、後日いつでも再作成できます。少なくとも、認証情報を変更して以前のユーザーがアクセスできないようにする必要があります。

## IAM ユーザーのパスワードを追加するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. パスワードを作成するユーザーの名前を選択します。
4. [セキュリティ認証情報] タブを選択し、[コンソールサインイン] で [コンソールアクセスを有効にする] を選択します。
5. [コンソールアクセスを有効にする] の [コンソールパスワード] で、IAM によってパスワードを自動的に生成するか、カスタムパスワードを作成するかを選択します。
  - IAM によって自動的にパスワードを生成するには、[Autogenerated password (自動生成パスワード)] を選択します。
  - カスタムパスワードを作成するには、[Custom password (カスタムパスワード)] を選択し、パスワードを入力します。



**Note**

作成するパスワードは、アカウントの[パスワードポリシー](#)の要件を満たす必要があります。

- サインイン時に新しいパスワードの作成を要求する場合は、[ユーザーは次回のサインインで新しいパスワードを作成する必要があります]を選択します。次に、[コンソールアクセスを有効にする]を選択します。

**Important**

[ユーザーは次回のサインインで新しいパスワードを作成する必要があります] オプションを選択した場合は、ユーザーにパスワードを変更するアクセス許可を持っていることを確認します。詳細については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」を参照してください。

- パスワードを表示してユーザーと共有するには、[コンソールパスワード] ダイアログボックスで [表示] を選択します。


**Important**

セキュリティ上の理由で、この手順を完了するとパスワードにアクセスできなくなりますが、いつでも新しいパスワードを作成できます。

## IAM ユーザーのパスワードを変更するには (コンソール)


- AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
- ナビゲーションペインで [Users (ユーザー)] を選択します。
- パスワードを変更するユーザーの名前を選択します。
- [セキュリティ認証情報] タブを選択し、[コンソールサインイン] で [コンソールアクセスの管理] を選択します。
- [コンソールアクセスを管理] で、[パスワードをリセット] を選択します (まだ選択されていない場合)。コンソールアクセスが無効になっている場合、パスワードは不要です。

6. [コンソールを通じたアクセス] で、IAM によってパスワードを自動的に生成するか、カスタムパスワードを作成するかを選択します。
  - IAM によって自動的にパスワードを生成するには、[Autogenerated password (自動生成パスワード)] を選択します。
  - カスタムパスワードを作成するには、[Custom password (カスタムパスワード)] を選択し、パスワードを入力します。

 Note

作成するパスワードは、アカウントの[パスワードポリシー](#)が現在設定されていれば、そのポリシーの要件を満たす必要があります。

7. サインイン時に新しいパスワードの作成を要求する場合は、[ユーザーは次回のサインインで新しいパスワードを作成する必要があります] を選択します。

 Important

[ユーザーは次回のサインインで新しいパスワードを作成する必要があります] オプションを選択した場合は、ユーザーにパスワードを変更するアクセス許可を持っていることを確認します。詳細については、「[IAM ユーザーに自分のパスワードを変更する権限を付与する](#)」を参照してください。

8. ユーザーのアクティブなコンソールセッションを取り消すには、[アクティブなコンソールセッションを取り消す] を選択します。次に、適用を選択します。

ユーザーのアクティブなコンソールセッションを取り消すと、IAM によって、すべてのアクションに対するすべてのアクセス許可を拒否する新しいインラインポリシーがユーザーにアタッチされます。このポリシーに含まれる条件によって制限が適用されるのは、アクセス許可を取り消した時点より前にセッションが作成された場合と、その時点から約 30 秒後までにセッションが作成された場合のみです。アクセス許可が取り消された後にユーザーが新しいセッションを作成した場合、拒否ポリシーはそのユーザーに適用されません。ユーザーがこの方法を使用して自分のアクティブなコンソールセッションを取り消した場合、ユーザーは即時に AWS Management Console からサインアウトされます。

**⚠ Important**

ユーザーのアクティブなコンソールセッションを正常に取り消すには、そのユーザーに対して PutUserPolicy アクセス許可を持っている必要があります。これにより、AWSRevokeOlderSessions インラインポリシーをユーザーにアタッチできます。

9. パスワードを表示してユーザーと共有するには、[コンソールパスワード] ダイアログボックスで [表示] を選択します。

**⚠ Important**

セキュリティ上の理由で、この手順を完了するとパスワードにアクセスできなくなりますが、いつでも新しいパスワードを作成できます。

IAM ユーザーのパスワードを削除 (無効化) するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. パスワードを削除するユーザーの名前を選択します。
4. [セキュリティ認証情報] タブを選択し、[コンソールサインイン] で [コンソールアクセスの管理] を選択します。
5. [コンソールアクセスを管理] で [コンソールアクセスを無効にする] を選択します (まだ選択していない場合)。コンソールアクセスが無効になっている場合、パスワードは不要です。
6. ユーザーのアクティブなコンソールセッションを取り消すには、[アクティブなコンソールセッションを取り消す] を選択します。次に [アクセスの無効化] を選択します。

**⚠ Important**

ユーザーのアクティブなコンソールセッションを正常に取り消すには、そのユーザーに対して PutUserPolicy アクセス許可を持っている必要があります。これにより、AWSRevokeOlderSessions インラインポリシーをユーザーにアタッチできます。

ユーザーのアクティブなコンソールセッションを取り消すと、IAM によって、すべてのアクションに対するすべてのアクセス許可を拒否する新しいインラインポリシーが IAM ユーザーに埋め込まれます。このポリシーに含まれる条件によって制限が適用されるのは、アクセス許可を取り消した時点より前にセッションが作成された場合と、その時点から約 30 秒後までにセッションが作成された場合のみです。アクセス許可が取り消された後にユーザーが新しいセッションを作成した場合、拒否ポリシーはそのユーザーに適用されません。ユーザーがこの方法を使用して自分のアクティブなコンソールセッションを取り消した場合、ユーザーは即時に AWS Management Console からサインアウトされます。

#### Important

パスワードを削除することで、AWS Management Console への IAM ユーザーアクセスを無効にできます。これにより、サインイン認証情報を使用して AWS Management Console にサインインすることができなくなります。権限を変更したり、引き受けたロールを使用してコンソールにアクセスできないようにしたりすることはありません。ユーザーがアクティブなアクセスキーを持っている場合、それらのキーは引き続き機能し、AWS CLI、Tools for Windows PowerShell、AWS API、または AWS Console Mobile Application 用のツールを介したアクセスを許可します。

## IAM ユーザーパスワードの作成、変更、削除 (AWS CLI)

AWS CLI API を使用して IAM ユーザーのパスワードを管理できます。

パスワードを作成するには (AWS CLI)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[aws iam get-login-profile](#) コマンドを実行します。
2. パスワードを作成するには、[aws iam create-login-profile](#) コマンドを実行します。

ユーザーのパスワードを変更するには (AWS CLI)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[aws iam get-login-profile](#) コマンドを実行します。
2. パスワードを変更するには、[aws iam update-login-profile](#) コマンドを実行します。

## ユーザーのパスワードを削除 (無効化) するには (AWS CLI)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[aws iam get-login-profile](#) コマンドを実行します。
2. (オプション) パスワードを前回使用した日付を確認するには、[aws iam get-user](#) コマンドを実行します。
3. パスワードを削除するには、[aws iam delete-login-profile](#) コマンドを実行します。

### ⚠ Important

ユーザーのパスワードを削除すると、ユーザーは AWS Management Console にサインインできなくなります。ユーザーがアクティブなアクセスキーを持っている場合、それらのキーは引き続き有効で、AWS CLI、Tools for Windows PowerShell、または AWS API 関数呼び出しで使用できます。AWS CLI、Tools for Windows PowerShell、または AWS API を使用してユーザーを AWS アカウント から削除する場合、まずこのオペレーションを使用してパスワードを削除する必要があります。詳細については、「[IAM ユーザーの削除 \(AWS CLI\)](#)」を参照してください。

## 指定した時間より前にユーザーのアクティブなコンソールセッションを取り消すには (AWS CLI)

1. 指定した時間より前に IAM ユーザーのアクティブなコンソールセッションを取り消すインラインポリシーを埋め込むには、以下のインラインポリシーを使用して次のコマンドを実行します:  
[aws iam put-user-policy](#)

このインラインポリシーはすべてのアクセス許可を拒否し、[aws:TokenIssueTime](#) 条件キーを含みます。インラインポリシーの Condition 要素で指定された時間より前に、ユーザーのアクティブなコンソールセッションを取り消します。aws:TokenIssueTime 条件キーの値は、独自の値に置き換えてください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {
```

```
    "aws:TokenIssueTime": "2014-05-07T23:47:00Z"  
  }  
}  
}  
}
```

2. (オプション) IAM ユーザーに埋め込まれているインラインポリシーの名前をリストするには、次のコマンドを実行します: [aws iam list-user-policies](#)
3. (オプション) IAM ユーザーに埋め込まれている名前付きのインラインポリシーを表示するには、次のコマンドを実行します: [aws iam get-user-policy](#)

## IAM ユーザーパスワードの作成、変更、削除 (AWS API)

AWS API を使用して IAM ユーザーのパスワードを管理できます。

### パスワードを作成するには (AWS API)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[GetLoginProfile](#) オペレーションを呼び出します。
2. パスワードを作成するには、[CreateLoginProfile](#) オペレーションを呼び出します。

### ユーザーのパスワードを変更するには (AWS API)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[GetLoginProfile](#) オペレーションを呼び出します。
2. パスワードを変更するには、[UpdateLoginProfile](#) オペレーションを呼び出します。

### ユーザーのパスワードを削除 (無効化) するには (AWS API)

1. (オプション) ユーザーがパスワードを持っているかどうかを確認するには、[GetLoginProfile](#) コマンドを実行します。
2. (オプション) パスワードを前回使用した日付を確認するには、[GetUser](#) コマンドを実行します。
3. パスワードを削除するには、[DeleteLoginProfile](#) コマンドを実行します。

**⚠ Important**

ユーザーのパスワードを削除すると、ユーザーは AWS Management Console にサインインできなくなります。ユーザーがアクティブなアクセスキーを持っている場合、それらのキーは引き続き有効で、AWS CLI、Tools for Windows PowerShell、または AWS API 関数呼び出しで使用できます。AWS CLI、Tools for Windows PowerShell、または AWS API を使用してユーザーを AWS アカウント から削除する場合、まずこのオペレーションを使用してパスワードを削除する必要があります。詳細については、「[IAM ユーザーの削除 \(AWS CLI\)](#)」を参照してください。

指定した時間より前にユーザーのアクティブなコンソールセッションを取り消すには (AWS API)

1. 指定した時間より前に IAM ユーザーのアクティブなコンソールセッションを取り消すインラインポリシーを埋め込むには、以下のインラインポリシーを使用して次のコマンドを実行します:

[PutUserPolicy](#)

このインラインポリシーはすべてのアクセス許可を拒否し、[aws:TokenIssueTime](#) 条件キーを含みます。インラインポリシーの Condition 要素で指定された時間より前に、ユーザーのアクティブなコンソールセッションを取り消します。aws:TokenIssueTime 条件キーの値は、独自の値に置き換えてください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {
        "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
      }
    }
  }
}
```

2. (オプション) IAM ユーザーに埋め込まれているインラインポリシーの名前をリストするには、次のコマンドを実行します: [ListUserPolicies](#)
3. (オプション) IAM ユーザーに埋め込まれている名前付きインラインポリシーを表示するには、次のコマンドを実行します: [GetUserPolicy](#)



## IAM ユーザーに自分のパスワードを変更する権限を付与する

### Note

フェデレーション ID を持つユーザーは、ID プロバイダーが定義したプロセスを使用してパスワードを変更します。[ベストプラクティス](#)として、人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには、ID プロバイダーとのフェデレーションの使用を必須とします。

IAM ユーザーに、AWS Management Console にサインインするためのパスワードを変更するアクセス許可を付与できます。これには以下の 2 つの方法があります。

- [アカウントのすべての IAM ユーザーが自分のパスワードを変更できるようにします。](#)
- [選択した IAM ユーザーだけが自分のパスワードを変更できるようにします。](#) このシナリオでは、すべてのユーザーが各自のパスワードを変更するオプションを無効にし、一部のユーザーにのみアクセス許可を付与する IAM ポリシーを使用します。この方法では、ユーザーは各自のパスワードと、必要に応じて各自のアクセスキーなどの他の認証情報を変更できます。

### Important

IAM ユーザーが強力なパスワードを作成することを求める [カスタムパスワードポリシーを設定](#) することをお勧めします。

すべての IAM ユーザーが自分のパスワードを変更できるようにします。

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Account settings] (アカウント設定) をクリックします。
3. [Password policy] (パスワードポリシー) セクションで、[Edit] (編集) を選択します。
4. カスタムパスワードポリシーを使用するには、[Custom] (カスタム) を選択します。
5. [Allow users to change their own password] (ユーザーにパスワードの変更を許可) を選択し、[save changes] (変更の保存) をクリックします。これにより、アカウントのすべてのユーザーに、そのユーザーだけに対する iam:ChangePassword アクションと iam:GetAccountPasswordPolicy アクションへのアクセスが付与されます。

6. パスワードを変更するための次の手順をユーザーに提供します: [IAM ユーザーが自分のパスワードを変更する方法](#)。

アカウントのパスワードポリシー (ユーザーに自分のパスワードの変更を許可する設定を含む) の変更可以使用できる AWS CLI、Tools for Windows PowerShell、および API コマンドの詳細については、「[パスワードポリシーの設定 \(AWS CLI\)](#)」を参照してください。

選択された IAM ユーザーが自分のパスワードを変更可能にするには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Account settings] (アカウント設定) をクリックします。
3. [Password policy] (パスワードポリシー) セクションで、[Allow users to change their own password] (ユーザーにパスワードの変更を許可) が選択されていないことを確認します。このチェックボックスがオンになっている場合、すべてのユーザーが自分のパスワードを変更できます ( 前の手順を参照 ) 。
4. パスワードの変更が許可されている必要があるユーザーを作成します (まだ存在していない場合)。詳細については、「[AWS アカウント での IAM ユーザーの作成](#)」を参照してください。
5. (オプション) 自分のパスワードの変更を許可するユーザーに対して IAM グループを作成し、そのグループに前のステップのユーザーを追加します。詳細については、「[IAM ユーザーグループの管理](#)」を参照してください。
6. 以下のポリシーをグループに割り当てます 詳細については、「[IAM ポリシーを管理する](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

}

このポリシーでは、ユーザーに [ChangePassword \(パスワードの変更\)](#) アクションへのアクセスを許可して、コンソール、AWS CLI、Tools for Windows PowerShell、または API から自分のパスワードのみを変更できるようにします。また、[GetAccountPasswordPolicy](#) アクションへのアクセス権も付与します。これにより、ユーザーは現在のパスワードポリシーを表示できます。このアクセス許可は、ユーザーが [Change password] (パスワードの変更) ページでアカウントパスワードポリシーを表示できるようにするために必要です。変更されたパスワードがポリシーの要件を確実に満たしているようにするために、ユーザーは現在のパスワードポリシーの読み取りが許可されている必要があります。

7. パスワードを変更するための次の手順をユーザーに提供します: [IAM ユーザーが自分のパスワードを変更する方法](#)。

## 詳細情報

認証情報の管理の詳細については、次のトピックを参照してください。

- [IAM ユーザーに自分のパスワードを変更する権限を付与する](#)
- [AWS でのユーザーパスワードの管理](#)
- [IAM ユーザー用のアカウントパスワードポリシーの設定](#)
- [IAM ポリシーを管理する](#)
- [IAM ユーザーが自分のパスワードを変更する方法](#)

## IAM ユーザーが自分のパスワードを変更する方法

IAM ユーザーに自分のパスワードを変更するアクセス許可が付与されている場合、ユーザーは AWS Management Console の特別なページを使用してこれを行うことができます。AWS CLI または AWS API を使用することもできます。

### トピック

- [必要なアクセス許可](#)
- [IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)
- [IAM ユーザー自身によるパスワードの変更方法 \(AWS CLI または AWS API\)](#)

## 必要なアクセス許可

IAM ユーザーのパスワードを変更するには、次のポリシーからのアクセス許可が必要です: [AWS: IAM ユーザーが \[セキュリティ認証情報\] ページで自分のコンソールパスワードを変更できるようにします](#)。

### IAM ユーザー自身によるパスワードの変更方法 (コンソール)

以下の手順では、IAM ユーザーが AWS Management Console を使用して自分のパスワードを変更する方法について説明します。

### IAM ユーザー自身のパスワードを変更するには (コンソール)

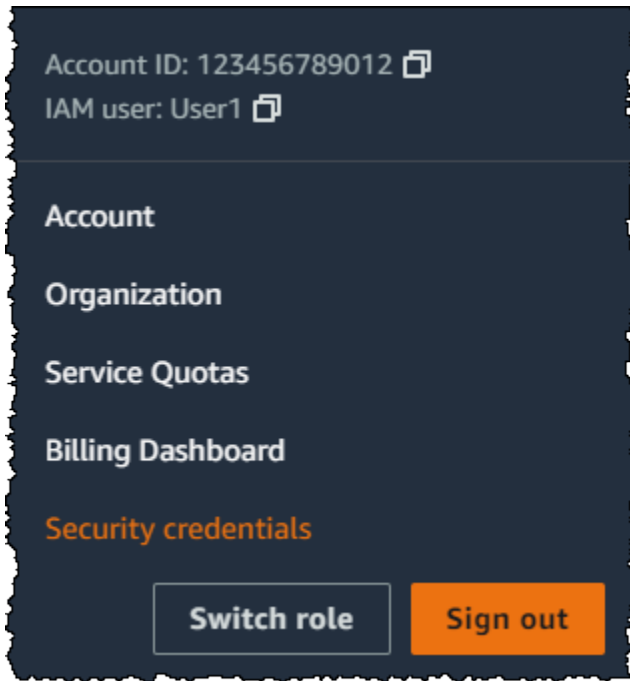
1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

#### Note

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある場合は、ページの下部にある [別のアカウントにサインイン] を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようにすることができます。

AWS アカウント アカウント ID の取得については、管理者にお問い合わせください。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



3. [AWS IAM 認証情報] タブで、[パスワードを変更] を選択します。
4. [Current password (現在のパスワード)] に現在のパスワードを入力します。[New password (新しいパスワード)] ボックスおよび [Confirm new password (新しいパスワードの確認)] ボックスに新しいパスワードを入力します。その後、[パスワードを更新] を選択します。

**Note**

新しいパスワードは、アカウントのパスワードポリシーの要件を満たしている必要があります。詳細については、「[IAM ユーザー用のアカウントパスワードポリシーの設定](#)」を参照してください。

## IAM ユーザー自身によるパスワードの変更方法 (AWS CLI または AWS API)

以下の手順では、IAM ユーザーが AWS CLI または AWS API を使用して自身のパスワードを変更する方法について説明します。

自身の IAM パスワードを変更するには、以下を使用します。

- AWS CLI: [aws iam change-password](#)
- AWS API: [ChangePassword](#)

## IAM ユーザーのアクセスキーの管理

 [Follow us on Twitter](#)

### Important

[ベストプラクティス](#)は、アクセスキーのような長期的認証情報を作成するのではなく、IAM ロールなどの一時的なセキュリティ認証情報を使用することです。アクセスキーを作成する前に、[長期的なアクセスキーの代替案](#)を確認してください。

アクセスキーは、IAM ユーザーまたは AWS アカウントのルートユーザーの長期的な認証情報です。アクセスキーを使用して、AWS CLI または AWS API (直接または AWS SDK を使用) にプログラムでリクエストに署名することができます。詳細については、「[AWS API リクエストの署名](#)」を参照してください。

アクセスキーは、アクセスキー ID (例: AKIAIOSFODNN7EXAMPLE) とシークレットアクセスキー (例: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY) の 2 つの部分で構成されています。リクエストを認証するために、アクセスキー ID とシークレットアクセスキーの両方を使用する必要があります。

アクセスキーペアを作成する場合は、アクセスキー ID とシークレットアクセスキーを安全な場所に保存します。このシークレットアクセスキーは、作成時にのみ使用できます。シークレットアクセスキーを紛失した場合は、そのアクセスキーを削除し、新しく作成する必要があります。詳細については、「[紛失したり忘れたりしたパスワードまたは AWS のアクセスキーのリセット](#)」を参照してください。

アクセスキーはユーザーごとに最大 2 つまで持つことができます。

### Important

アクセスキーを安全に管理します。[アカウント識別子を確認する](#)ためであっても、アクセスキーを不正な当事者に提供しないでください。提供すると、第三者がアカウントへの永続的なアクセスを取得する場合があります。

以下のトピックでは、アクセスキーに関連する管理タスクについて詳しく説明します。

トピック

- [アクセスキーを管理するために必要なアクセス許可](#)
- [アクセスキーの管理 \(コンソール\)](#)
- [アクセスキーの管理 \(AWS CLI\)](#)
- [アクセスキーの管理 \(AWS API\)](#)
- [アクセスキーの更新](#)
- [アクセスキーを保護する](#)
- [アクセスキーの監査](#)

## アクセスキーを管理するために必要なアクセス許可

### Note

iam:TagUser は、アクセスキーの説明を追加および編集するためのオプションのアクセス許可です。詳細については、「[IAM ユーザーのタグ付け](#)」を参照してください。

IAM ユーザーのアクセスキーを作成するには、次のポリシーのアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:GetUser",
        "iam:ListAccessKeys",
        "iam:TagUser"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

IAM ユーザーのアクセスキーを更新するには、次のポリシーのアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
```



```
"Statement": [
  {
    "Sid": "ManageOwnAccessKeys",
    "Effect": "Allow",
    "Action": [
      "iam:CreateAccessKey",
      "iam>DeleteAccessKey",
      "iam:GetAccessKeyLastUsed",
      "iam:GetUser",
      "iam:ListAccessKeys",
      "iam:UpdateAccessKey",
      "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  }
]
```

## アクセスキーの管理 (コンソール)

AWS Management Console を使用して IAM ユーザーのアクセスキーを管理することができます。

ユーザー自身のアクセスキーを作成、変更、または削除するには (コンソール)

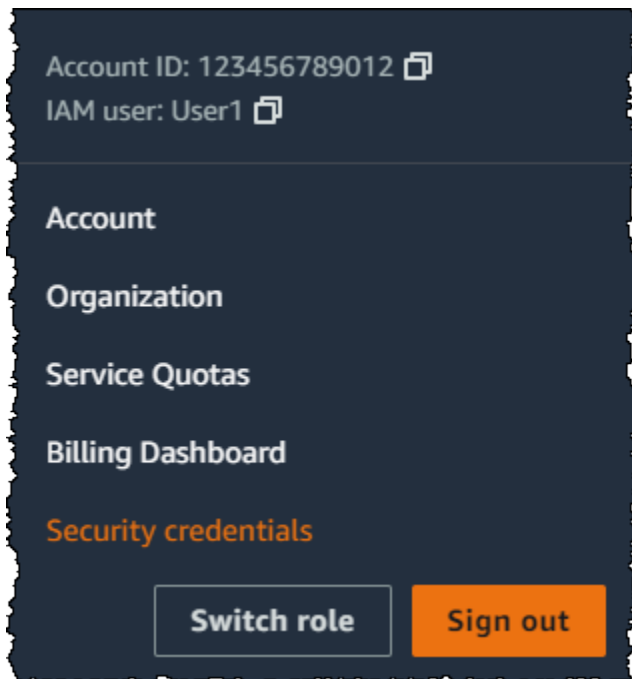
1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

### Note

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある場合は、ページの下部にある [別のアカウントにサインイン] を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようにすることができます。

AWS アカウント アカウント ID の取得については、管理者にお問い合わせください。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



次のいずれかを行います。

アクセスキーを作成するには

1. [Access keys (アクセスキー)] セクションで、[Create access key (アクセスキーを作成)] を選択します。既に 2 つのアクセスキーがある場合、このボタンは無効になるため、新しいアクセスキーを作成する前に、前のアクセスキーを削除する必要があります。
2. [Access key best practices & alternatives] (アクセスキーのベストプラクティスと代替方法) ページで、対象のユースケースを選択し、長期的なアクセスキーの作成を避けるのに役立つその他のオプションを確認します。依然としてユースケースにアクセスキーが必要だと考えられる場合は、[Other] (その他)、[Next] (次へ) の順に選択します。
3. (オプション) アクセスキーの説明タグに値を設定します。これにより、タグのキーと値のペアが IAM ユーザーに対し追加されます。後で、アクセスキーを識別し、更新する際にこの設定が役立ちます。タグキーには、アクセスキー ID が設定されます。タグ値には、入力したアクセスキーの説明が設定されます。完了したら、[Create access key] (アクセスキーを作成) を選択します。
4. [Retrieve access keys] (アクセスキーの取得) ページで、[Show] (表示) を選択してユーザーのシークレットアクセスキーの値を表示するか、[Download .csv file] (.csv ファイルをダウンロード) を選択します。これはシークレットアクセスキーを保存する唯一の機会です。シークレットアクセスキーを安全な場所に保存したら、[Done] (完了) を選択します。

## アクセスキーを無効にするには

- [Access keys] アクセスキーセクションで、無効にするキーを選択し、[Actions] (アクション)、[Deactivate] (無効化) の順に選択します。確認を求めるメッセージが表示されたら、[Deactivate (無効化)] を選択します。非アクティブ化されたアクセスキーは、2 つのアクセスキーの制限にカウントされます。

## アクセスキーを有効にするには

- [Access keys] (アクセスキー) セクションで、有効にするキーを選択し、[Actions] (アクション)、[Activate] (有効化) の順に選択します。

## 不要になったアクセスキーを削除するには

- [Access keys] (アクセスキー) セクションで、削除するキーを選択し、[Actions] (アクション)、[Delete] (削除) の順に選択します。ダイアログの指示に従って、まず [Deactivate] (無効化) をしてから、削除することを確認します。アクセスキーを完全に削除する前に、アクセスキーが使用されていないことを確認することをお勧めします。

## 別の IAM ユーザーのアクセスキーを作成、変更、または削除するには (コンソール)


1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. アクセスキーを管理するユーザー名を選択し、[Security credentials] タブを選択します。
4. [アクセスキー] セクションで、以下のいずれかを実行します。
  - アクセスキーを作成するには、[アクセスキーの作成] を選択します。このボタンが無効化されている場合は、新しいキーを作成する前に、既存のキーのいずれかを削除する必要があります。[Access key best practices & alternatives] (アクセスキーのベストプラクティスと代替案) ページで、ベストプラクティスと代替案を確認します。ユースケースを選択して、長期的なアクセスキーの作成を避けるのに役立つ他のオプションを確認します。依然としてユースケースにアクセスキーが必要だと考えられる場合は、[Other] (その他)、[Next] (次へ) の順に選択します。[Retrieve access key] (アクセスキーの取得) ページで、[Show] (表示) を選択し、ユーザーのシークレットアクセスキーの値を表示します。アクセスキー ID とシークレットアクセスキーを、ご使用のコンピュータの安全な場所に .csv ファイルとして保存するには、[Download .csv file] (.csv ファイルをダウンロード) を選択します。ユーザー用のアクセス

キー作成後、キーペアはデフォルトで有効状態になっているので、対象のユーザーはすぐにキーペアを使用できます。

- 有効状態のアクセスキーを無効にするには、[Actions] (アクション) を選択し、次に [Deactivate] (無効化) を選択します。
- 無効状態のアクセスキーを有効にするには、[Actions] (アクション) を選択し、次に [Activate] (有効化) を選択します。
- アクセスキーを削除するには、[Actions] (アクション) を選択し、次に [Delete] (削除) を選択します。まず、ダイアログの指示に従って [Deactivate] (無効化) を行い、次に削除することを確認します。AWS は、この処理を実行する前にキーを無効化して、そのキーが使用されていないことをテストすることを推奨しています。AWS Management Console を使用する場合は、キーを削除する前に非アクティブ化する必要があります。


IAM ユーザーのアクセスキーを一覧表示するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 対象のユーザー名を選択し、[セキュリティ認証情報] タブを選択します。[Access keys] (アクセスキー) セクションに、ユーザーのアクセスキーと各キーのステータスが表示されます。

 Note

ユーザーのアクセスキー ID のみが表示されます。シークレットアクセスキーは、キー作成時にのみ取得できます。

複数の IAM ユーザーのアクセスキー ID を表示するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 必要に応じて、以下の手順を行い、[アクセスキー ID] 列をユーザーテーブルに追加します。
  - a. 右端のテーブルの上で、設定アイコン ( ) を選択します。


- b. [Manage Columns (列の管理)] で、[アクセスキー ID] を選択します。
  - c. [Close (閉じる)] を選択して、ユーザーのリストに戻ります。
4. [アクセスキー ID] 列には、各アクセスキー ID とそれに続く状態が表示されます ([23478207027842073230762374023 (有効)], [22093740239670237024843420327 または (無効)] など)。

この情報を使用して、1 つ以上のアクセスキーを持つユーザーのアクセスキーを表示およびコピーできます。アクセスキーのないユーザーは、この列に [None (なし)] と表示されます。

#### Note

ユーザーのアクセスキー ID およびステータスのみが表示されます。シークレットアクセスキーは、キー作成時にのみ取得できます。

特定のアクセスキーを持つ IAM ユーザーを見つけるには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 検索ボックスで、検索するユーザーのアクセスキー ID を入力するか、貼り付けます。
4. 必要に応じて、以下の手順を行い、[アクセスキー ID] 列をユーザーテーブルに追加します。
  - a. 右端のテーブルの上で、設定アイコン ( ) を選択します。
  - b. [Manage Columns (列の管理)] で、[アクセスキー ID] を選択します。
  - c. [Close (閉じる)] を選択して、ユーザーのリストに戻り、フィルター処理されたユーザーが固有のアクセスキーを持つことを確認します。

## アクセスキーの管理 (AWS CLI)

AWS CLI から IAM ユーザーのアクセスキーを管理するには、以下のコマンドを実行します。

- アクセスキーを作成するには: [aws iam create-access-key](#)
- アクセスキーを有効化または無効化するには: [aws iam update-access-key](#)

- ユーザーアクセスキーを一覧表示するには: [aws iam list-access-keys](#)
- アクセスキーの最終使用日時を確認するには: [aws iam get-access-key-last-used](#)
- アクセスキーを削除するには: [aws iam delete-access-key](#)

## アクセスキーの管理 (AWS API)

AWS API から IAM ユーザーのアクセスキーを管理するには、以下のオペレーションを呼び出します。

- アクセスキーを作成するには: [CreateAccessKey](#)
- アクセスキーを有効化または無効化するには: [UpdateAccessKey](#)
- ユーザーアクセスキーを一覧表示するには: [ListAccessKeys](#)
- アクセスキーの最終使用日時を確認するには: [GetAccessKeyLastUsed](#)
- アクセスキーを削除するには: [DeleteAccessKey](#)

## アクセスキーの更新

セキュリティの**ベストプラクティス**として、IAM ユーザーのアクセスキーは、従業員が退職するときなど、必要に応じて更新することをお勧めします。IAM ユーザーは、必要な権限が付与されている場合、自分のアクセスキーを更新できます。

自身のアクセスキーを更新するために管理者からユーザーに IAM ユーザーアクセス許可を付与する方法については、「[AWS: IAM ユーザーが \[セキュリティ認証情報\] ページで自分のパスワード、アクセスキー、および SSH パブリックキーを管理できるようにします](#)」を参照してください。また、アカウントにパスワードポリシーを適用して、すべての IAM ユーザーにパスワードの更新を要求し、その頻度を定めることもできます。詳細については、「[IAM ユーザー用のアカウントパスワードポリシーの設定](#)」を参照してください。

### トピック

- [IAM ユーザーアクセスキーの更新 \(コンソール\)](#)
- [アクセスキーの更新 \(AWS CLI\)](#)
- [アクセスキーの更新 \(AWS API\)](#)

## IAM ユーザーアクセスキーの更新 (コンソール)

AWS Management Console からアクセスキーを更新できます。

## IAM ユーザーのアプリケーションを中断せずにアクセスキーを更新するには (コンソール)

1. 最初のアクセスキーがアクティブな間に、2 番目のアクセスキーを作成します。
  - a. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
  - b. ナビゲーションペインで [Users (ユーザー)] を選択します。
  - c. 対象のユーザー名を選択し、[セキュリティ認証情報] タブを選択します。
  - d. [Access keys (アクセスキー)] セクションで、[Create access key (アクセスキーを作成)] を選択します。[Access key best practices & alternatives] (アクセスキーのベストプラクティスと代替案) ページで、[Other] (その他)、[Next] (次へ) の順に選択します。
  - e. (オプション) アクセスキーの説明タグに値を設定して、この IAM ユーザーにタグキーと値のペアを追加します。後で、アクセスキーを識別し、更新する際にこの設定が役立ちます。タグキーには、アクセスキー ID が設定されます。タグ値には、入力したアクセスキーの説明が設定されます。完了したら、[Create access key] (アクセスキーを作成) を選択します。
  - f. [Retrieve access keys] (アクセスキーの取得) ページで、[Show] (表示) を選択してユーザーのシークレットアクセスキーの値を表示するか、[Download .csv file] (.csv ファイルをダウンロード) を選択します。これはシークレットアクセスキーを保存する唯一の機会です。シークレットアクセスキーを安全な場所に保存したら、[Done] (完了) を選択します。


ユーザー用のアクセスキー作成後、キーペアはデフォルトで有効状態になっているので、対象のユーザーはすぐにキーペアを使用できます。この時点で、ユーザーには 2 つのアクティブなアクセスキーがあります。

2. すべてのアプリケーションとツールを更新して新しいアクセスキーを使用します。
3. 最も古いアクセスキーの [Last used] (前回使用) を確認して、最初のアクセスキーが使用中かどうかを確認します。先に進む前に、数日間待ってから古いアクセスキーが使用されているかどうかを確認するという方法があります。
4. [Last used] (前回使用) の情報で、古いキーが使用された形跡がないことを示していても、最初のアクセスキーをすぐには削除しないことをお勧めします。代わりに、[Actions] (アクション)、[Deactivate] (無効化) の順に選択し、最初のアクセスキーを無効化します。
5. 新しいアクセスキーのみを使用して、アプリケーションが機能しているかどうかを確認してください。この時点で、元のアクセスキーをまだ使用しているツールやアプリケーションは AWS リソースへアクセスできなくなるので、機能が停止されます。そのようなアプリケーションやツールを見つけた場合は、最初のアクセスキーを再度有効にすることができます。次に、「[Step 3](#)」に戻り、新しいキーを使用するためにこのアプリケーションを更新します。



6. 一定期間待機して、すべてのアプリケーションとツールが更新されていることを確認した後、最初のアクセスキーを削除できます。
  - a. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
  - b. ナビゲーションペインで [Users (ユーザー)] を選択します。
  - c. 対象のユーザー名を選択し、[セキュリティ認証情報] タブを選択します。
  - d. 削除するアクセスキーの [Access keys] (アクセスキー) セクションで、[Actions] (アクション) を選択し、次に [Delete] (削除) を選択します。ダイアログの指示に従って、まず [Deactivate] (無効化) を行ってから、削除することを確認します。

どのアクセスキーを更新または削除する必要があるかを判断するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 必要に応じて、以下の手順を実行して [Access key age (アクセスキーの古さ)] 列をユーザーテーブルに追加します。
  - a. 右端のテーブルの上で、設定アイコン (  ) を選択します。
  - b. [Manage Columns (列の管理)] で、[Access key age (アクセスキーの古さ)] を選択します。
  - c. [Close (閉じる)] を選択して、ユーザーのリストに戻ります。
4. [Access key age (アクセスキーの古さ)] 列には、最も古いアクティブアクセスキーが作成されてから経過した日数が表示されます。この情報を使用して、更新または削除の必要があるアクセスキーを持つユーザーを検索できます。アクセスキーのないユーザーは、この列に [None (なし)] と表示されます。

## アクセスキーの更新 (AWS CLI)

AWS Command Line Interface からアクセスキーを更新できます。

## アプリケーションを中断せずにアクセスキーを更新するには (AWS CLI)

1. 最初のアクセスキーがアクティブな間に、2 番目のアクセスキーを作成します。このキーは、デフォルトでアクティブになります。次のコマンドを実行します。

- [aws iam create-access-key](#)

この時点で、ユーザーには 2 つのアクティブなアクセスキーがあります。

2. すべてのアプリケーションとツールを更新して新しいアクセスキーを使用します。
3. 次のコマンドを使用して最初のアクセスキーがまだ使用されているかどうかを確認します。

- [aws iam get-access-key-last-used](#)

先に進む前に、数日間待ってから古いアクセスキーが使用されているかどうかを確認するという方法があります。

4. ステップ [Step 3](#) で古いキーが使用されていないことがわかっていても、最初のアクセスキーはすぐに削除しないことをお勧めします。代わりに、次のコマンドを使用して最初のアクセスキーの状態を Inactive に変更します。

- [aws iam update-access-key](#)

5. 新しいアクセスキーのみを使用して、アプリケーションが機能しているかどうかを確認してください。この時点で、元のアクセスキーをまだ使用しているツールやアプリケーションは AWS リソースへアクセスできなくなるので、機能が停止されます。このようなアプリケーションまたはツールは、その状態を Active に戻すことで、最初のアクセスキーを再度有効にすることができます。次にステップ [Step 2](#) に戻り、新しいキーを使用するようにこのアプリケーションを更新します。

6. 一定期間待機して、すべてのアプリケーションとツールが更新されたことを確認したら、次のコマンドを使用して最初のアクセスキーを削除できます。

- [aws iam delete-access-key](#)

## アクセスキーの更新 (AWS API)

AWS API を使用してアクセスキーを更新できます。

## アプリケーションを中断せずにアクセスキーを更新するには (AWS API)

1. 最初のアクセスキーがアクティブな間に、2 番目のアクセスキーを作成します。このキーは、デフォルトでアクティブになります。次のオペレーションを呼び出します。

- [CreateAccessKey](#)

この時点で、ユーザーには 2 つのアクティブなアクセスキーがあります。

2. すべてのアプリケーションとツールを更新して新しいアクセスキーを使用します。
3. 次のオペレーションを呼び出して最初のアクセスキーがまだ使用されているかどうかを確認します。

- [GetAccessKeyLastUsed](#)

先に進む前に、数日間待ってから古いアクセスキーが使用されているかどうかを確認するという方法があります。

4. ステップ [Step 3](#) で古いキーが使用されていないことがわかっていても、最初のアクセスキーはすぐに削除しないことをお勧めします。代わりに、次のオペレーションを呼び出して最初のアクセスキーの状態を Inactive に変更します。

- [UpdateAccessKey](#)

5. 新しいアクセスキーのみを使用して、アプリケーションが機能しているかどうかを確認してください。この時点で、元のアクセスキーをまだ使用しているツールやアプリケーションは AWS リソースへアクセスできなくなるので、機能が停止されます。このようなアプリケーションまたはツールは、その状態を Active に戻すことで、最初のアクセスキーを再度有効にすることができます。次にステップ [Step 2](#) に戻り、新しいキーを使用するようにこのアプリケーションを更新します。
6. 一定期間待機して、すべてのアプリケーションとツールが更新されたことを確認したら、次のオペレーションを呼び出して最初のアクセスキーを削除できます。

- [DeleteAccessKey](#)

## アクセスキーを保護する

アクセスキーを持っているユーザーなら誰でも、AWS リソースに対して同じレベルのアクセス権を持ちます。したがって、AWS でのアクセスキーの保護は非常に困難で、しかも [責任共有モデル](#) に沿って行う必要があります。

アクセスキーを保護するのに役立つガイダンスについては、以下のセクションを参照してください。

**Note**

組織によっては、セキュリティ要件とポリシーがこのトピックに記載されているものとは異なる可能性があります。ここで示すのは、一般的なガイドラインとしての提案です。

## AWS アカウントのルートユーザー アクセスキーを削除する (または生成しない)

アカウントを保護する最善の方法の 1 つは、AWS アカウントのルートユーザー のアクセスキーを持たないことです。ルートユーザーのアクセスキーを持つ必要がある場合 (まれに) を除いて、キーを生成しないことをお勧めします。代わりに、日常の管理タスクのために AWS IAM Identity Center で管理ユーザーを作成してください。IAM Identity Center で管理ユーザーを作成する方法については、IAM Identity Center ユーザーガイドの「[はじめに](#)」を参照してください。

アカウントのルートユーザーアクセスキーが既にある場合は、そのキーを現在使用しているアプリケーション (ある場合) の場所を見つけ、ルートユーザーアクセスキーを IAM ユーザーアクセスキーに置き換えることをお勧めします。次いで、ルートユーザーアクセスキーを無効にして削除します。アクセスキーを更新する方法の詳細については、「[アクセスキーの更新](#)」を参照してください。

## 長期アクセスキーの代わりに一時的なセキュリティ認証情報 (IAM ロール) を使用する

多くの場合、期限のない長期のアクセスキー (IAM ユーザーのアクセスキーなど) は必要ありません。その代わりに、IAM ロールを作成し、一時的なセキュリティ認証情報を生成できます。一時的なセキュリティ認証情報は、アクセスキー ID とシークレットアクセスキーで構成されていますが、認証情報がいつ無効になるかを示すセキュリティトークンも含んでいます。

IAM ユーザーやルートユーザーに関連付けられているものなどの長期的なアクセスキーは、それらを手動で取り消すまで有効です。ただし、IAM ロールや AWS Security Token Service の他の機能を使用して取得した一時的なセキュリティ認証情報は、短期間で期限が切れます。認証情報が誤って開示された場合のリスクに備えて、一時的なセキュリティ認証情報を使用することができます。

以下のシナリオでは、IAM ロールと一時的なセキュリティ認証情報を使用します。

- Amazon EC2 インスタンスで実行されているアプリケーションまたは AWS CLI スクリプトがある場合。アプリケーション内で直接アクセスキーを使用しないでください。アクセスキーをアプリケーションに渡したり、アプリケーションに埋め込んだり、ソースからアクセスキーを読み取ったりしないでください。代わりに、アプリケーションに適したアクセス許可を持つ IAM ロールを定義し、[EC2 のロール](#)を使用して Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起

動します。これにより、IAM ロールが Amazon EC2 インスタンスに関連付けられます。この方法により、アプリケーションは AWS へのプログラムされた呼び出しに使用することもできる、一時的なセキュリティ認証情報を取得します。AWS SDK および AWS Command Line Interface (AWS CLI) は、そのロールから一時的なセキュリティ認証情報を自動的に取得できます。

- クロスアカウントアクセス許可を付与する必要がある場合。IAM ロールを使用してアカウント間の信頼を確立し、信頼できるアカウントへ制限されたアクセス許可を 1 つのアカウントのユーザーに付与します。詳細については、「[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)」を参照してください。
- モバイルアプリを持っている場合。暗号化ストレージ内であっても、アクセスキーをアプリに埋め込まないでください。代わりに、[Amazon Cognito](#) を使用して、アプリでユーザー ID を管理してください。このサービスでは、Login with Amazon、Facebook、Google、または OpenID Connect (OIDC) に対応している任意の ID プロバイダを使用してユーザーを認証できます。さらに、Amazon Cognito 認証情報プロバイダを使用して、AWS にリクエストを送信するためにアプリが使用する認証情報を管理できます。
- AWS に連携しようとしており、組織が SAML 2.0 をサポートしている場合。SAML 2.0 をサポートする ID プロバイダを持つ組織で作業する場合は、SAML を使用するようにプロバイダを設定します。SAML を使用して、AWS と認証情報を交換し、一連の一時的なセキュリティ認証情報を取り戻すことができます。詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。
- AWS に連携しようとしており、組織にオンプレミスのアイデンティティストアがある場合。ユーザーが組織内で認証できる場合、AWS リソースにアクセスするための一時的なセキュリティ認証情報を発行できるアプリケーションを記述することができます。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

#### Note

AWS リソースへのプログラムによるアクセスを必要とするアプリケーションで Amazon EC2 インスタンスを使用していますか? その場合には、[EC2 の IAM ロール](#) を使用します。

## IAM ユーザーのアクセスキーを適切に管理する

プログラムによる AWS へのアクセスを設定したい場合、IAM ユーザー用にアクセスキーを作成して、必要なアクセス許可のみをユーザーに付与します。

IAM ユーザーアクセスキーを保護するために、以下の注意事項を守ってください。

- アクセスキーを直接コードに埋め込まないでください。[AWS SDK](#) と [AWS コマンドラインツール](#) では、既知の場所にアクセスキーを置くことができるため、それらをコードで保持する必要がありません。

次のいずれかの場所にアクセスキーを置きます。

- AWS 認証情報ファイル。AWS SDK と AWS CLI では、AWS 認証情報ファイルに保存した認証情報が自動的に使用されます。

AWS 認証情報ファイルの使用については、SDK のドキュメントを参照してください。例としては、AWS SDK for Java デベロッパーガイドの「[AWS 認証情報とリージョンを設定する](#)」および AWS Command Line Interface ユーザーガイドの「[設定と認証情報ファイル](#)」などがあります。

AWS SDK for .NET と AWS Tools for Windows PowerShell の認証情報を保存するには、SDK ストアの使用をお勧めします。詳細については、AWS SDK for .NET デベロッパーガイドの「[SDK Store の使用](#)」を参照してください。

- 環境変数。マルチテナントシステムでは、システム環境変数ではなくユーザー環境変数を選択します。

環境変数を使用した認証情報の保存の詳細については、AWS Command Line Interface ユーザーガイドの「[環境変数](#)」を参照してください。

- 異なるアプリケーションには、異なるアクセスキーを使用します。こうすることで、アクセスキーが公開された場合に、アクセス許可を分離して、個々のアプリケーションに対するアクセスキーを無効化できます。アプリケーションごとに別々のアクセスキーを持つと、[AWS CloudTrail](#) ログファイルに個別のエントリが生成されます。この設定により、特定のアクションを実行したアプリケーションを簡単に判別できます。
- 必要に応じてアクセスキーを更新してください。アクセスキーが危険にさらされる可能性がある場合は、アクセスキーを更新し、以前のアクセスキーを削除してください。詳細については、「[アクセスキーの更新](#)」を参照してください。
- 使用していないアクセスキーを削除します。ユーザーが退職したら、そのユーザーがリソースにアクセスできないようにするため、対応する IAM ユーザーを削除します。アクセスキーが最後に使用された日時を調べるには、[GetAccessKeyLastUsed](#) API (AWS CLI コマンド: [aws iam get-access-key-last-used](#)) を使用します。
- 一時的な認証情報を使用し、最も機密性の高い API 操作ができるよう、多要素認証を設定します。IAM ポリシーを使用して、ユーザーが呼び出すことができる API オペレーションを指定できます。場合によっては、ユーザーが特に機密性の高いアクションを実行することを許可する前に、このユーザーに AWS MFA で認証することを求める追加のセキュリティが必要となることもあります。たとえば、ユーザーに Amazon EC2 RunInstances、DescribeInstances、および



StopInstances アクションの実行を許可するポリシーがすでに存在する可能性があります。ただし TerminateInstances のような有害なアクションを制限し、AWS MFA デバイスによって認証される場合に限り、ユーザーがそのアクションを実行できるように管理することもできます。詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

## AWS アクセスキーを使用してモバイルアプリにアクセスする

AWS モバイルアプリを使用して、限定された AWS サービスと機能にアクセスできます。モバイルアプリは、外出先でもインシデント対応をサポートするのに役立ちます。詳細およびアプリのダウンロードについては、「[AWS コンソールモバイルアプリケーション](#)」を参照してください。

コンソールのパスワードまたはアクセスキーを使用して、モバイルアプリにサインインできます。ベストプラクティスとして、ルートユーザーアクセスキーは使用しないでください。その代わりに、モバイルデバイスでパスワードまたは生体認証ロックを使用することに加えて、モバイルアプリを使用して AWS リソースを管理するための IAM ユーザーを作成することを強くお勧めします。モバイルデバイスを紛失した場合は、その IAM ユーザーのアクセス権を削除します。

### アクセスキーを使用してサインインするには (モバイルアプリ)

1. モバイルデバイスでアプリを開きます。
2. デバイ스에 ID を初めて追加する場合は、[Add an identity (ID を追加)]、[Access keys (アクセスキー)] の順に選択します。

別の ID を使用して既にサインインしている場合は、メニューアイコンを選択し、[Switch identity (ID を切り替える)] を選択します。次に、[Sign in as a different identity (別の ID としてサインイン)]、[Access keys (アクセスキー)] の順に選択します。

3. [Access keys (アクセスキー)] ページで、情報を入力します。
  - Access key ID (アクセスキー ID) – アクセスキー ID を入力します。
  - Secret access key (シークレットアクセスキー) – シークレットアクセスキーを入力します。
  - Identity name (ID 名) – モバイルアプリに表示される ID の名前を入力します。これは、IAM ユーザー名と一致する必要はありません。
  - Identity PIN (ID PIN) – 今後のサインイン時に使用する個人識別番号 (PIN) を作成します。



**Note**

AWS モバイルアプリで生体認証を有効にすると、作成した PIN ではなく、指紋認証または顔認識を使用して検証するように求められます。生体認証に失敗した場合は、代わりに PIN の入力を求められることがあります。

**4. [Verify and add keys] (確認してキーを追加) を選択します。**

モバイルアプリを使用して、一部のリソースにアクセスできるようになりました。

**関連情報**

以下のトピックでは、アクセスキーを使用するために AWS SDK および AWS CLI を設定するためのガイダンスを示しています。

- AWS SDK for Java デベロッパーガイドの「[AWS 認証情報とリージョンの設定](#)」。
- AWS SDK for .NET デベロッパーガイドの「[SDK Store の使用](#)」。
- AWS SDK for PHP デベロッパーガイドの「[SDK への認証情報の提供](#)」。
- Boto 3 (AWS SDK for Python) ドキュメントの「[設定](#)」。
- AWS Tools for Windows PowerShellユーザーガイドの「[AWS 認証情報の使用](#)」。
- AWS Command Line Interface ユーザーガイドの「[設定と認証情報ファイル](#)」。
- AWS SDK for .NET デベロッパーガイドの「[IAM ロールを使用したアクセス権の付与](#)」
- AWS SDK for Java 2.x で [Amazon EC2 用の IAM ロールを設定します](#)

**アクセスキーの監査**

コード内の AWS アクセスキーを確認して、キーが所有するアカウントのものであるかどうかを判断できます。アクセスキー ID は、[aws sts get-access-key-info](#) AWS CLI コマンドまたは [GetAccessKeyInfo](#) AWS API オペレーションを使用して渡すことができます。

AWS CLI および AWS API オペレーションは、AWS アカウント アクセスキーが属するアカウントの ID を返します。AKIA で始まるアクセスキー ID は、IAM ユーザーまたは AWS アカウントのルートユーザーの長期的な認証情報です。ASIA で始まるアクセスキー ID は、AWS STS オペレーションを使用して作成される一時的な認証情報です。レスポンスのアカウントがユーザーに属している場合、ルートユーザーとしてサインインしてルートユーザーのアクセスキーを確認できます。次に、[認証情報レポート](#)を取得して、キーを所有している IAM ユーザーを確認できます。ASIA アクセス

キーの一時認証情報をリクエストしたユーザーを確認するには、CloudTrail ログで AWS STS イベントを表示します。

セキュリティ上の理由から、[AWS CloudTrail ログを確認](#)して、AWS でアクションを実行したユーザーを調べることができます。ロール信頼ポリシーで `sts:SourceIdentity` 条件キーを使用すると、ユーザーがロールを引き受けるときに ID を指定するように要求できます。例えば、IAM ユーザーがセッション名として自分のユーザー名を指定するように要求できます。これにより、AWS の特定のアクションを実行したユーザーを特定できます。詳細については、「[sts:SourceIdentity](#)」を参照してください。

このオペレーションは、アクセスキーの状態を示しません。キーは、アクティブ、非アクティブ、または削除済みのいずれかです。アクティブなキーにオペレーションを実行するアクセス許可がない場合があります。削除済みアクセスキーを指定すると、キーが存在しないというエラーが返されることがあります。

## 紛失したり忘れたりしたパスワードまたは AWS のアクセスキーのリセット

### Important

AWS へのサインインに問題がある場合 ユーザーのタイプに応じて正しい [AWS サインインページ](#)が表示されていることを確認します。AWS アカウントのルートユーザー (アカウント所有者) の場合は、AWS アカウント の作成時に設定した認証情報を使用して AWS にサインインできます。IAM ユーザーの場合、アカウント管理者から AWS へのサインインに使用可能な認証情報をもらうことができます。サポートをリクエストする必要がある場合、このページのフィードバックリンクは使用しないでください。このフォームは AWS サポートではなく、AWS Support ドキュメントチームに送信されます。代わりに、「[お問い合わせ](#)」ページで [AWS アカウントにまだログインできない] を選択してから、表示されているサポートオプションのいずれかを選択します。

メインのサインインページで、電子メールアドレスを入力してルートユーザーとしてサインインするか、アカウント ID を入力して IAM ユーザーとしてサインインする必要があります。パスワードは、ユーザータイプに一致するサインインページでのみ入力できます。詳細については、「[AWS Management Console へのサインイン](#)」を参照してください。

パスワードやアクセスキーを紛失または忘れた場合、IAM からそれらを取得することはできません。代わりに、次の方法を使用してリセットできます。

- AWS アカウントのルートユーザー パスワード - ルートユーザーパスワードを忘れた場合は、AWS Management Console からパスワードをリセットできます。詳細については、このトピックの「[the section called “紛失または忘れたルートユーザーのパスワードのリセット”](#)」を参照してください。
- AWS アカウント のアクセスキー - アカウントのアクセスキーを忘れた場合は、既存のアクセスキーを無効にすることなく、新しいアクセスキーを作成できます。既存のキーを使用していない場合は、それらを削除できます。詳細については、「[ルートユーザーのアクセスキーの作成](#)」および「[ルートユーザーのアクセスキーの削除](#)」を参照してください。
- IAM ユーザーパスワード - ユーザーで、パスワードを忘れた場合は、パスワードのリセットを管理者に依頼する必要があります。管理者によるパスワードの管理方法については、「[IAM ユーザーのパスワードの管理](#)」を参照してください。
- IAM ユーザーアクセスキー - IAM ユーザーで、アクセスキーを忘れた場合は、新しいアクセスキーが必要です。独自のアクセスキーを作成するアクセス許可がある場合は、新しいアクセスキーを作成する手順を「[アクセスキーの管理 \(コンソール\)](#)」で参照してください。必要なアクセス許可を持っていない場合は、新しいアクセスキーの作成を管理者に依頼する必要があります。まだ古いキーを使用している場合は、古いキーを削除しないように管理者に依頼します。管理者によるアクセスキーの管理方法については、「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

## AWS での多要素認証 (MFA) の使用

 [Follow us on Twitter](#)

セキュリティを向上させるには、多要素認証 (MFA) を設定して AWS リソースを保護することを推奨します。AWS アカウントのルートユーザー および IAM ユーザーでは、MFA を使用することができます。ルートユーザーの MFA を有効化すると、ルートユーザーの認証情報のみが影響を受けます。アカウントの IAM ユーザーは固有の認証情報を持つ独立した ID であり、各 ID には固有の MFA 設定があります。

AWS アカウントのルートユーザー および IAM ユーザーに対し、現在サポートされている MFA タイプの任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。サポートされる MFA タイプについては、「[IAM ユーザーが利用可能な MFA タイプ](#)」を参照してください。MFA デバイスを複数使用する場合でも、そのユーザーとして AWS Management Console にログインしたり、AWS CLI を使用してセッションを作成したりするのに必要なのは、1 台の MFA デバイスだけです。

### Note

人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用を必須とすることをお勧めします。AWS IAM Identity Center の使用を検討したことがある場合 IAM Identity

Center を使用すると、複数の AWS アカウント へのアクセスを一元的に管理できます。ユーザーには、割り当てられたすべてのアカウントに対する MFA で保護された Single Sign-On によるアクセスを、1 つの場所から提供することができます。IAM Identity Center では、その内部でユーザー ID の作成および管理を行います。あるいは、既存の SAML 2.0 互換 ID プロバイダーにも簡単に接続することができます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM Identity Center とは?) を参照してください。

## IAM ユーザーが利用可能な MFA タイプ

MFA では、さらなるセキュリティが追加されます。ユーザーが AWS のウェブサイトやサービスにアクセスするときに、通常のサインイン認証情報に加えて、AWS でサポートされている MFA メカニズムからの一意の認証を求められるためです。AWS がサポートする MFA タイプは、パスキーとセキュリティキー、仮想認証機能アプリケーション、およびハードウェア TOTP トークンです。

### パスキーとセキュリティキー

AWS Identity and Access Management は MFA のパスキーとセキュリティキーをサポートします。FIDO 標準に基づいて、パスキーではパブリックキー暗号化が使用され、パスワードよりも安全性の高い、強力でフィッシング耐性のある認証が提供されます。AWS は、デバイスバインドパスキー (セキュリティキー) と同期パスキーの 2 種類のパスキーをサポートしています。

- セキュリティキー: これらは YubiKey などの物理デバイスで、認証の 2 番目の要素として使用されます。
- 同期パスキー: これらは、Google、Apple、Microsoft アカウントなどのプロバイダーの認証情報マネージャー、および 1Password、Dashlane、Bitwarden などのサードパーティーサービスを 2 番目の要素として使用します。

Apple MacBooks の Touch ID や PC の Windows Hello 顔認識などの組み込みの生体認証機能を使用して、認証情報マネージャーのロックを解除し、AWS にサインインできます。パスキーは、指紋、顔、またはデバイス PIN を使用して、選択したプロバイダーで作成されます。デバイス間でパスキーを同期することで、AWS へのサインインが容易になり、使いやすさと回復性が向上します。

FIDO 仕様と互換性のあるすべての [FIDO 認定製品](#) のリストが、FIDO アライアンスから提供されています。1 つのパスキーまたはセキュリティキーで、複数のルートユーザーアカウントと IAM ユーザーをサポートします。IAM ユーザーのパスキーとセキュリティキーの有効化の詳細については、「[パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)」を参照してください。

## 仮想認証アプリケーション

仮想認証アプリケーションは、電話やその他のデバイスで実行され、物理デバイスをエミュレートします。仮想認証アプリは、[タイムベースドワンタイムパスワード \(TOTP\) アルゴリズム](#)を実装しており、単一デバイスで複数のトークンをサポートします。ユーザーは、サインイン中にプロンプトが表示されたら、デバイスから有効なコードを入力する必要があります。ユーザーに割り当てられた各トークンは一意であることが必要です。ユーザーは、別のユーザーのトークンからコードを入力して認証を受けることはできません。

ハードウェアの購入承認の待機中、またはハードウェアの到着を待つ間に、仮想 MFA デバイスを使用することをお勧めします。仮想 MFA デバイスとして使用可能なサポートされているアプリケーションのリストについては、「[多要素認証 \(MFA\)](#)」を参照してください。IAM ユーザーの仮想 MFA デバイスを設定する手順については、「[仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)」を参照してください。

### ハードウェア TOTP トークン

ハードウェアデバイスでは、[タイムベースドワンタイムパスワード \(TOTP\) アルゴリズム](#)に基づいて 6 桁の数値コードが生成されます。サインイン時に、ユーザーはデバイスから取得した有効なコードを 2 番目のウェブページに入力する必要があります。ユーザーに割り当てられた各 MFA デバイスは一意であることが必要です。ユーザーは、別のユーザーのデバイスからコードを入力して認証することはできません。サポートされているハードウェア MFA デバイスの詳細については、「[多要素認証 \(MFA\)](#)」を参照してください。IAM ユーザーのハードウェア TOTP トークンを設定する手順については、「[ハードウェア TOTP トークンの有効化 \(コンソール\)](#)」を参照してください。

物理 MFA デバイスを使用する場合は、ハードウェア TOTP デバイスの代わりにセキュリティキーを使用することをお勧めします。セキュリティキーのメリットは、バッテリーが不要でフィッシング耐性があるという点です。さらに、セキュリティを強化するために、1 台のデバイスで複数のルートユーザーまたは IAM ユーザーをサポートしています。

#### Note

SMS テキストメッセージベース MFA – AWS では、SMS 多要素認証 (MFA) の有効化のサポートを終了しました。SMS テキストメッセージベースの MFA を使用する IAM ユーザーを抱えているお客様は、別のいずれかの方法 ([パスキーまたはセキュリティキー](#)、[仮想 \(ソフトウェアベースの\) MFA デバイス](#)、または [ハードウェア MFA デバイス](#)) に切り替えることをお勧めします。割り当てられた SMS MFA デバイスを使用して、アカウントのユーザーを識別することができます。これを行うには、IAM コンソールに移動して、ナビゲーションペインの [ユーザー] を選択し、テーブルの [MFA] 列の [SMS] を使用してユーザーを探します。

## トピック

- [AWS でのユーザーの MFA デバイスの有効化](#)
- [MFA ステータスのチェック](#)
- [仮想デバイスとハードウェア MFA デバイスの再同期](#)
- [MFA デバイスの無効化](#)
- [MFA デバイスの紛失および故障時の対応](#)
- [MFA 保護 API アクセスの設定](#)
- [サンプルコード: 多要素認証での認証情報のリクエスト](#)

## AWS でのユーザーの MFA デバイスの有効化

MFA を設定する手順は、使用している MFA デバイスのタイプによって異なります。

### トピック

- [MFA デバイスを有効にするための一般的な手順](#)
- [パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)
- [仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)
- [ハードウェア TOTP トークンの有効化 \(コンソール\)](#)
- [仮想 MFA デバイスの有効化と管理 \(AWS CLI または AWS API\)](#)

### MFA デバイスを有効にするための一般的な手順

MFA をセットアップして使用するための手順の概要および関連する情報へのリンクは、以下のとおりです。

#### 注記

この英語のビデオも見ることができますが、「[AWS 多要素認証 \(MFA\) と AWS 予算アラートの設定方法](#)」を参照してください。

1. 以下のいずれかの MFA デバイスを入手します。以下のタイプでの任意の組み合わせについて、AWS アカウントのルートユーザー または IAM ユーザーあたり、最大 8 台の MFA デバイスを有効にできます。



- 仮想 MFA デバイス。これは、[標準ベースの TOTP \(時刻ベースのワンタイムパスワード\) アルゴリズムである RFC 6238](#) に準拠するソフトウェアアプリです。アプリは、電話や他のデバイスにインストールできます。仮想 MFA デバイスとして使用できるサポートされるアプリケーションのリストについては、「[多要素認証](#)」を参照してください。
- [AWS でサポートされている設定](#) のパスキーまたはセキュリティキー。FIDO 仕様と互換性のあるすべての [FIDO 認定製品](#) のリストが、FIDO アライアンスから提供されています。
- サードパーティプロバイダー提供の、ハードウェアベースの MFA デバイス (トークンデバイスなど)。これらのトークンは AWS アカウント でのみ使用されます。詳細については、「[ハードウェア TOTP トークンの有効化 \(コンソール\)](#)」を参照してください。AWS とセキュアに共有された一意のトークンシードを持つトークンのみを使用できます。トークンシードは、トークンの生成時に生成されるシークレットキーです。他のソースから購入したトークンは、IAM では機能しません。互換性を確保するには、[OTP トークン](#) または [OTP ディスプレイカード](#) のいずれかのリンクからハードウェア MFA デバイスを購入する必要があります。

## 2. MFA デバイスを有効にします。

- 仮想もしくはハードウェア TOTP トークン – IAM ユーザーの仮想 MFA デバイスは、AWS CLI コマンドまたは AWS API オペレーションを使用して有効化できます。AWS CLI、AWS API、Tools for Windows PowerShell、またはその他のコマンドラインツールを使用して AWS アカウントのルートユーザー の MFA デバイスを有効にすることはできません。ただし、AWS Management Console を使用して、ルートユーザーの MFA デバイスを有効化することができます。
- パスキーとセキュリティキー – パスキーまたはセキュリティキーを持つルートユーザーと IAM ユーザーは、AWS CLI API や AWS API からではなく、AWS Management Console からのみ有効化を実行できます。

各タイプの MFA デバイスを有効にする方法の詳細については、以下のページを参照してください。

- 仮想 MFA デバイス: [仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)
- パスキーとセキュリティキー: [パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)
- ハードウェア TOTP トークン: [ハードウェア TOTP トークンの有効化 \(コンソール\)](#)

## 3. 複数の MFA デバイスを有効にする (推奨)

- AWS アカウント 内の AWS アカウントのルートユーザー と IAM ユーザーに対しては、複数の MFA デバイスを有効にすることをお勧めします。これにより、AWS アカウント のセキュリティレベルを引き上げ、AWS アカウントのルートユーザー などの権限の高いユーザーに対するアクセスの管理を簡素化できます。



- AWS アカウントのルートユーザー および IAM ユーザーに対し、「[現在サポートされている MFA タイプ](#)」の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。MFA デバイスが複数ある場合でも、そのユーザーとして AWS Management Console にログインしたり、AWS CLI を使用してセッションを作成したりするのに必要なのは、1 台の MFA デバイスだけです。IAM ユーザーは既存の MFA デバイスで認証して、追加の MFA デバイスを有効または無効にする必要があります。
  - MFA デバイスが紛失、盗難、またはアクセス不能になった場合は、残りの MFA デバイスのいずれかを使用して、AWS アカウントでの回復手順を実行することなく AWS アカウントにアクセスできます。MFA デバイスが紛失または盗難に遭った場合は、関連付けられている IAM プリンシパルからそのデバイスを切り離してください。
  - 複数の MFA を使用すると、地理的に離れた場所にいる従業員やリモートで作業している従業員は、ハードウェアベースの MFA を使用して AWS にアクセスできます。ハードウェアデバイスを 1 台送付したり、1 台のハードウェアデバイスを従業員間で物理的に交換したりする必要はありません。
  - IAM プリンシパルに追加の MFA デバイスを使用すると、1 つ以上の MFA を日常的に使用できると同時に、物理 MFA デバイスをボールドなどの安全な物理的な場所に保存したり、バックアップや冗長性を確保したりできます。
4. ログインするか、または AWS リソースにアクセスする場合には、MFA デバイスを使用します。
- パスキーとセキュリティキー – AWS ウェブサイトにアクセスするには、認証情報を入力し、プロンプトが表示されたら、パスキーの種類に応じて FIDO セキュリティキーをタップするか、デバイス PIN を入力するか、指紋または顔を提示します。
  - 仮想 MFA デバイスおよびハードウェア MFA デバイス – AWS のウェブサイトにアクセスするには、ユーザー名とパスワードに加えて、デバイスの MFA コードが必要です。

MFA で保護された API オペレーションにアクセスするには、以下のものがが必要です。

- MFA コード
- MFA デバイスの ID (物理デバイスのデバイスシリアル番号または AWS で定義された仮想デバイスの ARN)
- 通常のアクセスキー ID とシークレットアクセスキー

#### メモ

- FIDO セキュリティキーの MFA 情報を AWS STS API オペレーションに渡して一時的認証情報をリクエストすることはできません。

- AWS CLI コマンドまたは AWS API 操作を使用して [FIDO セキュリティキー](#) を有効にすることはできません。
- 同じ名前を複数のルートまたは IAM MFA デバイスに使用することはできません。

詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

### パスキーまたはセキュリティキーの有効化 (コンソール)

パスキーは、AWS リソースを保護するために使用できる [多要素認証 \(MFA\) デバイス](#) の一種です。AWS は、同期パスキーと、セキュリティキーとも呼ばれるデバイスにバインドされたパスキーをサポートします。

同期パスキーを使用すると、IAM ユーザーは、多くのデバイスで (新しいデバイスであっても)、FIDO サインイン認証情報にアクセスできます。すべてのアカウントですべてのデバイスを再登録する必要はありません。同期パスキーには、Google、Apple、Microsoft などのファーストパーティ認証情報マネージャーと、1Password、Dashlane、Bitwarden などのサードパーティ認証情報マネージャーが 2 番目の要素として含まれます。デバイス上の生体認証 (例: TouchID、FaceID、Windows Hello) を使用して、選択した認証情報マネージャーのロックを解除してパスキーを使用することもできます。

または、デバイスにバインドされたパスキーは、コンピュータの USB ポートに接続する FIDO セキュリティキーにバインドされます。プロンプトが表示されたらタップしてサインインプロセスを安全に完了します。すでに他のサービスで FIDO セキュリティキーを使用していて、[AWS でサポートされている構成](#) (Yubico の YubiKey 5 シリーズなど) がある場合は、そのキーも AWS で使用できます。それ以外の場合、AWS で MFA 用に WebAuthn を使用するには、FIDO セキュリティキーを購入する必要があります。さらに、FIDO セキュリティキーは、同じデバイスで複数の IAM ユーザーまたはルートユーザーをサポートできるため、アカウントセキュリティのユーティリティが強化されます。両方のデバイスタイプの仕様と購入情報については、「[多要素認証](#)」を参照してください。

AWS アカウントのルートユーザー および IAM ユーザーに対し、[\[現在サポートされている MFA タイプ\]](#) の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。MFA デバイスが複数ある場合でも、そのユーザーとして AWS Management Console にログインしたり、AWS CLI を使用してセッションを作成したりするのに必要なのは、1 台の MFA デバイスだけです。複数の MFA デバイスを登録することをお勧めします。例えば、組み込みの認証アプリを登録し、物理的に安全な場所に保管するセキュリティキーも登録することができます。組み込みの認証アプリを使用できない場合は、登録済みのセキュリティキーを使用できます。認証アプリについては、認証アプリが搭載された

デバイスを紛失したり破損したりした場合に、アカウントにアクセスできなくなるのを防ぐため、それらのアプリのクラウドバックアップまたは同期機能を有効にすることもお勧めします。

#### Note

人間のユーザーが AWS にアクセスする場合には、一時的な認証情報の使用を推奨します。ユーザーは、ID プロバイダーを使用して AWS にフェデレーションし、会社の認証情報と MFA 設定で認証できます。AWS へのアクセスとビジネスアプリケーションを管理する場合は、IAM Identity Center の使用をお勧めします。詳細については、「[IAM Identity Center ユーザーガイド](#)」を参照してください。

## トピック

- [必要なアクセス許可](#)
- [自身の IAM ユーザーのパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)
- [別の IAM ユーザーのパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)
- [パスキーまたはセキュリティキーを置き換える](#)
- [パスキーとセキュリティキーを使用するためのサポートされる設定](#)

## 必要なアクセス許可

重要な MFA 関連のアクションを保護しながら、独自の IAM ユーザー用に FIDO パスキーを管理するには、次のポリシーのアクセス許可が必要です。

#### Note

ARN は静的な値であり、認証機能を登録するためにどのプロトコルが使用されたかを示すものではありません。U2F は廃止されたため、新しい実装はすべて WebAuthn を使用しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
```


```
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "DenyAllExceptListedIfNoMFA",
    "Effect": "Deny",
    "NotAction": [
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
    ],
    "Resource": "*",
    "Condition": {
        "BoolIfExists": {
            "aws:MultiFactorAuthPresent": "false"
        }
    }
}
]
```

自身の IAM ユーザーのパスキーまたはセキュリティキーを有効にする (コンソール)

自身の IAM ユーザーのパスキーまたはセキュリティキーは、AWS CLI API や AWS API からではなく、AWS Management Console からのみ有効にできます。セキュリティキーを有効にするには、そのデバイスに物理的にアクセスできる必要があります。

自身の IAM ユーザーのパスキーまたはセキュリティキーを有効にするには (コンソール)

1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

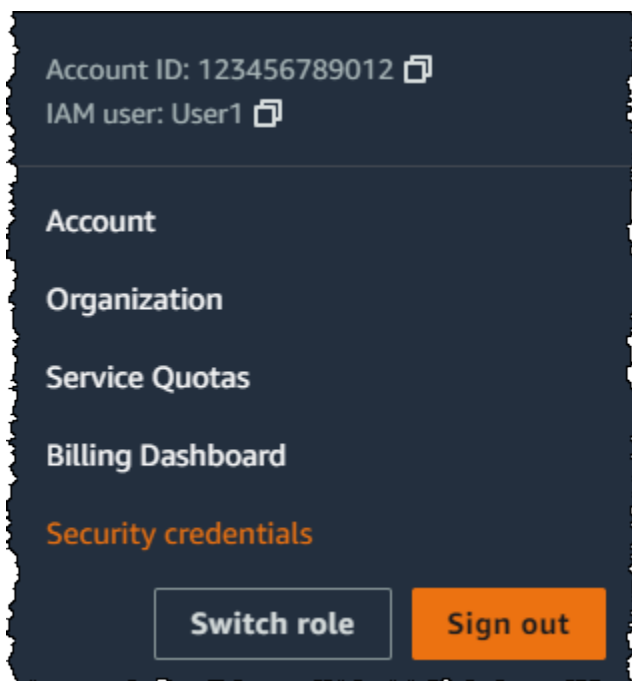
 Note

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある

場合は、ページの下部にある[別のアカウントにサインイン]を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようになります。

AWS アカウント アカウント ID の取得については、管理者にお問い合わせください。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



3. 選択した IAM ユーザーのページで、[セキュリティ認証情報] タブを選択します。
4. [Multi-factor authentication (MFA) (多要素認証 (MFA)) で、[Assign MFA device] (MFA デバイスの割り当て) を選択します。
5. [MFA デバイス名] ページで、[デバイス名] を入力し、[パスキーまたはセキュリティキー] を選択し、[次へ] を選択します。
6. [デバイスの設定] で、パスキーを設定します。顔や指紋などの生体認証データを使用、デバイス PIN を使用、またはコンピュータの USB ポートに FIDO セキュリティキーを挿入してタップすることで、パスキーを作成します。
7. ブラウザの指示に従って、[続行] を選択します。

これで、AWS で使用するパスキーまたはセキュリティキーが登録されました。AWS Management Consoleでの MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

別の IAM ユーザーのパスキーまたはセキュリティキーを有効にする (コンソール)

別の IAM ユーザーのパスキーまたはセキュリティキーは、AWS CLI API や AWS API からではなく、AWS Management Console からのみ有効にすることができます。

別の IAM ユーザーのパスキーまたはセキュリティキーを有効にするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. [ユーザー] で、MFA を有効化するユーザーの名前を選択します。
4. 選択した IAM ユーザーのページで、[セキュリティ認証情報] タブを選択します。
5. [Multi-factor authentication (MFA) (多要素認証 (MFA))] で、[Assign MFA device] (MFA デバイスの割り当て) を選択します。
6. [MFA デバイス名] ページで、[デバイス名] を入力し、[パスキーまたはセキュリティキー] を選択し、[次へ] を選択します。
7. [デバイスの設定] で、パスキーを設定します。顔や指紋などの生体認証データを使用、デバイス PIN を使用、またはコンピュータの USB ポートに FIDO セキュリティキーを挿入してタップすることで、パスキーを作成します。
8. ブラウザの指示に従って、[続行] を選択します。

これで、別の IAM ユーザーが AWS で使用するパスキーまたはセキュリティキーが登録されました。AWS Management Consoleでの MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

パスキーまたはセキュリティキーを置き換える

[現在サポートされている MFA タイプ](#)の任意の組み合わせで、一度に最大 8 台の MFA デバイスを、AWS アカウントのルートユーザー および IAM ユーザーに割り当てることができます。ユーザーが FIDO の認証装置を紛失した場合や、何らかの理由で交換する必要がある場合、最初に古い FIDO 認証装置を無効化する必要があります。その後、そのユーザー用に新しい MFA デバイスを追加できます。

- IAM ユーザーに関連付けられているデバイスを非アクティブ化するには、「[MFA デバイスの無効化](#)」を参照してください。
- IAM ユーザー用に新しい FIDO セキュリティキーを追加するには、「[自身の IAM ユーザーのパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)」を参照してください。

新しいパスキーまたはセキュリティキーにアクセスできない場合は、新しい仮想 MFA デバイスまたはハードウェア TOTP トークンを有効化します。手順については、以下のいずれかを参照してください。

- [仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)
- [ハードウェア TOTP トークンの有効化 \(コンソール\)](#)

### パスキーとセキュリティキーを使用するためのサポートされる設定

現在サポートされている設定を使用して、IAM での多要素認証 (MFA) メソッドとして、FIDO2 のデバイスにバインドされたパスキー (セキュリティキーとも呼ばれます) を使用できます。これには、IAM でサポートされている FIDO2 デバイスや、FIDO2 をサポートしているブラウザなどが含まれます。FIDO2 デバイスを登録する前に、お使いのブラウザとオペレーティングシステム (OS) のバージョンが最新であることを確認してください。機能の動作は、ブラウザ、認証システム、および OS クライアントによって異なる場合があります。あるブラウザでデバイスの登録に失敗した場合は、別のブラウザで登録を試みることができます。

FIDO2 はオープンな認証標準の FIDO U2F の拡張であり、公開鍵暗号に基づくセキュリティと同等の高レベルのセキュリティを提供します。FIDO2 は、W3C Web 認証仕様 (WebAuthn API) と、アプリケーション層プロトコルである FIDO アライアンスの Client-to-Authenticator Protocol (CTAP) で構成されています。CTAP は、ブラウザやオペレーティングシステムなどのクライアントまたはプラットフォームと外部認証システムとの間の通信を可能にします。AWS で FIDO 認定の認証機能を有効にすると、セキュリティキーにより、AWS でのみ使用するための新しいキーペアが作成されます。まず、認証情報を入力します。プロンプトが表示されたら、AWS によって発行された認証チャレンジに応答するセキュリティキーをタップします。FIDO2 標準の詳細については、「[FIDO2 プロジェクト](#)」を参照してください。

### AWS でサポートされている FIDO2 デバイス

IAM では、USB、Bluetooth、または NFC 経由でデバイスに接続する FIDO2 セキュリティデバイスをサポートしています。IAM は、TouchID、FaceID、Windows Hello などのプラットフォーム認証機能もサポートしています。



**Note**

AWS は、FIDO2 デバイスを検証するためにコンピュータの物理的 USB ポートにアクセスする必要があります。セキュリティキーは、仮想マシン、リモート接続、またはブラウザのシークレットモードでは機能しません。

FIDO アライアンスでは、FIDO 仕様と互換性のあるすべての [FIDO2 製品](#) のリストを公開しています。

**FIDO2 をサポートするブラウザ**

ウェブブラウザで実行される FIDO2 セキュリティデバイスの可用性は、ブラウザとオペレーティングシステムの組み合わせによって異なります。現在、以下のブラウザでセキュリティキーの使用がサポートされています。

	macOS 10.15+	Windows 10	Linux	IOS 14.5 以 降	Android 7 以 降
Chrome	はい	はい	はい	はい	いいえ
Safari	はい	いいえ	いいえ	はい	いいえ
Edge	はい	はい	いいえ	はい	いいえ
Firefox	はい	はい	いいえ	はい	いいえ

**Note**

現在、FIDO2 をサポートしている Firefox のほとんどのバージョンでは、デフォルト状態で、そのサポートが有効になっていません。Firefox で FIDO2 のサポートを有効にする手順については、「[FIDO セキュリティキーのトラブルシューティング](#)」を参照してください。

YubiKey など、FIDO2 認定デバイスのブラウザのサポートについては、「[FIDO2 および U2F のオペレーティングシステムとウェブブラウザのサポート](#)」を参照してください。

## ブラウザプラグイン

AWS は、FIDO2 をネイティブにサポートするブラウザのみをサポートしています。AWS は FIDO2 ブラウザのサポートを追加するためのプラグインの使用をサポートしていません。一部のブラウザプラグインは FIDO と互換性がなく、FIDO2 セキュリティキーで予期しない結果が生じることがあります。

ブラウザプラグインの無効化やその他のトラブルシューティングのヒントについては、「[FIDO セキュリティキーを有効にできない](#)」を参照してください。

## デバイス証明書

FIPS 検証や FIDO 証明書レベルなど、デバイス関連の証明書を取得して割り当てるのは、セキュリティキーの登録時だけです。デバイス証明書は [FIDO Alliance Metadata Service \(MDS\)](#) から取得できます。セキュリティキーの証明書ステータスまたはレベルが変更されても、デバイスタグに自動的に反映されることはありません。デバイスの証明書情報を更新するには、デバイスを登録し直して、更新された証明書情報を取得します。

AWS では、FIDO MDS から取得したデバイス登録時の条件キーとして、FIPS-140-2、FIPS-140-3、および FIDO 証明書レベルの証明書タイプが用意されています。希望する証明書タイプとレベルに基づいて、IAM ポリシーに特定の認証者の登録を指定できます。詳細については、以下のポリシーを参照してください。

### デバイス証明書のポリシーの例

以下のユースケースは、FIPS 証明書を持つ MFA デバイスを登録できるようにするサンプルポリシーを示しています。

#### トピック

- [ユースケース 1: FIPS-140-2 L2 証明書を持つデバイスのみの登録を許可する](#)
- [ユースケース 2: FIPS-140-2 L2 および FIDO L1 証明書を持つデバイスの登録を許可する](#)
- [ユースケース 3: FIPS-140-2 L2 または FIPS-140-3 L2 証明書を持つデバイスの登録を許可する](#)
- [ユースケース 4: FIPS-140-2 L2 認証を持ち、仮想認証システムやハードウェア TOTP などのその他の種類の MFA をサポートするデバイスの登録を許可する](#)

#### ユースケース 1: FIPS-140-2 L2 証明書を持つデバイスのみの登録を許可する

```
{
  "Version": "2012-10-17",
```

```
"Statement": [{
  "Effect": "Allow",
  "Action": "iam:EnableMFADevice",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "iam:RegisterSecurityKey" : "Create"
    }
  }
},
{
  "Effect": "Allow",
  "Action": "iam:EnableMFADevice",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "iam:RegisterSecurityKey" : "Activate",
      "iam:FIDO-FIPS-140-2-certification": "L2"
    }
  }
}
]
```

## ユースケース 2: FIPS-140-2 L2 および FIDO L1 証明書を持つデバイスの登録を許可する

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Create"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
```

```

        "StringEquals": {
            "iam:RegisterSecurityKey" : "Activate",
            "iam:FIDO-FIPS-140-2-certification": "L2",
            "iam:FIDO-certification": "L1"
        }
    }
}
]
}

```

### ユースケース 3: FIPS-140-2 L2 または FIPS-140-3 L2 証明書を持つデバイスの登録を許可する

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Create"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Activate",
        "iam:FIDO-FIPS-140-2-certification": "L2"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Activate",
        "iam:FIDO-FIPS-140-3-certification": "L2"
      }
    }
  }
}

```

```
    }
  }
}
]
```

ユースケース 4: FIPS-140-2 L2 認証を持ち、仮想認証システムやハードウェア TOTP などのその他の種類の MFA をサポートするデバイスの登録を許可する

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:RegisterSecurityKey": "Create"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:RegisterSecurityKey": "Activate",
          "iam:FIPS-140-2-certification": "L2"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "Null": {
          "iam:RegisterSecurityKey": "true"
        }
      }
    }
  ]
}
```

```
]
}
```

## AWS CLI および AWS API

AWS は、パスキーとセキュリティキーの使用を AWS Management Console でのみサポートしています。MFA に対するパスキーとセキュリティキーの使用は [AWS CLI](#) と [AWS API](#) ではサポートされていません。また、[MFA 保護 API オペレーション](#)へのアクセスでもサポートされていません。

## 追加リソース

- AWS でのパスキーとセキュリティキーの使用の詳細については、「[パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)」を参照してください。
- AWS でのパスキーとセキュリティキーのトラブルシューティングについては、「[FIDO セキュリティキーのトラブルシューティング](#)」を参照してください。
- FIDO2 サポートに関する一般的な業界情報については、「[FIDO2 プロジェクト](#)」を参照してください。

## 仮想 Multi-Factor Authentication (MFA) デバイスの有効化 (コンソール)

電話や他のデバイスを仮想多要素認証 (MFA) デバイスとして使用できます。これを行うには、[標準ベースの TOTP \(時刻ベースのワンタイムパスワード\) アルゴリズムである RFC 6238](#) に準拠するモバイルアプリをインストールします。これらのアプリは、6 桁の認証コードを生成します。これらはセキュリティ保護されていないモバイルデバイス上で実行できるため、仮想 MFA から、FIDO デバイスと同レベルのセキュリティが提供されない場合があります。ハードウェアの購入承認の待機中、またはハードウェアの到着を待つ間に、仮想 MFA デバイスを使用することをお勧めします。

一般的な仮想 MFA アプリでは、複数の仮想デバイスの作成がサポートされているため、複数の AWS アカウント またはユーザーに対しても同じアプリを使用できます。AWS アカウントのルートユーザー および IAM ユーザーに対し、[\[現在サポートされている MFA タイプ\]](#)の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。MFA デバイスが複数ある場合でも、そのユーザーとして AWS Management Console にログインしたり、AWS CLI を使用してセッションを作成したりするのに必要なのは、1 台の MFA デバイスだけです。複数の MFA デバイスを登録することをお勧めします。認証アプリについては、認証アプリが搭載されたデバイスを紛失したり破損したりした場合に、アカウントにアクセスできなくなるのを防ぐため、それらのアプリのクラウドバックアップまたは同期機能を有効にすることもお勧めします。

使用できる仮想 MFA アプリのリストについては、「[多要素認証](#)」を参照してください。AWS では、6 桁の OTP を生成する仮想 MFA アプリが必要です。

## トピック

- [必要な許可](#)
- [IAM ユーザーの仮想 MFA デバイスの有効化 \(コンソール\)](#)
- [仮想 MFA デバイスの交換](#)

### 必要な許可

IAM ユーザーの仮想 MFA デバイスを更新するには、次のポリシーのアクセス許可が必要です: [AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の MFA デバイスを管理できるようにします](#)。

### IAM ユーザーの仮想 MFA デバイスの有効化 (コンソール)

AWS Management Console で IAM を使用して、アカウントのユーザーの仮想 MFA デバイスを有効化および管理することができます。IAM リソース (仮想 MFA デバイスを含む) にタグをアタッチして、タグへのアクセスを特定、整理、制御することができます。仮想 MFA デバイスにタグを付けることができるのは、AWS CLI または AWS API を使用する場合のみです。AWS CLI または AWS API を使用して、MFA デバイスを有効化および管理するには、「[仮想 MFA デバイスの有効化と管理 \(AWS CLI または AWS API\)](#)」を参照してください。IAM リソースのタグ付けの詳細については、「[IAM リソースのタグ付け](#)」を参照してください。

#### Note

MFA を設定するには、ユーザーの仮想 MFA デバイスをホストするハードウェアに物理的にアクセスできる必要があります。例えば、スマートフォンで実行される仮想 MFA デバイスを使用するユーザー用に MFA を設定するとします。その場合、ウィザードを完了するには、そのスマートフォンを利用できる必要があります。このため、ユーザーが自分の仮想 MFA デバイスを設定して管理できるようにすることをお勧めします。この場合、必要な IAM アクションを実行する権限をユーザーに付与する必要があります。このアクセス許可を付与する IAM ポリシーの詳細および例については、「[IAM チュートリアル: ユーザーに自分の認証情報および MFA 設定を許可する](#)」およびポリシーの例「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の MFA デバイスを管理できるようにします](#)」を参照してください。



## IAM ユーザーの仮想 MFA デバイスを有効にするには (コンソール)

1. AWS Management Consoleにサインインして、IAM コンソールを開きます <https://console.aws.amazon.com/iam/>。
2. ナビゲーションペインで [Users] (ユーザー) を選択します。
3. [Users] (ユーザー) のリストで、IAM ユーザー名を選択します。
4. [Security Credentials] タブを選択します。[Multi-factor authentication (MFA) (多要素認証 (MFA))] で、[Assign MFA device] (MFA デバイスの割り当て) を選択します。
5. ウィザードで [デバイス名] を入力し、[認証アプリ]、[次へ] の順に選択します。

IAM が QR コードを含む仮想 MFA デバイスの設定情報を生成して表示します。図は、QR コードに対応していないデバイスでの手動入力に利用できる「シークレット設定キー」を示しています。

6. 仮想 MFA アプリを開きます。仮想 MFA デバイスをホストするために使用できるアプリケーションのリストについては、「[多要素認証](#)」を参照してください。

仮想 MFA アプリが複数の仮想 MFA デバイスまたはアカウントをサポートしている場合は、新しい仮想 MFA デバイスまたはアカウントを作成するオプションを選択します。

7. MFA アプリが QR コードをサポートしているかどうかを確認してから、次のいずれかを実行します。
  - ウィザードから [Show QR code] (QR コードの表示) を選択し、アプリを使用して QR コードをスキャンします。例えば、カメラアイコンまたは [Scan code] (スキャンコード) に似たオプションを選択し、デバイスのカメラを使用してコードをスキャンします。
  - 同じウィザードで [Show secret key] (シークレットキーを表示) を選択した後、MFA アプリにシークレットキーを入力します。

これで仮想 MFA デバイスはワンタイムパスワードの生成を開始します。

8. [デバイスの設定] ページで、[MFA コード 1] ボックスに、現在仮想 MFA デバイスに表示されているワンタイムパスワードを入力します。デバイスが新しいワンタイムパスワードを生成するまで待ちます (最長 30 秒)。生成されたら [MFA code 2 (MFA コード 2)] ボックスに 2 つ目のワンタイムパスワードを入力します。[Add MFA] (MFA を追加) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けられますが、その MFA デバイスは同期されません。これは、時刻ベースのワンタイムパスワード (TOTP) の有効期間が短いために起こります。その場合は、[デバイスの再同期](#)ができます。

これで仮想 MFA デバイスを AWS で使用できます。AWS Management Console での MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

### 仮想 MFA デバイスの交換

AWS アカウントのルートユーザー および IAM ユーザーに対し、「[現在サポートされている MFA タイプ](#)」の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。ユーザーがデバイスを紛失したか、何らかの理由で交換する必要がある場合、最初に古いデバイスを非アクティブ化する必要があります。その後、新しいデバイスをユーザーに追加できます。

- 別の IAM ユーザーに関連付けられているデバイスを非アクティブ化するには、「[MFA デバイスの無効化](#)」を参照してください。
- 別の IAM ユーザーの交換用の仮想 MFA デバイスを追加するには、上記の「[IAM ユーザーの仮想 MFA デバイスの有効化 \(コンソール\)](#)」の手順に従います。
- AWS アカウントのルートユーザー ユーザーの交換用の仮想 MFA デバイスを追加するには、[AWS アカウントのルートユーザー \(コンソール\) の仮想 MFA デバイスを有効にします](#) の手順に従います。

### ハードウェア TOTP トークンの有効化 (コンソール)

ハードウェア TOTP トークンは、タイムベースドワンタイムパスワード (TOTP) アルゴリズムに基づいて、6 桁の数値コードを生成します。ユーザーは、サインインプロセス中に求められたら、デバイスから有効なコードを入力する必要があります。ユーザーに割り当てられる各 MFA デバイスは一意であり、他のユーザーのデバイス向けのコードでは認証されません。MFA デバイスをアカウント間またはユーザー間で共有することはできません。

ハードウェア TOTP トークンと [FIDO セキュリティキー](#) は、いずれもお客様が購入する物理デバイスです。ハードウェア MFA デバイスは、ユーザーが AWS にサインインすると認証用の TOTP コー

トを生成します。バッテリーに依存しているため、時間の経過とともにバッテリーの交換と AWS との再同期が必要になる場合があります。パブリックキー暗号化を利用した FIDO セキュリティキーは、バッテリーを必要とせず、シームレスな認証プロセスを提供します。フィッシング耐性を高めるためには、FIDO セキュリティキーを使用することをお勧めします。FIDO セキュリティキーは、TOTP デバイスの代わりに使用できる、よりセキュアな手段です。さらに、FIDO セキュリティキーは、同じデバイスで複数の IAM ユーザーまたはルートユーザーをサポートできるため、アカウントセキュリティのユーティリティが強化されます。両方のデバイスタイプの仕様と購入情報については、「[多要素認証](#)」を参照してください。

IAM ユーザーのハードウェア TOTP デバイスは、AWS Management Console、コマンドライン、または IAM API を使用して有効にすることができます。AWS アカウントのルートユーザーの MFA デバイス設定を有効にするには、「[AWS アカウントのルートユーザー用にハードウェア TOTP トークンを有効にする \(コンソール\)](#)」を参照してください。

AWS アカウントのルートユーザー および IAM ユーザーに対し、「[現在サポートされている MFA タイプ](#)」の任意の組み合わせで、最大 8 台の MFA デバイスを登録できます。MFA デバイスが複数ある場合でも、そのユーザーとして AWS Management Console にログインしたり、AWS CLI を使用してセッションを作成したりするのに必要なのは、1 台の MFA デバイスだけです。

#### Important

MFA デバイスを紛失したりアクセスできなくなったりした場合に、ユーザーが引き続きアカウントにアクセスできるようにするため、複数の MFA デバイスを利用可能にしておくことをお勧めします。

#### Note

コマンドラインから MFA デバイスを有効化するには、[aws iam enable-mfa-device](#) を使用します。IAM API で MFA デバイスを有効化するには、[EnableMFADevice](#) オペレーションを使用します。

## トピック

- [必要なアクセス許可](#)
- [自身の IAM ユーザーでハードウェア TOTP トークンを有効にする \(コンソール\)](#)

- [別の IAM ユーザーのハードウェア TOTP トークンを有効にする \(コンソール\)](#)
- [物理的な MFA デバイスの交換](#)

## 必要なアクセス許可

重要な MFA 関連のアクションを保護しながら、IAM ユーザー用にハードウェア TOTP トークンを管理するには、以下のポリシーによるアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "DenyAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}",
      "Condition": {
        "BoolIfExists": {
          "aws:MultiFactorAuthPresent": "false"
        }
      }
    }
  ]
}
```

## 自身の IAM ユーザーでハードウェア TOTP トークンを有効にする (コンソール)

自分のハードウェア TOTP トークンは、AWS Management Console から有効化できます。

### Note

ハードウェア TOTP トークンを有効にする前に、そのデバイスに対し物理的なアクセス権限を持つ必要があります。

## 自身の IAM ユーザーのハードウェア TOTP トークンを有効にするには (コンソール)

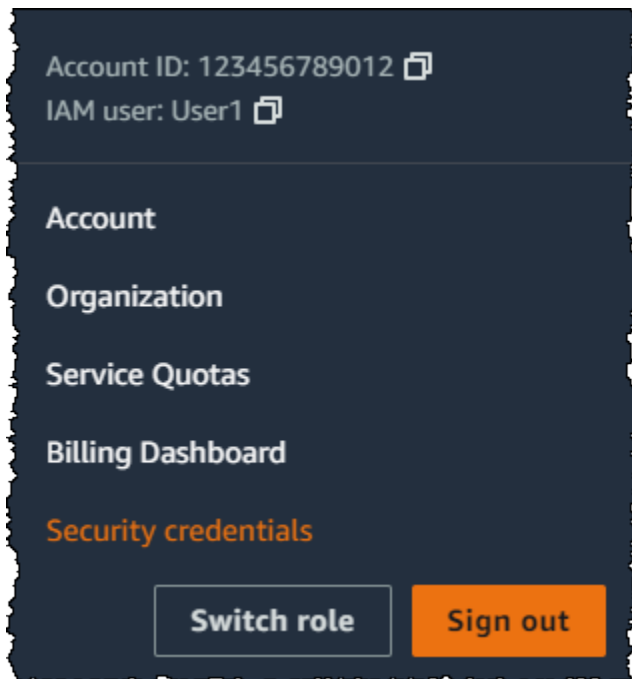
1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

### Note

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある場合は、ページの下部にある[別のアカウントにサインイン]を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようになります。

AWS アカウント アカウント ID の取得については、管理者にお問い合わせください。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



3. [AWS IAM 認証情報] タブの [多要素認証 (MFA)] セクションで、[MFA デバイスの割り当て] を選択します。
4. ウィザード内で [Device name] (デバイス名) に入力した後、[Hardware TOTP token] (ハードウェア TOTP トークン)、[Next] (次へ) の順に選択します。
5. 対象デバイスのシリアルナンバーを入力します。シリアルナンバーは、通常、デバイスの背面にあります。
6. MFA デバイスに表示されている 6 桁の数字を [MFA code 1 (MFA コード 1)] に入力します。デバイス前面のボタンを押して数字を表示する場合があります。



7. デバイスがコードを更新するまで 30 秒ほど待ち、更新後に表示された 6 桁の数字を [MFA code 2 (MFA コード 2)] に入力します。デバイス前面のボタンを再度押して新しい数字を表示する場合があります。
8. [Add MFA] (MFA を追加) を選択します。

**⚠ Important**

認証コードを生成した後すぐに、リクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けられませんが、その MFA デバイスは同期されなくなります。これは、タイム

ベースドワンタイムパスワード (TOTP) の有効期間が短いため起こります。その場合は、[デバイスの再同期](#)ができます。

デバイスを AWS で使用する準備が整いました。AWS Management Consoleでの MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

別の IAM ユーザーのハードウェア TOTP トークンを有効にする (コンソール)

AWS Management Console から別の IAM ユーザーのハードウェア TOTP トークンを有効にできません。

別の IAM ユーザーのハードウェア TOTP トークンを有効にするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. MFA を有効化するユーザーの名前を選択します。
4. [Security Credentials] タブを選択します。[Multi-factor authentication (MFA) (多要素認証 (MFA))] で、[Assign MFA device] (MFA デバイスの割り当て) を選択します。
5. ウィザード内で [Device name] (デバイス名) に入力した後、[Hardware TOTP token] (ハードウェア TOTP トークン)、[Next] (次へ) の順に選択します。
6. 対象デバイスのシリアルナンバーを入力します。シリアルナンバーは、通常、デバイスの背面にあります。
7. MFA デバイスに表示されている 6 桁の数字を [MFA code 1 (MFA コード 1)] に入力します。デバイス前面のボタンを押して数字を表示する場合があります。



8. デバイスがコードを更新するまで 30 秒ほど待ち、更新後に表示された 6 桁の数字を [MFA code 2 (MFA コード 2)] に入力します。デバイス前面のボタンを再度押して新しい数字を表示する場合があります。
9. [Add MFA] (MFA を追加) を選択します。



**⚠ Important**

認証コードを生成した後すぐに、リクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けられませんが、その MFA デバイスは同期されなくなります。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。その場合は、[デバイスの再同期](#)ができます。

デバイスを AWS で使用する準備が整いました。AWS Management Consoleでの MFA 利用の詳細については、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

### 物理的な MFA デバイスの交換

「[現在サポートされている MFA タイプ](#)」の任意の組み合わせで、一度に最大 8 台の MFA デバイスを、AWS アカウントのルートユーザー および IAM ユーザーに割り当てることができます。ユーザーがデバイスを紛失したか、何らかの理由で交換する必要がある場合、最初に古いデバイスを非アクティブ化する必要があります。その後、新しいデバイスをユーザーに追加できます。

- 現在ユーザーに関連付けられているデバイスを非アクティブ化するには、「[MFA デバイスの無効化](#)」を参照してください。
- IAM ユーザーに対し、交換用ハードウェア TOTP トークンを追加するには、このトピックの前半にある手順「[別の IAM ユーザーのハードウェア TOTP トークンを有効にする \(コンソール\)](#)」の各ステップに従います。
- AWS アカウントのルートユーザー に対し、交換用ハードウェア TOTP トークンを追加するには、このトピックの上記手順「[AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする \(コンソール\)](#)」の各ステップに従います。

### 仮想 MFA デバイスの有効化と管理 (AWS CLI または AWS API)

IAM ユーザーの仮想 MFA デバイスを有効にするには、AWS CLI コマンドまたは AWS API オペレーションを使用します。AWS CLI、AWS API、Tools for Windows PowerShell、またはその他のコマンドラインツールを使用して AWS アカウントのルートユーザー の MFA デバイスを有効にすることはできません。ただし、AWS Management Console を使用して、ルートユーザーの MFA デバイスを有効化することができます。

AWS Management Console から MFA デバイスを有効にすると、コンソールがお客様に代わって複数のステップを実行します。代わりに AWS CLI、Tools for Windows PowerShell、または AWS API を使用して仮想デバイスを作成する場合は、それらのステップを適切な順序で自分で実行する必要があります。たとえば、仮想 MFA デバイスを作成するには、IAM オブジェクトを作成し、文字列または QR コードグラフィックとしてコードを抽出する必要があります。次に、デバイスを IAM ユーザーと同期させて関連付ける必要があります。詳細については、[New-IAMVirtualMFADevice](#) の [Examples] (例) セクションを参照してください。物理デバイスの場合は作成ステップを飛ばし、デバイスを同期してユーザーに関連付ける手順へ直接進んでください。

IAM リソース (仮想 MFA デバイスを含む) にタグをアタッチして、タグへのアクセスを特定、整理、制御することができます。仮想 MFA デバイスにタグを付けることができるのは、AWS CLI または AWS API を使用する場合のみです。

SDK または CLI を使用する IAM ユーザーは、[EnableMFADevice](#) を呼び出して追加の MFA デバイスを有効にするか、[DeactivateMFADevice](#) を呼び出して既存の MFA デバイスを無効にできます。これを正常に行うには、まず [GetSessionToken](#) を呼び出し、既存の MFA デバイスで MFA コードを送信する必要があります。この呼び出しで、一時的なセキュリティ認証情報を返されます。この認証情報を使用して、MFA 認証を必要とする API オペレーションに署名できます。リクエストとレスポンスの例については、「[GetSessionToken—信頼されていない環境にあるユーザー向けの一時的認証情報](#)」を参照してください。

IAM で仮想デバイスのエンティティを作成し、仮想 MFA デバイスを表すには

これらのコマンドは、次のコマンドの多くでシリアルナンバーの代わりに使用されるデバイスの ARN を提供します。

- AWS CLI: [aws iam create-virtual-mfa-device](#)
- AWS API: [CreateVirtualMFADevice](#)

AWS を使用するように MFA デバイスを有効にするには

これらのコマンドでは、デバイスを AWS と同期させて、ユーザーと関連付けます。デバイスが仮想デバイスの場合、仮想デバイスの ARN をシリアルナンバーとして使用します。

#### Important

認証コードを生成した後すぐに、リクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けら

れますが、その MFA デバイスは同期されなくなります。これは、時刻ベースのワンタイムパスワード (TOTP) の有効期間が短いために起こります。この場合は、次のコマンドを使用してデバイスを再同期できます。

- AWS CLI: [aws iam enable-mfa-device](#)
- AWS API: [EnableMFADevice](#)

デバイスを無効にするには

以下のコマンドでは、デバイスとユーザーの関連付けを解除し、デバイスを無効化します。デバイスが仮想デバイスの場合、仮想デバイスの ARN をシリアルナンバーとして使用します。別途、仮想デバイスのエンティティも削除する必要があります。

- AWS CLI: [aws iam deactivate-mfa-device](#)
- AWS API: [DeactivateMFADevice](#)

仮想 MFA デバイスエンティティを一覧表示するには

以下のコマンドでは、仮想 MFA デバイスのエンティティを一覧表示します。

- AWS CLI: [aws iam list-virtual-mfa-devices](#)
- AWS API: [ListVirtualMFADevices](#)

仮想 MFA デバイ스에 タグを付けるには

仮想 MFA デバイ스에 タグを付けるには、次のコマンドを使用します。

- AWS CLI: [aws iam tag-mfa-device](#)
- AWS API: [TagMFADevice](#)

仮想 MFA デバイスのタグを一覧表示するには

仮想 MFA デバイ스에 アタッチされたタグを一覧表示するには、次のコマンドを使用します。

- AWS CLI: [aws iam list-mfa-device-tags](#)
- AWS API: [ListMFADeviceTags](#)

仮想 MFA デバイスのタグを解除するには

仮想 MFA デバイスにアタッチされたタグを削除するには、次のコマンドを使用します。

- AWS CLI: [aws iam untag-mfa-device](#)
- AWS API: [UntagMFADevice](#)

MFA デバイスを再同期するには

AWS が受け入れられないコードをデバイスが生成している場合は、以下のコマンドを使用します。デバイスが仮想デバイスの場合、仮想デバイスの ARN をシリアルナンバーとして使用します。

- AWS CLI: [aws iam resync-mfa-device](#)
- AWS API: [ResyncMFADevice](#)

IAM で仮想 MFA デバイスのエンティティを削除するには

デバイスのユーザーへの関連付けを解除した後、そのデバイスのエンティティを削除できます。

- AWS CLI: [aws iam delete-virtual-mfa-device](#)
- AWS API: [DeleteVirtualMFADevice](#)

紛失または故障中の仮想 MFA デバイスを復旧するには


仮想化 MFA アプリをホストするユーザーのデバイスが紛失、交換、または機能しなくなることがあります。この場合、ユーザーが自分で復旧することはできません。ユーザーは、デバイスを無効にするために管理者に連絡する必要があります。詳細については、「[MFA デバイスの紛失および故障時の対応](#)」を参照してください。

## MFA ステータスのチェック

IAM コンソールで、AWS アカウントのルートユーザー または IAM ユーザーで有効な MFA デバイスが有効化されているかどうかを確認します。

ルートユーザーの MFA ステータスを確認するには


1. ルートユーザー認証情報を使用して AWS Management Console にサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。
3. [Multi-factor Authentication (MFA)] で、MFA が有効か無効かを確認します。MFA がアクティブ化されていない場合は、アラート記号  が表示されます。

アカウントに対して MFA を有効化する場合は、以下のいずれかを参照してください。

- [AWS アカウントのルートユーザー \(コンソール\) の仮想 MFA デバイスを有効にします](#)
- [AWS アカウントのルートユーザー のパスキーまたはセキュリティキーを有効にする \(コンソール\)](#)
- [AWS アカウントのルートユーザー 用にハードウェア TOTP トークンを有効にする \(コンソール\)](#)

IAM ユーザーの MFA ステータスを確認するには

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択します。
3. 必要に応じて、以下の手順を実行して、[MFA] 列を USERS テーブルに追加します。
  - a. 右端のテーブルの上で、設定アイコン  を選択します。
  - b. [Manage Columns (列の管理)] で、[MFA] を選択します。
  - c. (オプション) ユーザーテーブルに表示しない列見出しのチェックボックスをオフにします。
  - d. [Close (閉じる)] を選択して、ユーザーのリストに戻ります。
4. [MFA] 列に、有効になっている MFA デバイスが表示されます。そのユーザーにアクティブな MFA デバイスがない場合は、コンソールに [None] (なし) と表示されます。ユーザーの MFA デバイスが有効になっている場合は、MFA 列に、そのデバイスのタイプを示す値 ([Virtual] (仮想)、[Security Key] (セキュリティキー)、[Hardware] (ハードウェア)、または SMS) が表示されます。

**Note**

AWS は、間もなく SMS 多要素認証 (MFA) のサポートを終了します。SMS テキストメッセージベース MFA を使用する IAM ユーザーのお客様は、以下の別のいずれかの方法 ([仮想 \(ソフトウェアベースの\) MFA デバイス](#)、[FIDO セキュリティキー](#)、または [ハードウェア MFA デバイス](#)) に切り替えることをお勧めします。割り当てられた SMS MFA デバイスを使用して、アカウントのユーザーを識別することができます。これを行うには、IAM コンソールに移動して、ナビゲーションペインの [ユーザー] を選択し、テーブルの [MFA] 列の [SMS] を使用してユーザーを探します。

5. ユーザーの MFA デバイスに関する追加情報を表示するには、確認する MFA ステータスのユーザー名を選択します。次に、[認証情報] タブを選択します。
6. そのユーザーにアクティブな MFA デバイスがない場合は、コンソールに [MFA デバイスなし] と表示されます。[多要素認証 (MFA)] セクションで [MFA デバイスを割り当てて AWS 環境のセキュリティを強化します]。ユーザーの MFA デバイスが有効化されている場合、[Multi-factor authentication (MFA)] (多要素認証 (MFA)) セクションには、対象のデバイスに関する詳細が表示されます。
  - デバイス名
  - デバイスのタイプ
  - デバイスの ID (物理デバイスのシリアル番号、または AWS での仮想デバイスの ARN)
  - デバイスの作成日時

デバイスを削除または再同期するには、そのデバイスの横にあるラジオボタンをオンにし、[Remove] (削除) または [Resync] (再同期) を選択します。

MFA の有効化の詳細については、以下を参照してください。

- [仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)
- [パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)
- [ハードウェア TOTP トークンの有効化 \(コンソール\)](#)

## 仮想デバイスとハードウェア MFA デバイスの再同期

AWS コンソールを使用して、仮想およびハードウェア Multi-Factor Authentication (MFA) デバイスを再同期できます。ユーザーのデバイスが使用しようとしたときに同期されていない場合、ユーザーのサインインは失敗し、IAM はユーザーにデバイスの再同期を促します。

### Note

FIDO セキュリティキーは同期しなくなることはありません。FIDO セキュリティキーが紛失または破損した場合は、それを非アクティブにすることができます。MFA デバイスタイプを無効にする方法については、「[別の IAM ユーザーの MFA デバイスを無効化するには \(コンソール\)](#)」を参照してください。

AWS 管理者は、IAM ユーザーの仮想およびハードウェア MFA デバイスがシステムと同期されていない場合、それらのデバイスを再同期できます。

AWS アカウントのルートユーザー MFA デバイスが動作していない場合は、IAM コンソールを使用してデバイスを再同期することができます。デバイスを正常に再同期できない場合は、デバイスの関連付けを解除して再関連付けする必要がある場合があります。方法の詳細については、「[MFA デバイスの無効化](#)」および「[AWS でのユーザーの MFA デバイスの有効化](#)」を参照してください。

### トピック

- [必要なアクセス許可](#)
- [仮想およびハードウェア MFA デバイスの再同期 \(IAM コンソール\)](#)
- [仮想およびハードウェア MFA デバイスの再同期 \(AWS CLI\)](#)
- [仮想およびハードウェア MFA デバイスの再同期 \(AWS API\)](#)

### 必要なアクセス許可

独自の IAM ユーザーの仮想またはハードウェア MFA デバイスを再同期するには、次のポリシーのアクセス許可が必要です。このポリシーでは、デバイスを作成することや非アクティブ化することはできません。

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```
{
  "Sid": "AllowListActions",
  "Effect": "Allow",
  "Action": [
    "iam:ListVirtualMFADevices"
  ],
  "Resource": "*"
},
{
  "Sid": "AllowUserToViewAndManageTheirOwnUserMFA",
  "Effect": "Allow",
  "Action": [
    "iam:ListMFADevices",
    "iam:ResyncMFADevice"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "BlockAllExceptListedIfNoMFA",
  "Effect": "Deny",
  "NotAction": [
    "iam:ListMFADevices",
    "iam:ListVirtualMFADevices",
    "iam:ResyncMFADevice"
  ],
  "Resource": "*",
  "Condition": {
    "BoolIfExists": {
      "aws:MultiFactorAuthPresent": "false"
    }
  }
}
]
```

## 仮想およびハードウェア MFA デバイスの再同期 (IAM コンソール)

IAM コンソールを使用して、仮想およびハードウェア MFA デバイスを再同期できます。

独自の IAM ユーザーの仮想またはハードウェア MFA デバイスを再同期するには (コンソール)

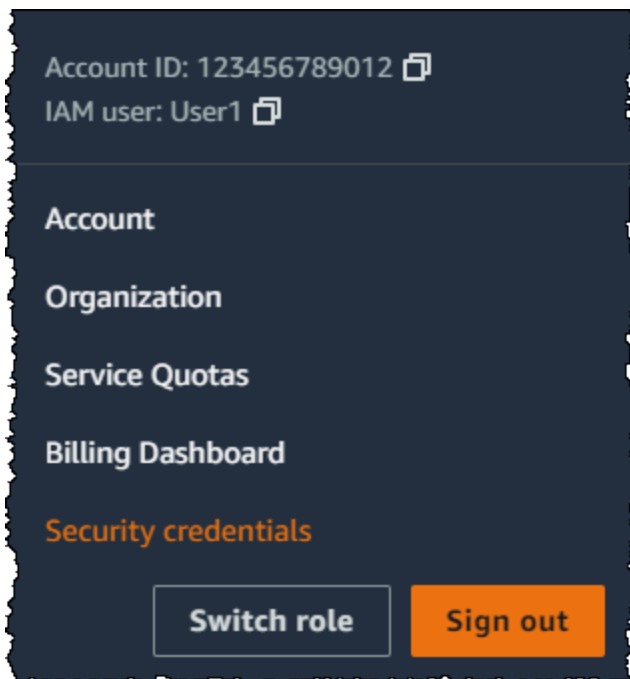
1. AWS アカウント ID またはアカウントエイリアス、IAM ユーザー名、およびパスワードを使用して [IAM コンソール](#) にサインインします。

**Note**

利便性のため、AWS サインインページは、ブラウザ cookie を使用して IAM ユーザー名とアカウント情報を記憶します。以前に別のユーザーとしてサインインしたことがある場合は、ページの下部にある[別のアカウントにサインイン]を選択し、メインのサインインページに戻ります。そこから、AWS アカウント ID またはアカウントエイリアスを入力して、アカウントの IAM ユーザーサインインページにリダイレクトされるようになります。

AWS アカウント アカウント ID の取得については、管理者にお問い合わせください。

2. 右上のナビゲーションバーで自分のユーザー名を選択し、続いて [Security credentials] (セキュリティ認証情報) を選択します。



3. [AWS IAM 認証情報] タブの [多要素認証 (MFA)] セクションで、MFA デバイスの横にあるラジオボタンを選択し、[再同期] を選択します。
4. デバイスで連続して生成された次の 2 つのコードを [MFA コード 1] および [MFA コード 2] に入力します。次に、[Resync] (再同期) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、リクエスト処理中に見えますが、デバイスは同期されていません。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。

別の IAM ユーザーの仮想またはハードウェア MFA デバイスを再同期するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択してから、MFA デバイスを再同期する必要があるユーザーの名前を選択します。
3. [ Security credentials ] タブを選択します。[多要素認証 (MFA)] セクションで、MFA デバイスの横にあるラジオボタンを選択し、[再同期] を選択します。
4. デバイスで連続して生成された次の 2 つのコードを [MFA コード 1] および [MFA コード 2] に入力します。次に、[Resync] (再同期) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、リクエスト処理中に見えますが、デバイスは同期されていません。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いために起こります。

サインインする前に ルートユーザーMFA を再同期するには (コンソール)

1. [Amazon Web Services Sign In With Authentication Device (認証デバイスによる Amazon Web Services へのサインイン)] ページで、[認証デバイスに問題がありますか?] を選択します。[Click here] (ここをクリックしてください)。

**Note**

他のテキスト (例: MFA を使用してサインイン や 認証デバイスのトラブルシューティング) が表示される場合があります。ただし、提供されている機能は同じです。

2. [Re-Sync With Our Servers (サーバーとの再同期)] セクションで、デバイスで連続して生成された次の 2 つのコードを [MFA code 1 (MFA コード 1)] および [MFA code 2 (MFA コード 2)] に入力します。次に、[Re-sync authentication device (認証デバイスの再同期)] を選択します。
3. 必要に応じて、パスワードをもう一度入力し、[サインイン] を選択します。次に、MFA デバイスを使用してサインインを完了します。

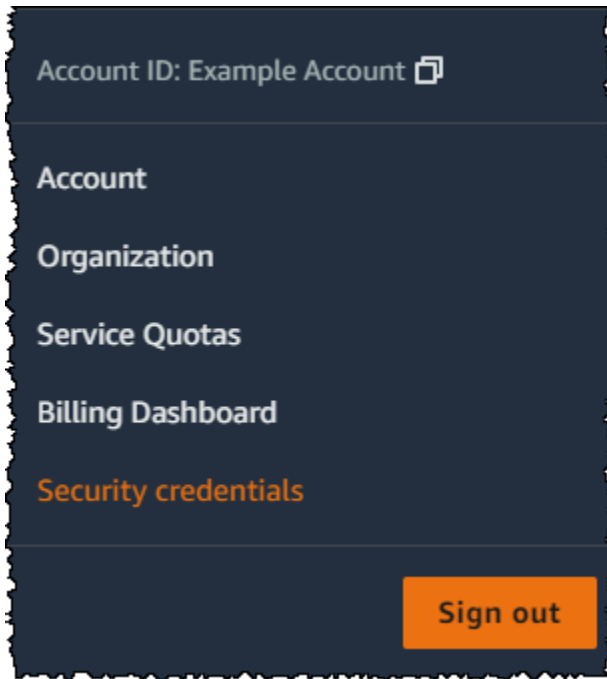
サインインした後に ルートユーザーMFA デバイスを再同期するには (コンソール)

1. Root user (ルートユーザー) を選択して AWS アカウント の E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

**Note**

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[IAM ユーザーのサインイン] ページが表示された場合、ページ下部付近で [ルートユーザーの電子メールを使用してサインインする] を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

2. ナビゲーションバーの右側で、使用するアカウント名を選択し、次に [Security Credentials] (セキュリティ認証情報) を選択します。必要に応じて、[Continue to Security credentials] (セキュリティ認証情報に進む) を選択します。



3. ページの [Multi-Factor Authentication (MFA)] セクションを展開します。
4. デバイスの横にあるラジオボタンを選択して、[Resync] (再同期) を選択します。
5. [Resync MFA device] (MFA デバイスを再同期) ダイアログボックスで、デバイスで連続して生成された次の 2 つのコードを [MFA code 1] (MFA コード 1) および [MFA code 2] (MFA コード 2) に入力します。次に、[Resync] (再同期) を選択します。

**⚠ Important**

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合、MFA デバイスはユーザーとは正常に関連付けられませんが、その MFA デバイスは同期しません。これは、タイムベースドワンタイムパスワード (TOTP) の有効期間が短いため起こります。

### 仮想およびハードウェア MFA デバイスの再同期 (AWS CLI)

AWS CLI から仮想およびハードウェア MFA デバイスを再同期できます。

IAM ユーザーの仮想 MFA デバイスまたはハードウェア MFA デバイスを再同期するには (AWS CLI)

コマンドプロンプトで、[aws iam resync-mfa-device](#) コマンドを発行します。

- 仮想 MFA デバイス: デバイスの Amazon リソースネーム (ARN) をシリアル番号として入力します。

```
aws iam resync-mfa-device --user-name Richard --serial-number  
arn:aws:iam::123456789012:mfa/RichardsMFA --authentication-code1 123456 --  
authentication-code2 987654
```

- ハードウェア MFA デバイス: ハードウェアデバイスのシリアル番号をシリアル番号として指定します。形式はベンダー固有です。例えば、Amazon から gemalto トークンを購入できます。シリアル番号は通常、4 つの文字の後に 4 つの数字が続きます。

```
aws iam resync-mfa-device --user-name Richard --serial-number ABCD12345678 --  
authentication-code1 123456 --authentication-code2 987654
```

#### Important

コードを生成したら、即時にリクエストを送信します。コードを生成した後にリクエストを送信するまで時間がかかりすぎる場合は、コードの有効期間が短いためリクエストは送信されません。

## 仮想およびハードウェア MFA デバイスの再同期 (AWS API)

IAM には同期を実行する API コールがあります。この場合、仮想およびハードウェア MFA デバイスユーザーにこの API コールに対するアクセス許可を付与することをお勧めします。API コールに基づいてツールを構築して、ユーザーが必要に応じてデバイスを再同期できるようにします。

IAM ユーザーの仮想またはハードウェア MFA デバイスを再同期するには (AWS API)

- [ResyncMFADevice](#) リクエストを送信します。

## MFA デバイスの無効化

IAM ユーザーとして多要素認証 (MFA) デバイスにサインインできない場合は、管理者に連絡してください。

管理者として、他の IAM ユーザー用に端末を無効にすることができます。これにより、ユーザーは MFA を使用しないでサインインできます。MFA デバイスが置き換えられる際、またはデバイスが

時的に使用できない時に、一時的な対処方法として行われる場合があります。ただし、ユーザーの新しいデバイスを可能な限り速やかに有効にすることをお勧めします。MFA デバイスを有効化する方法については [the section called “MFA デバイスの有効化”](#) を参照してください。

#### Note

API または AWS CLI を使用して AWS アカウント からユーザーを削除する場合は、ユーザーの MFA デバイスを無効化または削除する必要があります。この変更は、ユーザーを削除するプロセスの一環として行います。ユーザーの削除についての詳細は、[IAM ユーザーの管理](#) を参照してください。

## トピック

- [MFA デバイスの無効化 \(コンソール\)](#)
- [MFA デバイスの無効化 \(AWS CLI\)](#)
- [MFA デバイスの無効化 \(AWS API\)](#)

## MFA デバイスの無効化 (コンソール)

別の IAM ユーザーの MFA デバイスを無効化するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. ユーザーの MFA デバイスを非アクティブ化するには、削除する MFA を持つユーザーの名前を選択します。
4. [ Security credentials ] タブを選択します。
5. [多要素認証 (MFA)] で MFA デバイスの横にあるラジオボタンを選択し、[削除] を選択し、さらにまた [削除] を選択します。

デバイスは AWS から削除されます。このデバイスは、再び有効化されて AWS ユーザーや AWS アカウントのルートユーザーに関連付けられるまで、サインインやリクエストの認証に使用できなくなります。



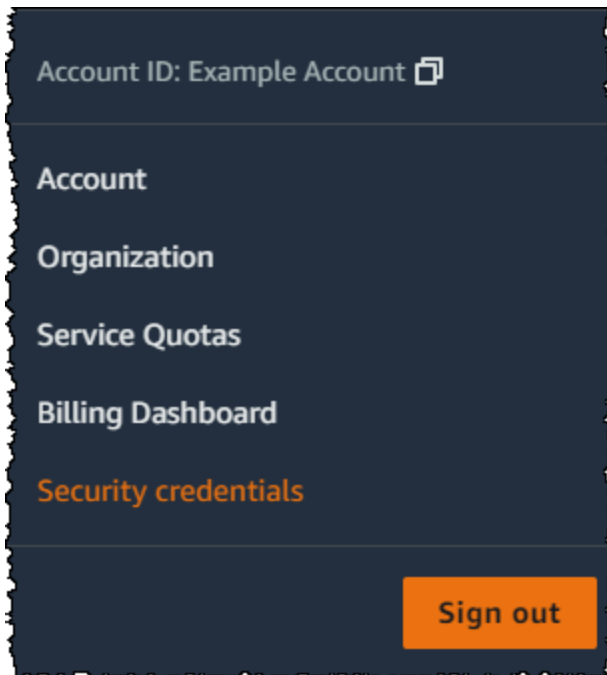
## AWS アカウントのルートユーザー の MFA デバイスを無効にするには (コンソール)

1. Root user (ルートユーザー) を選択して AWS アカウント の E メールアドレスを入力し、アカウント所有者として [IAM コンソール](#) にサインインします。次のページでパスワードを入力します。

**Note**

ルートユーザーでは、「IAM ユーザーとしてサインイン」ページにサインインすることはできません。[IAM ユーザーのサインイン] ページが表示された場合、ページ下部付近で [ルートユーザーの電子メールを使用してサインインする] を選択します。ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「[ルートユーザーとして AWS Management Console へサインインする](#)」を参照してください。

2. ナビゲーションバーの右側で、使用するアカウント名を選択し、次に [Security Credentials] (セキュリティ認証情報) を選択します。必要に応じて、[Continue to Security credentials] (セキュリティ認証情報に進む) を選択します。



3. [Multi-factor authentication (MFA)] (多要素認証 (MFA)) セクションで、無効化する MFA デバイスの横にあるラジオボタンをオンにし、[Remove] (削除) を選択します。
4. [削除] を選択します。

AWS アカウント 向けの MFA デバイスの無効化が完了しました。AWS アカウント に関連付けられている E メールを Amazon Web Services からの確認メッセージで確認します。この E メールでは、Amazon Web Services の Multi-Factor Authentication (MFA) が無効化されたことを通知します。メッセージは @amazon.com または @aws.amazon.com から送信されます。

### MFA デバイスの無効化 (AWS CLI)

IAM ユーザーの MFA デバイスを無効化するには (AWS CLI)

- 次のコマンドを実行します。 [aws iam deactivate-mfa-device](#)

### MFA デバイスの無効化 (AWS API)

IAM ユーザーの MFA デバイスを無効化するには (AWS API)

- 呼び出すオペレーション: [DeactivateMFADevice](#)

## MFA デバイスの紛失および故障時の対応

[仮想 MFA デバイス](#) または [ハードウェア TOTP トークン](#) が正常に機能しているようでも、AWS リソースへのアクセスに使用できない場合は、AWS と同期していない可能性があります。仮想またはハードウェア MFA デバイスの同期については、「[仮想デバイスとハードウェア MFA デバイスの再同期](#)」を参照してください。[FIDO セキュリティキー](#) は同期しなくなることはありません。

AWS アカウントのルートユーザー [多要素認証 \(MFA\) デバイス](#) が紛失したり、破損したり、または機能しない場合は、アカウントへのアクセスを回復できます。IAM ユーザーは、デバイスを無効にするために管理者に連絡する必要があります。

#### Important

MFA デバイスを紛失したりアクセスできなくなったりした場合でも、IAM ユーザー用に複数の MFA デバイスを有効にして、アカウントに引き続きアクセスできるようにすることをお勧めします。AWS アカウント ルートユーザーと IAM ユーザーには、現在サポートされている MFA タイプを任意に組み合わせた最大 8 台の MFA デバイスを登録できます。

## ルートユーザー MFA デバイスの回復

AWS アカウントのルートユーザー [\[多要素認証 \(MFA\) デバイス\]](#)が紛失したり、破損したり、機能しなかったりする場合は、同じ AWS アカウントのルートユーザー に登録された別の MFA デバイスを使用してサインインできます。ルートユーザーの MFA デバイスが 1 つだけ有効になっている場合は、代替の認証方法を使用できます。つまり、MFA デバイスでサインインできない場合は、アカウントに登録されている E メールと主要連絡先の電話番号を使用して ID を確認してサインインすることができます。

代替の認証要素を使用してルートユーザーとしてサインインするには、アカウントに関連付けられている E メールと主要連絡先の電話番号にアクセスできる必要があります。主要連絡先の電話番号を更新する必要がある場合は、ルートユーザーではなく、管理者アクセス権を持つ IAM ユーザーとしてサインインすれば可能です。アカウント連絡先情報の更新に関するその他の手順については、AWS Billing ユーザーガイドの「[連絡先情報の編集](#)」を参照してください。E メールと主要連絡先の電話番号にアクセスできない場合は、[AWS Support](#) に連絡する必要があります。

### Important

アカウントを正常に回復するために、ルートユーザーにリンクされている E メールアドレスと連絡先電話番号を最新の状態に保つことをお勧めします。詳細については、「AWS Account Management リファレンスガイド」の「[AWS アカウントの主要連絡先の更新](#)」を参照してください。

認証の代替要因を AWS アカウントのルートユーザー として使用してログインするには

1. [ルートユーザー] を選択し、AWS アカウント のメールアドレスを入力して、アカウント所有者として [AWS Management Console](#) にサインインします。次のページでパスワードを入力します。
2. 「追加の検証が必要」ページで、認証に使用する MFA メソッドを選択し、[次へ] を選択します。

### Note

次のような代替テキストが表示される場合があります。MFA を使用してサインイン、認証デバイスのトラブルシューティング、または MFA のトラブルシューティングですが、機能は同じです。いずれの場合も、認証の代替要因を使用してアカウントの E メール

ルアドレスと主要連絡先の電話番号を確認できない場合は、[AWS Support](#) に連絡して、MFA デバイスを非アクティブ化するように依頼する必要があります。

3. 使用している MFA の種類に応じて表示されるページは異なりますが、[MFA のトラブルシューティング] オプションの機能は同じです。[追加の検証が必要] ページまたは「[多要素認証] ページで、[MFA のトラブルシューティング] を選択します。
4. 必要に応じて、パスワードをもう一度入力し、[サインイン] を選択します。
5. [認証デバイスのトラブルシューティング] ページの [別の認証要素を使用してサインイン] セクションで [別の要素を使用してサインイン] を選択します。
6. 「別の認証要素を使用してサインイン」 ページで、E メールアドレスの確認を行ってアカウントを認証します。[確認 E メールを送信] を選択します。
7. AWS アカウントに関連付けられているメールで、Amazon ウェブサービス (recover-mfa-no-reply@verify.signin.aws) からのメッセージを確認します。E メールの指示にしたがって操作します。

アカウントに E メールが表示されない場合は、迷惑メールフォルダを確認するか、ブラウザに戻り、[Resend the email (E メールの再送信)] を選択します。

8. E メールアドレスを確認した後も、アカウントの認証を続けることができます。主要連絡先の電話番号を確認するには、[Call me now] (すぐに連絡を受ける) を選択します。
9. AWS からの呼び出しに応答し、プロンプトが表示されたら、AWS ウェブサイトの電話番号の 6 桁の番号を入力します。

AWS からの呼び出しを受けない場合は、[サインイン] を選択してコンソールに再度サインインし、やり直してください。または、「[Lost or unusable Multi-Factor Authentication \(MFA\) device](#)」(多要素認証 (MFA) の紛失または使用不可) を参照して、サポートにお問い合わせください。

10. 電話番号を確認した後、[Sign in to the console (コンソールにサインイン)] を選択してアカウントにサインインすることができます。
11. 次の手順は、使用している MFA のタイプによって異なります。
  - 仮想 MFA デバイスの場合は、デバイスからアカウントを削除します。次に、[AWS セキュリティ認証情報](#) ページにアクセスし、古い仮想 MFA デバイスのエンティティを削除してから、新しいキーペアを作成することができます。
  - FIDO セキュリティキーの場合は、[AWS Security Credentials](#) ページに移動し、新しい FIDO セキュリティキーを有効にする前に古い FIDO セキュリティキーを非アクティブにします。

- ハードウェア TOTP トークンの場合は、デバイスの修理または交換についてサードパーティープロバイダーに問い合わせてください。新しいデバイスを受け取るまで、認証の代替要因を使用してサインインを続けることができます。新しいハードウェア MFA デバイスを入手したら、[AWS セキュリティ認証情報](#) ページに移動し、新しいものを作成する前に古いハードウェア MFA デバイスエンティティを削除します。

#### Note

紛失または盗難された MFA デバイスを同じタイプのデバイスで置き換える必要はありません。例えば、FIDO セキュリティキーを破棄して新しいキーを注文すると、新しい FIDO セキュリティキーを受け取るまで、仮想 MFA またはハードウェア TOTP トークンを使用できます。

#### Important

MFA デバイスを紛失したか、盗難に遭った場合は、別の認証要素を使用してサインインし、代替の MFA デバイスを確立した後、攻撃者が認証デバイスを盗み、現在のパスワードも持っている可能性がある場合に備えて、root ユーザーのパスワードを変更してください。詳細については、「AWS Account Management リファレンスガイド」の「[AWS アカウントのルートユーザーのパスワードを変更する](#)」を参照してください。

## IAM ユーザー MFA デバイスの回復

IAM ユーザーである場合、デバイスが紛失したり機能しなくなったりしても、自分でデバイスを回復することはできません。管理者に連絡して、デバイスを非アクティブ化するように依頼する必要があります。その後、新しいデバイスを有効にすることができます。

IAM ユーザーとしての MFA デバイスに関するヘルプ情報を入手するには

- お使いの IAM ユーザーのユーザー名やパスワードの設定を行った AWS システム管理者あるいは別の担当者にご連絡ください。管理者は、サインインできるように「[MFA デバイスの無効化](#)」の説明に従って、MFA デバイスを非アクティブ化する必要があります。
- 次の手順は、使用している MFA のタイプによって異なります。

- 仮想 MFA デバイスの場合は、デバイスからアカウントを削除します。次に、「[仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)」の説明に従って、仮想デバイスを有効にします。
- FIDO セキュリティキーの場合は、デバイスの交換についてサードパーティープロバイダーに問い合わせてください。新しい FIDO セキュリティキーを受け取ったら、「[パスキーまたはセキュリティキーの有効化 \(コンソール\)](#)」で説明されているとおりに有効にします。
- ハードウェア TOTP トークンの場合は、デバイスの修理または交換についてサードパーティープロバイダーに問い合わせてください。新しい物理 MFA デバイスを入手したら、「[ハードウェア TOTP トークンの有効化 \(コンソール\)](#)」の説明に従ってデバイスを有効にします。

#### Note

紛失または盗難された MFA デバイスを同じタイプのデバイスで置き換える必要はありません。任意の組み合わせの MFA デバイスを最大で 8 台用意できます。例えば、FIDO セキュリティキーを破棄して新しいキーを注文すると、新しい FIDO セキュリティキーを受け取るまで、仮想 MFA またはハードウェア TOTP トークンを使用できます。

3. MFA デバイスを紛失または盗難にあった場合は、アタッカーがその認証デバイスから現在のパスワードを入手している可能性があるため、パスワードを変更してください。詳細については、[IAM ユーザーのパスワードの管理](#) を参照してください。

## MFA 保護 API アクセスの設定

IAM ポリシーを使用して、ユーザーが呼び出すことができる API オペレーションを指定できます。場合によっては、ユーザーが特に重要なアクションを実行することを許可する前に、このユーザーに AWS 多要素認証 (MFA) で認証することを求める追加のセキュリティが必要となることもあります。

たとえば、ユーザーに Amazon EC2 RunInstances、DescribeInstances、および StopInstances アクションの実行を許可するポリシーがすでに存在する可能性があります。ただし TerminateInstances のような有害なアクションを制限し、AWS MFA デバイスによって認証される場合に限り、ユーザーがそのアクションを実行できるように管理することもできます。

### トピック

- [概要](#)
- [シナリオ: クロスアカウントの委任の MFA 保護](#)



- [シナリオ: 現在のアカウントの API オペレーションへのアクセスに対する MFA 保護](#)
- [シナリオ: リソースベースのポリシーを持つリソースの MFA 保護](#)

## 概要

API オペレーションに MFA 保護を追加する場合、次のタスクが関連します:

1. 管理者は、MFA 認証を必要とする API リクエストを行う必要がある各ユーザーに対し、AWS MFA デバイスを設定します。このプロセスは、[AWS でのユーザーの MFA デバイスの有効化](#) で説明されています。
2. 管理者はユーザーに対し、ユーザーが AWS MFA デバイスで認証されているかどうかを確認する Condition 要素を含むポリシーを作成します。
3. ユーザーは、後述の MFA 保護のシナリオに応じて、MFA パラメータをサポートする AWS STS API オペレーションである [AssumeRole](#) または [GetSessionToken](#) を呼び出します。この呼び出しの一部としてユーザーは、このユーザーに関連付けられるデバイスのデバイス識別子を含めます。また、ユーザーは、デバイスが生成する時間ベースのワンタイムパスワード (TOTP) も含めます。そのつど、ユーザーは一時的な認証情報を取り戻し、その情報を使用して AWS に追加リクエストを行うことができます。

### Note

サービスの API オペレーションの MFA 保護は、サービスが一時的なセキュリティ認証情報をサポートする場合にのみ使用できます。サポートするサービスの一覧については、「[一時的セキュリティ認証情報を使用して AWS にアクセスする](#)」を参照してください。

承認が失敗した場合、AWS は他の承認されていないアクセスと同様に "Access Denied" というエラーメッセージを返します。MFA 保護 API ポリシーが存在する場合、ユーザーが有効な MFA 認証なしで API オペレーションを呼び出す場合に、AWS はポリシーで指定される API オペレーションへのアクセスを拒否します。また、API オペレーションへのリクエストのタイムスタンプがポリシーで指定される許可範囲外である場合にも、そのオペレーションは拒否されます。ユーザーは、MFA コードとデバイスのシリアルナンバーを使って新しい一時的な認証情報をリクエストすることで、再度、MFA の認証を受ける必要があります。

## MFA 条件を指定した IAM ポリシー

MFA 条件を指定したポリシーは、次にアタッチすることができます:



- IAM ユーザーまたはグループ
- Amazon S3 バケット、Amazon SQS キュー、または Amazon SNS トピックなどのリソース
- ユーザーが引き受けることのできる IAM ロールの信頼ポリシー

ポリシーの MFA 条件を使用して、次のプロパティを確認することができます。

- 存在 `aws:MultiFactorAuthPresent` ユーザーが MFA を使って認証されたことを単に確認するには、条件で `True` キーが `Bool` であることを確認します。ユーザーが短期的な認証情報で認証した場合に限りキーが表示されます。アクセスキーのような長期的な認証情報に、このキーは含まれません。
- 期間 - MFA 認証後、指定された期間内のみアクセス権を与える場合、数値条件タイプを使用して `aws:MultiFactorAuthAge` キーの有効期間を値 (3600 秒など) と比較します。MFA が使用されなかった場合、`aws:MultiFactorAuthAge` キーは存在しません。

以下の例は、MFA 認証の存在をテストする MFA 条件が含まれる、IAM ロールの信頼ポリシーを示します。このポリシーにより、Principal 要素 (ACCOUNT-B-ID を有効な AWS アカウント ID に置き換えます) で指定された AWS アカウントのユーザーは、このポリシーがアタッチされたロールを引き受けることができます。ただし、このようなユーザーは、MFA を使用して認証されたユーザーの場合のみ、ロールを引き受けることができます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

MFA の条件種別の詳細については、「[AWS グローバル条件コンテキストキー](#)」、「[数値条件演算子](#)」、および「[条件キーの有無をチェックする条件演算子](#)」を参照してください。

GetSessionToken または AssumeRole を選択する

AWS STS には、ユーザーが MFA 情報を渡す際に使用できる 2 つの API オペレーションである GetSessionToken と AssumeRole があります。ユーザーが一時的な認証情報を取得するために呼び出す API オペレーションは、以下のどのシナリオがあてはまるかによって異なります。

以下のシナリオでは、**GetSessionToken** を使用します。

- リクエストを作成する IAM ユーザーと同じ AWS アカウントのリソースにアクセスする API オペレーションの呼び出し。GetSessionToken リクエストによる一時的な認証情報で IAM および AWS STS API にアクセスできるのは、認証情報のリクエストに MFA 情報を含めた場合のみである点に注意してください。GetSessionToken によって返される一時的な認証情報には MFA 情報が含まれているので、認証情報によって実行される各 API オペレーションで MFA を確認できます。
- リソースに基づくポリシー (MFA 条件を含む) で保護されているリソースへのアクセス。

GetSessionToken オペレーションの目的は、MFA を使用してユーザーを認証することです。認証オペレーションを制御するためにポリシーを使用することはできません。

以下のシナリオでは、**AssumeRole** を使用します。

- 同じまたは異なる AWS アカウントのリソースにアクセスする API オペレーションの呼び出し。API コールには、任意の IAM または AWS STS API を含めることができます。アクセスを保護するには、ユーザーがロールを引き受けた時点で MFA を強制する必要があります。AssumeRole によって返される一時的な認証情報のコンテキストには MFA 情報が含まれていないので、MFA に対する個別の API オペレーションを確認できません。このため、リソースベースのポリシーで保護されたリソースへのアクセスを制限するために、GetSessionToken を使用する必要があります。

これらのシナリオの実行方法に関する詳細は、本書で後から提供されます。

MFA 保護された API アクセスについて重要な点

API オペレーションの MFA 保護の次の点に注意してください:

- MFA 保護は、一時的なセキュリティ認証情報を使用した場合のみ利用できます。この認証情報は、AssumeRole または GetSessionToken を使用して取得する必要があります。
- AWS アカウントのルートユーザー 認証情報で MFA 保護 API アクセスを使用することはできません。
- U2F セキュリティキーで MFA 保護 API アクセスを使用することはできません。
- フェデレーションユーザーに AWS サービス利用のために MFA デバイスを割り当てることはできませんので、そのようなユーザーは MFA が管理する AWS リソースへアクセスできません。(次のポイントを参照してください。)

- 一時的な認証情報を返す他の AWS STS API オペレーションは、MFA をサポートしていません。AssumeRoleWithWebIdentity と AssumeRoleWithSAML の場合、ユーザーは外部プロバイダーによって認証されており、プロバイダーが MFA を要求したかどうかを AWS が判断することはできません。GetFederationToken については、MFA は必ずしも特定のユーザーに関連付けられていません。
- 同様に、長期的な認証情報 (IAM ユーザーアクセスキーと ルートユーザーアクセスキー) は失効しないため、MFA 保護 API アクセスでは使用できません。
- AssumeRole と GetSessionToken は、MFA 情報がない状態でも呼び出すことができます。この場合、発信者は一時的な認証情報を取り戻しますが、その一時的認証情報のセッション情報では、ユーザーが MFA を使用して認証されたことが示されません。
- API オペレーションに対する MFA 保護を確立するには、ポリシーに MFA 条件を追加します。MFA の使用を適用するには、ポリシーに `aws:MultiFactorAuthPresent` 条件キーが含まれている必要があります。クロスアカウントの委任では、ロールの信頼ポリシーに条件キーが含まれている必要があります。
- 別の AWS アカウント に自分のアカウントのリソースへのアクセスを許可する場合、リソースのセキュリティは、信頼されたアカウント (つまりお客様のアカウントではない他のアカウント) の設定によって決まります。これは、多要素認証を要求する場合にも当てはまります。仮想 MFA デバイスを作成するアクセス許可がある、信頼されるアカウント内のアイデンティティは、ロールの信頼ポリシーのその部分を満たすために MFA クレームを作成できます。他のアカウントのメンバーに多要素認証を必要とする自身の AWS リソースへのアクセスを許可する前に、信頼されるアカウントの所有者がセキュリティのベストプラクティスを取り入れていることを確認する必要があります。たとえば、信頼されるアカウントは、MFA デバイス管理 API オペレーションなどの機密性の高い API オペレーションへのアクセスを指定する信頼できるアイデンティティに制限する必要があります。
- ポリシーに MFA の条件が含まれている場合、ユーザーが MFA 認証済みでないか、無効な MFA デバイス識別子または無効な TOTP を入力すると、リクエストは拒否されます。

### シナリオ: クロスアカウントの委任の MFA 保護

このシナリオでは、ユーザーが AWS MFA デバイスを使用して認証された場合にのみ、別のアカウントの IAM ユーザーにアクセスを委任します。クロスアカウントの委任に関する詳細については、「[ロールに関する用語と概念](#)」を参照してください。

アカウント A (アクセス対象のリソースを所有している信頼するアカウント) があり、このアカウントに管理者権限を持つ IAM ユーザー Anaya がいると仮定します。Anaya は、アカウント B (信頼さ

れたアカウント) のユーザー Richard にアクセス許可を付与するつもりですが、Richard がロールを引き受ける前に、MFA によって認証されていることを確認する必要があります。

1. 信頼するアカウント A で、Anaya は CrossAccountRole という IAM ロールを作成し、ロールの信頼ポリシーのプリンシパルをアカウント B のアカウント ID に設定します。この信頼ポリシーは、AWS STS AssumeRole アクションに対するアクセス許可を付与します。Anaya は、次の例に示すように、信頼ポリシーに MFA の条件も追加します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

2. Anaya は、ロールが実行できることを指定するアクセス許可ポリシーを、ロールに追加します。MFA 保護を使用するロールのアクセス許可ポリシーは、他のロールアクセス許可ポリシーと同じです。次の例では、Anaya がこのロールに追加するポリシーを示しています。これによって、引き受けるユーザーがアカウント A のテーブル Books で任意の Amazon DynamoDB アクションを実行することを許可します。また、このポリシーは、コンソールでアクションを実行するときに必要な dynamodb:ListTables アクションも許可します。

#### Note

アクセス許可ポリシーに MFA 条件は含まれません。ユーザーがロールを引き受けることができるかどうかを決定するためにのみ MFA 認証が使用されていることに注意してください。ユーザーがロールを引き受けた場合、それ以上の MFA の確認は行われません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TableActions",
      "Effect": "Allow",
      "Action": "dynamodb:*",
      "Resource": "arn:aws:dynamodb:*:ACCOUNT-A-ID:table/Books"
    }
  ]
}
```

```
    },
    {
      "Sid": "ListTables",
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    }
  ]
}
```

3. 信頼されたアカウント B で管理者は、IAM ユーザー Richard が AWS MFA デバイスを使用して設定されていること、および Richard がデバイスの ID を知っていることを確認します。ハードウェア MFA デバイスの場合には、デバイス ID はシリアル番号であり、仮想 MFA デバイスの場合には、デバイス ID はデバイスの ARN です。
4. アカウント B で、管理者は、AssumeRole アクションを呼び出すことを許可する次のポリシーを、ユーザー Richard (または彼が属するグループ) にアタッチします。リソースは、Anaya がステップ 1 で作成したロールの ARN に設定されます。このポリシーに MFA 条件が含まれないことに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sts:AssumeRole"],
    "Resource": ["arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole"]
  }]
}
```

5. アカウント B で、Richard (または Richard が実行中のアプリケーション) が AssumeRole を呼び出します。API コールには、引き受けるロールの ARN (arn:aws:iam::*ACCOUNT-A-ID*:role/CrossAccountRole)、MFA デバイスの ID、Richard がデバイスから取得した現在の TOTP が含まれます。

Richard が AssumeRole を呼び出すと、AWS は Richard に有効な認証情報 (MFA の要件など) があるかどうかを確認します。その場合、Richard は正常にロールを引き受け、ロールの一時的な認証情報を使用して、アカウント A の Books という名前のテーブルで DynamoDB アクションを実行できます。

AssumeRole を呼び出すプログラムの例については、「[MFA 認証での AssumeRole の呼び出し](#)」を参照してください。

## シナリオ: 現在のアカウントの API オペレーションへのアクセスに対する MFA 保護

このシナリオでは、使用する AWS アカウント でユーザーが AWS MFA デバイスを使用して認証された場合のみ、機密性の高い API オペレーションにアクセスできるようにする必要があります。

アカウント A があり、そこに EC2 インスタンスの作業を必要としている開発者グループが存在すると仮定します。通常の開発者は、インスタンスの作業を実行できますが、`ec2:StopInstances` または `ec2:TerminateInstances` アクションへのアクセス許可を付与されていません。このような「有害な」特権的アクションをいくつかの信頼されたユーザーに制限するために、これらの機密性の高い Amazon EC2 アクションを許可するポリシーに MFA 保護を追加します。

このシナリオでは、信頼されたユーザーの 1 人がユーザー Sofia です。ユーザー Anaya は、アカウント A の管理者です。

1. Anaya は、Sofia が AWS MFA デバイスで設定されていること、および Sofia がこのデバイスの ID を知っていることを確認します。ハードウェア MFA デバイスの場合には、デバイス ID はシリアル番号であり、仮想 MFA デバイスの場合には、デバイス ID はデバイスの ARN です。
2. Anaya は EC2-Admins という名のグループを作成し、ユーザー Sofia をグループに追加します。
3. Anaya は、次のポリシーを EC2-Admins グループにアタッチします。このポリシーは、ユーザーが MFA を使用して認証された場合のみ、ユーザーに Amazon EC2 `StopInstances` アクションと `TerminateInstances` アクションを呼び出すアクセス許可を与えます。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

4.

### Note

このポリシーを有効にするために、ユーザーはまずサインアウトして、再度サインインする必要があります。



ユーザー Sofia が Amazon EC2 インスタンスを停止または終了する必要がある場合、Sofia (または Sofia が実行しているアプリケーション) は `GetSessionToken` を呼び出します。この API オペレーションは、MFA デバイスの ID と Sofia がデバイスから取得した現在の TOTP を渡します。

5. ユーザー Sofia (または Sofia が使用しているアプリケーション) は、`GetSessionToken` から提供される一時的な認証情報を使用して Amazon EC2 の `StopInstances` または `TerminateInstances` アクションを呼び出します。

`GetSessionToken` を呼び出すプログラムの例については、本書で後から説明する「[MFA 認証での `GetSessionToken` の呼び出し](#)」を参照してください。

### シナリオ: リソースベースのポリシーを持つリソースの MFA 保護

このシナリオでは、S3 バケット、SQS キュー、または SNS トピックの所有者であるとします。リソースにアクセスする任意の AWS アカウントの任意のユーザーが AWS MFA デバイスによって認証されていることを確認する必要があります。

このシナリオは、最初にユーザーにロールを引き受けることを要求せずに、クロスアカウント MFA 保護を提供する方法を説明します。この場合、ユーザーは次の 3 つの条件を満たしている場合にリソースにアクセスできます。これは、ユーザーが MFA によって認証される必要があること、`GetSessionToken` から一時的なセキュリティ認証情報を取得できること、および、リソースのポリシーによって信頼されているアカウントにということです。

アカウント A に存在し、S3 バケットを作成すると仮定します。さまざまな AWS アカウントに存在するユーザーが、MFA を使って認証されている場合にのみ、このバケットへのアクセスを許可します。

このシナリオでは、ユーザー Anaya はアカウント A の管理者です。ユーザー Nikhil は、アカウント C の IAM ユーザーです。

1. アカウント A で、Anaya が Account-A-bucket という名前のバケットを作成します。
2. Anaya はバケットにバケットポリシーを追加します。このポリシーは、アカウント A、アカウント B、またはアカウント C のユーザーに、バケット内の Amazon S3 `PutObject` および `DeleteObject` アクションの実行を許可します。ポリシーは、MFA 条件を含みます。

```
{
  "Version": "2012-10-17",
```



```
"Statement": [{
  "Effect": "Allow",
  "Principal": {"AWS": [
    "ACCOUNT-A-ID",
    "ACCOUNT-B-ID",
    "ACCOUNT-C-ID"
  ]},
  "Action": [
    "s3:PutObject",
    "s3>DeleteObject"
  ],
  "Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],
  "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
}]
}
```

### Note

Amazon S3 により MFA 削除機能が ルートアカウントアクセス (のみ) に提供されています。バケットのバージョンング状態を設定する際に、Amazon S3 MFA 削除機能を有効にできます。IAM ユーザーでは Amazon S3 MFA 削除機能を使うことはできず、また MFA 保護 API とは別のアクセス管理となっています。バケットの削除権限を持つ IAM ユーザーであっても、Amazon S3 MFA 削除機能を使用してバケットの削除を行うことはできません。Amazon S3 MFA 削除機能の詳細については、[MFA Delete](#) を参照してください。

3. アカウント C で管理者は、ユーザー Nikhil が AWS MFA デバイスを使用して設定されていること、および Nikhil がデバイスの ID を知っていることを確認します。ハードウェア MFA デバイスの場合には、デバイス ID はシリアル番号であり、仮想 MFA デバイスの場合には、デバイス ID はデバイスの ARN です。
4. アカウント C で、Nikhil (または Nikhil が実行中のアプリケーション) が `GetSessionToken` を呼び出します。呼び出しには、MFA デバイスの ID または ARN、および Nikhil がデバイスから取得する現在の TOTP が含まれます。
5. Nikhil (または Nikhil が使用中のアプリケーション) は、`GetSessionToken` から返された一時的な認証情報を使用して Amazon S3 の `PutObject` アクションを呼び出し、ファイルを `Account-A-bucket` にアップロードします。

`GetSessionToken` を呼び出すプログラムの例については、本書で後から説明する「[MFA 認証での `GetSessionToken` の呼び出し](#)」を参照してください。

**Note**

この場合、AssumeRole が返す一時的認証情報は機能しません。ユーザーはロールを引き受けるために MFA 情報を提供できますが、AssumeRole が返す一時的認証情報には MFA 情報は含まれません。この情報は、このポリシーの MFA 条件を満たすために必要です。

## サンプルコード: 多要素認証での認証情報のリクエスト

以下の例では、GetSessionToken と AssumeRole オペレーションを呼び出し、MFA 認証パラメータを渡す方法を示しています。GetSessionToken の呼び出しにはアクセス権限は必要ありませんが、AssumeRole を呼び出すポリシーがあることが必要です。返される認証情報は、アカウントのすべての S3 バケットを一覧表示するために使用されます。

### MFA 認証での GetSessionToken の呼び出し

以下の例は、GetSessionToken を呼び出し、MFA 認証情報を渡す方法を示しています。GetSessionToken オペレーションによって返される一時的な認証情報は、アカウントのすべての S3 バケットを一覧表示するために使用されます。

このコードを実行するユーザー (あるいはこのユーザーが含まれるグループ) にアタッチされるポリシーは、返された一時的な認証情報へのアクセス権限を提供します。このコード例のポリシーでは、Amazon S3 ListBuckets オペレーションをリクエストするアクセス許可をユーザーに付与しています。

以下のコード例は、GetSessionToken を使用する方法を示しています。

### CLI

#### AWS CLI

IAM ID のために短期間有効な認証情報のセットを取得するには

次の `get-session-token` コマンドは、呼び出しを実行する IAM ID のために短期間有効な認証情報のセットを取得します。結果として得られる認証情報は、ポリシーによって多要素認証 (MFA) が必要とされるリクエストのために使用できます。認証情報は生成されてから 15 分後に失効します。

```
aws sts get-session-token \
```

```
--duration-seconds 900 \  
--serial-number "YourMFADeviceSerialNumber" \  
--token-code 123456
```

出力:

```
{  
  "Credentials": {  
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",  
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",  
    "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4B1CFFxWNE10PTgk5TthT  
+FvwqnKwRc0IfrrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/  
IvU1dYUg2RVAJBanLiHb4IgrmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgkBN9bkUDNCJiBeb/  
AX1zBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wn0E49mR1/+0tkIKG07fAE",  
    "Expiration": "2020-05-19T18:06:10+00:00"  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetSessionToken](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 一定期間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。一時的な認証情報をリクエストするために使用される認証情報は、現在のシェルのデフォルトから推測されます。他の認証情報を指定するには、`-ProfileName` または `-AccessKey/-SecretKey` パラメータを使用します。

```
Get-STSSessionToken
```

出力:

```
AccessKeyId          Expiration  
SecretAccessKey      SessionToken  
-----  
-----
```

```
EXAMPLEACCESSKEYID           2/16/2015 9:12:28 PM
examplesecretaccesskey...     SamPleTokenN.....
```

例 2: 1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。リクエストを行うために使用される認証情報は、指定されたプロファイルから取得されます。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

出力:

```
AccessKeyId           Expiration
SecretAccessKey       SessionToken
-----
-----
EXAMPLEACCESSKEYID   2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleTokenN.....
```

例 3: プロファイル「myprofilename」で認証情報が指定されているアカウントに関連付けられた MFA デバイスの識別番号とデバイスから提供された値を使用して、1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

出力:

```
AccessKeyId           Expiration
SecretAccessKey       SessionToken
-----
-----
EXAMPLEACCESSKEYID   2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleTokenN.....
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetSessionToken](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

MFA トークンを渡してセッショントークンを取得し、それを使用してアカウントの Amazon S3 バケットを一覧表示します。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
      sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
    MFA
                               device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
    role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
```

```
print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetSessionToken](#)」を参照してください。

## MFA 認証での AssumeRole の呼び出し

以下の例は、AssumeRole を呼び出し、MFA 認証情報を渡す方法を示しています。AssumeRole によって返される一時的セキュリティ認証情報は、アカウントのすべての Amazon S3 バケットを一覧表示するために使用されます。

このシナリオの詳細については、「[シナリオ: クロスアカウントの委任の MFA 保護](#)」を参照してください。

以下のコード例は、AssumeRole を使用する方法を示しています。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
```

```
{
    /// <summary>
    /// This example shows how to use the AWS Security Token
    /// Service (AWS STS) to assume an IAM role.
    ///
    /// NOTE: It is important that the role that will be assumed has a
    /// trust relationship with the account that will assume the role.
    ///
    /// Before you run the example, you need to create the role you want to
    /// assume and have it trust the IAM account that will assume that role.
    ///
    /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
    /// for help in working with roles.
    /// </summary>

    private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

    static async Task Main()
    {
        // Create the SecurityToken client and then display the identity of
the
        // default user.
        var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

        var client = new
Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

        // Get and display the information about the identity of the default
user.
        var callerIdRequest = new GetCallerIdentityRequest();
        var caller = await client.GetCallerIdentityAsync(callerIdRequest);
        Console.WriteLine($"Original Caller: {caller.Arn}");

        // Create the request to use with the AssumeRoleAsync call.
        var assumeRoleReq = new AssumeRoleRequest()
        {
            DurationSeconds = 1600,
            RoleSessionName = "Session1",
            RoleArn = roleArnToAssume
        };

        var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);
```



```
        // Now create a new client based on the credentials of the caller
        assuming the role.
        var client2 = new AmazonSecurityTokenServiceClient(credentials:
        assumeRoleRes.Credentials);

        // Get and display information about the caller that has assumed the
        defined role.
        var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
        Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
    }
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[AssumeRole](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
```

```
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
# credentials.
#
# Parameters:
#     -n role_session_name -- The name of the session.
#     -r role_arn -- The ARN of the role to assume.
#
# Returns:
#     [access_key_id, secret_access_key, session_token]
#     And:
#     0 - If successful.
#     1 - If an error occurred.
#####
function sts_assume_role() {
    local role_session_name role_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function sts_assume_role"
        echo "Assumes a role in the AWS account and returns the temporary
credentials:"
        echo "  -n role_session_name -- The name of the session."
        echo "  -r role_arn -- The ARN of the role to assume."
        echo ""
    }

    while getopt n:r:h option; do
        case "${option}" in
            n) role_session_name=${OPTARG} ;;
            r) role_arn=${OPTARG} ;;
            h)
                usage
            )
        esac
    done
```

```
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done

response=$(aws sts assume-role \
  --role-session-name "$role_session_name" \
  --role-arn "$role_arn" \
  --output text \
  --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[AssumeRole](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[AssumeRole](#)」を参照してください。

## CLI

## AWS CLI

ロールを引き受けるには

次の `assume-role` コマンドは、IAM ロール `s3-access-example` のために短期間有効な認証情報のセットを取得します。

```
aws sts assume-role \  
  --role-arn arn:aws:iam::123456789012:role/xaccounts3access \  
  --role-session-name s3-access-example
```

出力:

```
{  
  "AssumedRoleUser": {  
    "AssumedRoleId": "ARO3XFRBF535PLBIFPI4:s3-access-example",  
    "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-  
access-example"  
  },  
  "Credentials": {  
    "SecretAccessKey": "9drTJvcXLB89EXAMPLEELB8923FB892xMFI",  
    "SessionToken": "AQoXdzELDDY/////////  
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/  
qwjzP2iEXAMPLEEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg  
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtY1FVgAUj  
+7Indz3LU0aTWk1WKIjHmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM  
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B  
IcrxSpnWEXAMPLEXSDFTAQAM6D19zR0tXoybnlrZIwML1Mi1Kcgo50ytwU=",  
    "Expiration": "2016-03-15T00:05:07Z",  
    "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"  
  }  
}
```

コマンドの出力には、AWS に対する認証に使用できるアクセスキー、シークレットキー、およびセッショントークンが含まれています。

AWS CLI を使用する場合は、ロールに関連付けられた名前付きプロファイルを設定できます。プロファイルを使用すると、AWS CLI は `assign-role` を呼び出し、ユーザーのために認証情報を管理します。詳細については、「AWS CLI ユーザーガイド」の「[AWS CLI で IAM ロールを使用する](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[AssumeRole](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
```

```
* ]
* }
*
* For more information, see "Editing the Trust Relationship for an Existing
* Role" in the AWS Directory Service guide.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to
                assume (for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
                (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
            .region(region)
            .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }
}
```



```
public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();

        // Display the time when the temp creds expire.
        Instant exTime = myCreds.expiration();
        String tokenInfo = myCreds.sessionToken();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(exTime);
        System.out.println("The token " + tokenInfo + " expires on " +
exTime);

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AssumeRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

クライアントを作成します。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

IAM ロールを割り当てます。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  }
}
```

```
    } catch (err) {  
      console.error(err);  
    }  
  };  
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[AssumeRole](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js  
const AWS = require("aws-sdk");  
// Set the region  
AWS.config.update({ region: "REGION" });  
  
var roleToAssume = {  
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",  
  RoleSessionName: "session1",  
  DurationSeconds: 900,  
};  
var roleCreds;  
  
// Create the STS service object  
var sts = new AWS.STS({ apiVersion: "2011-06-15" });  
  
//Assume Role  
sts.assumeRole(roleToAssume, function (err, data) {  
  if (err) console.log(err, err.stack);  
  else {  
    roleCreds = {  
      accessKeyId: data.Credentials.AccessKeyId,  
      secretAccessKey: data.Credentials.SecretAccessKey,  
      sessionToken: data.Credentials.SessionToken,  
    };  
    stsGetCallerIdentity(roleCreds);  
  }  
}
```

```
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[AssumeRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 一時的な認証情報一式 (アクセスキー、シークレットキー、セッショントークン) を返します。この認証情報は、リクエストしたユーザーが通常はアクセスできない AWS リソースに 1 時間アクセスするために使用できます。返される認証情報には、引き受けているロールのアクセスポリシーと提供されたポリシーで許可されている権限があります (提供されたポリシーを使用して、引き受けているロールのアクセスポリシーで定義されている権限を超える権限を付与することはできません)。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-Policy "...JSON policy..." -DurationInSeconds 3600
```

例 2: 引き受けているロールのアクセスポリシーで定義されているのと同じ権限を持つ、1 時間有効の一時的な認証情報を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600
```

例 3: コマンドレットの実行に使用されるユーザー認証情報に関連付けられている MFA からシリアル番号と生成されたトークンを提供する一時的な認証情報一式を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

例 4: 顧客アカウントで定義されているロールを引き受けた一時的な認証情報一式を返します。第三者が引き受けることができるロールごとに、顧客アカウントは、ロールを引き受けるたび、`-ExternalId` パラメータで渡す必要がある識別子を使用してロールを作成する必要があります。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600 -ExternalId "ABC123"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AssumeRole](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

MFA トークンを必要とする IAM ロールを想定し、一時的な認証情報を使用してアカウントの Amazon S3 バケットを一覧表示します。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.
```

```
:param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                        grants access to list the other account's buckets.
:param session_name: The name of the STS session.
:param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                        device, this is an ARN.
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the
role.
"""
response = sts_client.assume_role(
    RoleArn=assume_role_arn,
    RoleSessionName=session_name,
    SerialNumber=mfa_serial_number,
    TokenCode=mfa_totp,
)
temp_credentials = response["Credentials"]
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AssumeRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
#           are used.
# @param sts_client [Aws::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end
```



- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AssumeRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;

    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :          {}",
e.user_id().unwrap_or_default());
            println!("Account:          {}",
e.account().unwrap_or_default());
            println!("Arn      :          {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[AssumeRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
    async throws -> STSClientTypes.Credentials {
    let input = AssumeRoleInput(
        roleArn: role.arn,
        roleSessionName: sessionName
    )
    do {
        let output = try await stsClient.assumeRole(input: input)

        guard let credentials = output.credentials else {
            throw ServiceHandlerError.authError
        }

        return credentials
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AssumeRole](#)」を参照してください。

## 使用していない AWS 認証情報の検索


AWS アカウント のセキュリティを高めるには、不要な IAM ユーザー認証 (パスワードおよびアクセスキー) を削除します。たとえば、ユーザーが組織を離れる際、または今後 AWS へのアクセスが不要な場合には、使用されていた認証情報を検索し、それらが無効になっていることを確認する必要があります。必要のなくなった認証情報は削除することが理想的です。それらは必要に応じて、後日いつでも再作成できます。少なくとも、パスワードを変更するか、アクセスキーを無効にして、以前のユーザーがアクセスできないようにする必要があります。

もちろん、使用していないの定義は異なる可能性があります。通常は特定の期間内に使用されなかった認証情報を指します。

## 使用していないパスワードの検索

AWS Management Console を使用して、ユーザーのパスワード使用状況情報を表示できます。多数のユーザーがいる場合は、コンソールを使用して、ユーザーごとのコンソールパスワードの最終使用日時に関する情報を含む認証情報レポートをダウンロードできます。また、AWS CLI または IAM API から情報にアクセスすることも可能です。

未使用のパスワードを検索するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 必要に応じて、[Console last sign-in (コンソールへの前回サインイン)] 列を USERS テーブルに追加します。
  - a. 右端のテーブルの上で、設定アイコン (  ) を選択します。
  - b. [表示する列の選択] で、[コンソールの最終サインイン] を選択します。
  - c. [確認] を選択して、ユーザーのリストに戻ります。
4. [Console last sign-in] (コンソールへの最終サインイン) 列に、ユーザーがコンソールを通じて AWS に最後にサインインした日付が表示されます。この情報を使用して、特定の期間以上サイ

サインしていないユーザーとパスワードを検索できます。この列には、一度もサインしていないユーザーとパスワードに、[なし]と表示されます。[なし]は、パスワードが設定されていないユーザーを表します。最近使用されていないパスワードは削除の対象となります。

### Important

サービス上の問題により、前回使用したパスワードに関するデータには、2018年5月3日22時50分(太平洋夏時間)から2018年5月23日14時08分(太平洋夏時間)までのパスワードの使用は含まれません。これは、IAM コンソールに表示される[前回サインイン](#)した日付および[IAM 認証情報レポート](#)でパスワードを前回使用した日付として[GetUser API オペレーション](#)から返される日付に影響を与えます。該当する期間中にユーザーがサインインした場合、パスワードを前回使用した日付として返される日付は、2018年5月3日より前にユーザーが最後にサインインした日付になります。2018年5月23日14時08分(太平洋夏時間)より後にサインインしたユーザーの場合、パスワードを前回使用した日付として返される日付は正確です。

前回使用したパスワードに関する情報を使用して未使用の認証情報を特定して削除する(例: 過去90日間にAWSにサインインしなかったユーザーを削除する)場合は、2018年5月23日より後の日付を含めるように評価ウィンドウを調整してください。または、ユーザーがアクセスキーを使用してAWSにプログラムでアクセスする場合は、前回使用したアクセスキーの情報を参照できます。これはすべての日付について正確であるためです。

コンソールで認証情報レポートをダウンロードして、使用されていないパスワードを検索するには(コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[認証情報レポート] を選択します。
3. [レポートをダウンロード] を選択して、`status_reports_<date>T<time>.csv` という名前のカンマ区切り値 (CSV) のファイルをダウンロードします。5 番目の列には、日付または以下のいずれか 1 つを含む `password_last_used` 列が表示されます。
  - 該当なし – 割り当てられているパスワードが 1 つもないユーザー。
  - [no\_information (情報なし)] – IAM パスワードの有効期間の追跡を開始した 2014 年 10 月 20 日以降にパスワードを使用していないユーザー。

## 未使用のパスワードを見つけるには (AWS CLI)

未使用のパスワードを見つけるには、次のコマンドを実行します。

- [aws iam list-users](#) は、ユーザーのリストを返します。各ユーザーには、PasswordLastUsed 値が設定されています。値がない場合は、ユーザーがパスワードを持っていないか、IAM がパスワードの有効期限の追跡を開始した 2014 年 10 月 20 日以降にパスワードが使用されていないことを示します。

## 未使用のパスワードを見つけるには (AWS API)

未使用のパスワードを見つけるには、次のオペレーションを呼び出します。

- [ListUsers](#) は、ユーザーのコレクションを返します。各ユーザーには、<PasswordLastUsed> 値が設定されています。値がない場合は、ユーザーがパスワードを持っていないか、IAM がパスワードの古さの追跡を開始した 2014 年 10 月 20 日以降にパスワードが使用されていないことを示します。


認証情報レポートをダウンロードするためのコマンドについては、「[認証情報レポートの取得 \(AWS CLI\)](#)」を参照してください。

## 使用していないアクセスキーの検索

AWS Management Console を使用して、ユーザーのアクセスキーの使用状況に関する情報を表示できます。多数のユーザーがいる場合は、コンソールを使用して認証情報レポートをダウンロードし、ユーザーごとのアクセスキーの最終使用日時を検索できます。また、AWS CLI または IAM API から情報にアクセスすることも可能です。

使用していないアクセスキーを検索するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. 必要に応じて、[Access key last used (アクセスキー前回使用)] 列をユーザーテーブルに追加します。

- a. 右端のテーブルの上で、設定アイコン  
(  )  
を選択します。
  - b. [表示する列の選択] で、[最後に使用したアクセスキー] を選択します。
  - c. [確認] を選択して、ユーザーのリストに戻ります。
4. [Access key last used (アクセスキー前回使用)] 列には、ユーザーがプログラムで最後に AWS にアクセスしてから経過した日数が表示されます。この情報を使用して、指定の期間以上使用されていないアクセスキーを持つユーザーを検索できます。この列のアクセスキーのないユーザーに、[-] と表示されます。最近使用されていないアクセスキーは削除の対象となります。

認証情報レポートをダウンロードして、使用していないアクセスキーを検索するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[認証情報レポート] を選択します。
3. [レポートをダウンロード] を選択して、`status_reports_<date>T<time>.csv` という名前のカンマ区切り値 (CSV) のファイルをダウンロードします。列 11 ~ 13 にはアクセスキー 1 を最後に使用した日付、アクセスキーのリージョン、およびサービス情報が含まれます。列 16 ~ 18 にはアクセスキー 2 の同様の情報が含まれます。ユーザーがアクセスキーを持っていないか、IAM がアクセスキーの有効期限の追跡を開始した 2015 年 4 月 22 日以降にアクセスキーが使用されていない場合、値 [該当なし] が表示されます。

未使用のアクセスキーを見つけるには (AWS CLI)

以下のコマンドを使用して、未使用のアクセスキーを見つけることができます。

- [aws iam list-access-keys](#) は AccessKeyID を含む、ユーザーのアクセスキーに関する情報を返します。
- [aws iam get-access-key-last-used](#) はアクセスキー ID を使用して、LastUsedDate、アクセスキーを最後に使用した Region、最後に要求されたサービスの ServiceName を含む出力を返します。LastUsedDate が見つからない場合、IAM がアクセスキーの有効期限の追跡を開始した 2015 年 4 月 22 日以降にアクセスキーは使用されていません。

未使用のアクセスキーを見つけるには (AWS API)

未使用のアクセスキーを見つけるには、以下のオペレーションを呼び出します。

- [ListAccessKeys](#) は指定したユーザーに関連付けられたアクセスキーの AccessKeyID 値のリストを返します。
- [GetAccessKeyLastUsed](#) はアクセスキー ID を使用して、値のコレクションを返します。これには、LastUsedDate、アクセスキーを最後に使用した Region、最後に要求されたサービスの ServiceName が含まれています。値が見つからない場合、ユーザーがアクセスキーを持っていないか、IAM がアクセスキーの古さの追跡を開始した 2015 年 4 月 22 日以降にアクセスキーが使用されていないことを示します。

認証情報レポートをダウンロードするためのコマンドについては、「[認証情報レポートの取得 \(AWS CLI\)](#)」を参照してください。

## AWS アカウント の認証情報レポートの取得

アカウントのすべてのユーザーと、ユーザーの各種認証情報 (パスワード、アクセスキー、MFA デバイスなど) のステータスが示された認証情報レポートを生成し、ダウンロードできます。認証情報レポートは、AWS Management Console、[AWS SDK](#)、[コマンドラインツール](#)、または IAM API から取得できます。

認証情報レポートを使用して、監査やコンプライアンスの作業を支援することができます。このレポートを使用して、パスワードやアクセスキーの更新など、認証情報ライフサイクルの要件の効果を監査できます。外部の監査人にこのレポートを提供することも、監査人が直接このレポートをダウンロードできるようにアクセス権限を付与することもできます。

認証情報レポートは、4 時間ごとに 1 回生成できます。レポートをリクエストすると、IAM はまず AWS アカウント について過去 4 時間以内にレポートが生成されたかどうかを確認します。レポートが生成されている場合は、最新のレポートがダウンロードされます。アカウントの最新のレポートが 4 時間以上前のものであるか、アカウントに以前のレポートがない場合、IAM は新しいレポートを生成してダウンロードします。

### トピック

- [必要なアクセス許可](#)
- [レポートフォーマットの理解](#)
- [認証情報レポートの取得 \(コンソール\)](#)
- [認証情報レポートの取得 \(AWS CLI\)](#)
- [認証情報レポートの取得 \(AWS API\)](#)



## 必要なアクセス許可

レポートを作成してダウンロードするには、以下のアクセス許可が必要です。

- 認証情報レポートを作成生成するには: `iam:GenerateCredentialReport`
- レポートをダウンロードするには: `iam:GetCredentialReport`

## レポートフォーマットの理解

認証情報レポートは、カンマ区切り値 ( CSV ) ファイルとしてフォーマットされます。CSV ファイルを一般的なスプレッドシートソフトウェアで開いて分析を実行できます。また、CSV ファイルをプログラマ的に利用するアプリケーションを作成して、カスタム分析を実行することもできます。

CSV ファイルには、次の列が含まれます。

`user`

ユーザーのわかりやすい名前。

`arn`

ユーザーの Amazon リソースネーム ( ARN )。ARN の詳細については、[IAM ARN](#)を参照してください。

`user_creation_time`

ユーザーが作成された日時 ([ISO 8601 日付/時刻形式](#))。

`password_enabled`

ユーザーがパスワードを持っている場合、この値は TRUE です。それ以外の場合は FALSE です。AWS アカウントのルートユーザー の値は常に `not_supported` です。

`password_last_used`

AWS アカウントのルートユーザー またはユーザーのパスワードを使用して最後に AWS ウェブサイトにサインインした日時「([ISO 8601 日付/時刻形式](#))」。AWS ユーザーの最終サインイン時刻をキャプチャするウェブサイトには、AWS Management Console、AWS ディスカッションフォーラム、および AWS Marketplace があります。5 分以内にパスワードが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。

- 次のような場合、このフィールドの値は `no_information` になります。
  - ユーザーのパスワードが使用されたことがない場合。

- IAM が 2014 年 10 月 20 日にこの情報の追跡を開始してから、ユーザーのパスワードが使用されていないなど、パスワードに関連付けられたサインインデータがない場合。
- ユーザーがパスワードを所有していない場合、このフィールドの値は N/A (該当なし) です。

### ⚠ Important

サービス上の問題により、前回使用したパスワードに関するデータには、2018 年 5 月 3 日 22 時 50 分 (太平洋夏時間) から 2018 年 5 月 23 日 14 時 08 分 (太平洋夏時間) までのパスワードの使用は含まれません。これは、IAM コンソールに表示される[前回サインイン](#)した日付および [IAM 認証情報レポート](#)でパスワードを前回使用した日付として [GetUser API オペレーション](#)から返される日付に影響を与えます。該当する期間中にユーザーがサインインした場合、パスワードを前回使用した日付として返される日付は、2018 年 5 月 3 日より前にユーザーが最後にサインインした日付になります。2018 年 5 月 23 日 14 時 08 分 (太平洋夏時間) より後にサインインしたユーザーの場合、パスワードを前回使用した日付として返される日付は正確です。

前回使用したパスワードに関する情報を使用して未使用の認証情報を特定して削除する (例: 過去 90 日間に AWS にサインインしなかったユーザーを削除する) 場合は、2018 年 5 月 23 日より後の日付を含めるように評価ウィンドウを調整してください。または、ユーザーがアクセスキーを使用して AWS にプログラムでアクセスする場合は、前回使用したアクセスキーの情報を参照できます。これはすべての日付について正確であるためです。

### password\_last\_changed

ユーザーのパスワードが最後に設定された日時 ([ISO 8601 日付/時刻形式](#))。ユーザーがパスワードを所有していない場合、このフィールドの値は N/A (該当なし) です。AWS アカウント (ルート) の値は常に not\_supported です。

### password\_next\_rotation

アカウントにパスワードの更新を必要とする[パスワードポリシー](#)がある場合、このフィールドには、新しいパスワードを設定するようユーザーに求める日時 ([ISO 8601 日付/時刻形式](#)) が保存されます。AWS アカウント (ルート) の値は常に not\_supported です。

### mfa\_active

ユーザーに対して[多要素認証](#) (MFA) デバイスが有効になっていた場合、この値は TRUE です。それ以外の場合は、FALSE です。

### access\_key\_1\_active

ユーザーがアクセスキーを所有し、そのアクセスキーのステータスが `Active` である場合、この値は `TRUE` です。それ以外の場合は、`FALSE` です。

### access\_key\_1\_last\_rotated

ユーザーのアクセスキーが作成または最後に変更された日時 ([ISO 8601 日付/時刻形式](#))。ユーザーがアクティブなアクセスキーを所有していない場合、このフィールドの値は `N/A` (該当なし) です。

### access\_key\_1\_last\_used\_date

AWS API リクエストの署名にユーザーのアクセスキーが直近に使用されたときの [ISO 8601 日付/時刻形式](#)の日付と時刻。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。

次のような場合、このフィールドの値は `N/A` (該当なし) になります。

- ユーザーにアクセスキーがない場合。
- アクセスキーが一度も使用されていない場合。
- IAM が 2015 年 4 月 22 日にこの情報の追跡を開始してから、アクセスキーが使用されていない場合。

### access\_key\_1\_last\_used\_region

アクセスキーが直近に使用された [AWS リージョン](#)。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。

次のような場合、このフィールドの値は `N/A` (該当なし) になります。

- ユーザーにアクセスキーがない場合。
- アクセスキーが一度も使用されていない場合。
- アクセスキーが最後に使用されたのが、IAM が 2015 年 4 月 22 日にこの情報の追跡を開始するより前の場合。
- 最後に使用したサービスは、Amazon S3 など、リージョン固有ではありません。

### access\_key\_1\_last\_used\_service

アクセスキーを使用して最近アクセスされた AWS サービス。このフィールドの値として、サービスの s3 名前空間 (Amazon S3 の、Amazon EC2 の `ec2` など) が使用されます。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。

次のような場合、このフィールドの値は `N/A` (該当なし) になります。

- ユーザーにアクセスキーがない場合。
- アクセスキーが一度も使用されていない場合。
- アクセスキーが最後に使用されたのが、IAM が 2015 年 4 月 22 日にこの情報の追跡を開始するより前の場合。

#### access\_key\_2\_active

ユーザーが 2 つ目のアクセスキーを所有し、その 2 つ目のキーのステータスが Active である場合、この値は TRUE です。それ以外の場合は、FALSE です。

#### Note

ユーザーはアクセスキーを 2 つまで持つことができ、最初にキーを更新してから前のキーを削除すると、ローテーションが簡単になります。アクセスキーの更新の詳細については、「[アクセスキーの更新](#)」を参照してください。

#### access\_key\_2\_last\_rotated

ユーザーの 2 つ目のアクセスキーが作成された、または最後に更新された日時 ([ISO 8601 日付/時刻形式](#))。ユーザーが 2 つ目のアクティブなアクセスキーを所有していない場合、このフィールドの値は N/A (該当なし) です。

#### access\_key\_2\_last\_used\_date

AWS API リクエストの署名にユーザーの 2 つ目のアクセスキーが直近に使用されたときの [ISO 8601 日付/時刻形式](#) の日付と時刻。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。

次のような場合、このフィールドの値は N/A (該当なし) になります。

- ユーザーに 2 つ目のアクセスキーがない場合。
- ユーザーの 2 つ目のアクセスキーが一度も使用されていない場合。
- ユーザーの 2 つ目のアクセスキーが最後に使用されたのが、IAM が 2015 年 4 月 22 日にこの情報の追跡を開始するより前の場合。

#### access\_key\_2\_last\_used\_region

ユーザーの 2 つ目のアクセスキーが直近に使用された [AWS リージョン](#)。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。次のような場合、このフィールドの値は N/A (該当なし) になります。

- ユーザーに 2 つ目のアクセスキーがない場合。

- ユーザーの 2 つ目のアクセスキーが一度も使用されていない場合。
- ユーザーの 2 つ目のアクセスキーが最後に使用されたのが、IAM が 2015 年 4 月 22 日にこの情報の追跡を開始するより前の場合。
- 最後に使用したサービスは、Amazon S3 など、リージョン固有ではありません。

#### access\_key\_2\_last\_used\_service

ユーザーの 2 つ目のアクセスキーを使用して最近アクセスされた AWS サービス。このフィールドの値として、サービスの s3 名前空間 (Amazon S3 の、Amazon EC2 の ec2 など) が使用されます。15 分以内にアクセスキーが複数回使用された場合、最初の使用のみがこのフィールドに記録されます。次のような場合、このフィールドの値は N/A (該当なし) になります。

- ユーザーに 2 つ目のアクセスキーがない場合。
- ユーザーの 2 つ目のアクセスキーが一度も使用されていない場合。
- ユーザーの 2 つ目のアクセスキーが最後に使用されたのが、IAM が 2015 年 4 月 22 日にこの情報の追跡を開始するより前の場合。

#### cert\_1\_active

ユーザーが X.509 署名証明書を所有し、その証明書のステータスが Active である場合、この値は TRUE です。それ以外の場合は、FALSE です。

#### cert\_1\_last\_rotated

ユーザーの署名証明書が作成または最後に変更された日時 ([ISO 8601 日付/時刻形式](#))。ユーザーがアクティブな署名証明書を所有していない場合、このフィールドの値は N/A (該当なし) です。

#### cert\_2\_active

ユーザーが 2 つ目の X.509 署名証明書を所有し、その証明書のステータスが Active である場合、この値は TRUE です。それ以外の場合は、FALSE です。

#### Note

証明書のローテーションを容易にするために、ユーザーは最大で 2 個の X.509 署名証明書を持つことができます。

#### cert\_2\_last\_rotated

ユーザーの 2 つ目の署名証明書が作成または最後に変更された日時 ([ISO 8601 日付/時刻形式](#))。ユーザーが 2 つ目のアクティブな署名証明書を所有していない場合、このフィールドの値は N/A (該当なし) です。

## 認証情報レポートの取得 (コンソール)

AWS Management Consoleを使用して、認証情報レポートをカンマ区切り値 (CSV) ファイルとしてダウンロードできます。

認証情報レポートをダウンロードするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[認証情報レポート] を選択します。
3. [レポートをダウンロード] を選択します。

## 認証情報レポートの取得 (AWS CLI)

認証情報レポートをダウンロードするには (AWS CLI)

1. 認証情報レポートを生成します。AWS には、単一のレポートが格納されます。レポートが存在する場合、認証情報レポートを生成すると、以前のレポートが上書きされます。 [aws iam generate-credential-report](#)
2. 最後に生成されたレポートを表示します: [aws iam get-credential-report](#)

## 認証情報レポートの取得 (AWS API)

認証情報レポートをダウンロードするには (AWS API)

1. 認証情報レポートを生成します。AWS には、単一のレポートが格納されます。レポートが存在する場合、認証情報レポートを生成すると、以前のレポートが上書きされません。 [GenerateCredentialReport](#)
2. 最後に生成されたレポートを表示します: [GetCredentialReport](#)

## CodeCommit での IAM の使用: Git 認証情報、SSH キー、および AWS アクセスキー

CodeCommit はマネージド型バージョン管理サービスであり、AWS クラウド内のプライベート Git リポジトリをホストします。CodeCommit を使用するには、Git クライアントを CodeCommit リポジトリと通信できるように設定します。この設定の一環として、CodeCommit がユーザーの認証に

使用できる IAM 認証情報を指定します。IAM は、次の 3 種類の認証情報で CodeCommit をサポートしています。

- Git 認証情報。HTTPS 経由の CodeCommit リポジトリとの通信に使用できる、IAM によって生成されたユーザー名とパスワードのペアです。
- SSH キー。IAM ユーザーに関連付けて SSH 経由で CodeCommit リポジトリと通信できる、ローカルで生成されたパブリックキーとプライベートキーのペアです。
- [AWS アクセスキー](#)。AWS CLI に含まれる認証情報ヘルパーを使用して、HTTPS 経由で CodeCommit リポジトリと通信できます。

#### Note

別の AWS アカウントのリポジトリにアクセスするには、SSH キーまたは Git 認証情報を使用できません。別の IAM ユーザーおよびグループに対して、CodeCommit リポジトリへのアクセスを設定する方法を学ぶには AWS アカウントの詳細については、「[AWS CodeCommit ユーザーガイド](#)」の「[ロールを使用した AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する](#)」を参照してください。

以下のセクションに各オプションの詳細を示します。

## CodeCommit で Git 認証情報および HTTPS を使用する (推奨)

Git 認証情報では、IAM ユーザーの静的ユーザー名とパスワードのペアを生成し、その認証情報を HTTPS 接続に使用します。また、静的 Git 認証情報をサポートするサードパーティー製ツールや統合開発環境 (IDE) でもこの認証情報を使用できます。

これらの認証情報はサポートされているすべてのオペレーティングシステムで共通であり、ほとんどの認証情報管理システム、開発環境、その他のソフトウェア開発ツールと互換性があるため、推奨される方法です。Git 認証情報のパスワードはいつでもリセットできます。また、認証情報が不要になった場合は、非アクティブ化または削除できます。

#### Note

Git 認証情報用のユーザー名やパスワードは選択できません。IAM は、これらのクレデンシャルを生成して、AWS のセキュリティ標準と CodeCommit の安全なリポジトリを確実に満たすようにします。認証情報は、生成時に 1 回のみダウンロードできます。必ず、認証情報



を安全な場所に保存してください。必要に応じて、パスワードをいつでもリセットできますが、そうすると古いパスワードを使用した接続が無効になります。接続するには、新しいパスワードを使用するように接続を再構成する必要があります。

詳細については、以下のトピックを参照してください。

- IAM ユーザーを作成するには、「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。
- CodeCommit で Git 認証情報を生成して使用するには、[AWS CodeCommit ユーザーガイド](#)の「Git 認証情報を使用する HTTPS ユーザーの場合」を参照してください。

#### Note

Git 認証情報を生成した後で IAM ユーザーの名前を変更しても、Git 認証情報のユーザー名は変更されません。ユーザー名とパスワードは変わらず、有効のままです。

サービス固有の認証情報を更新するには

1. 現在使用されているセットに加えて、2 つ目のサービス固有の認証情報セットを作成します。
2. 新しい認証情報のセットを使用するようにすべてのアプリケーションを更新して、アプリケーションが動作することを確認します。
3. 元の認証情報の状態を「非アクティブ」に変更します。
4. すべてのアプリケーションが動作していることを確認します。
5. 非アクティブ化したサービス固有の認証情報を削除します。

## CodeCommit で SSH キーと SSH を使用する

SSH 接続では、SSH 認証用に Git および CodeCommit が使用するパブリックキーとプライベートキーのファイルをローカルマシンで作成します。パブリックキーは IAM ユーザーに関連付け、プライベートキーはローカルマシンに保存します。詳細については、以下のトピックを参照してください。

- IAM ユーザーを作成するには、「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。

- SSH 公開キーを作成して IAM ユーザーに関連付けるには、[AWS CodeCommit ユーザーガイドの「Linux、macOS、または Unix での SSH 接続について」](#) または 「Windows での SSH 接続について」を参照してください。

#### Note

パブリックキーは ssh-rsa 形式または PEM 形式でエンコードされます。パブリックキーの最小のビットの長さは 2048 ビットですが、パブリックキーの最大長は 16384 ビットです。これはアップロードするファイルのサイズとは異なります。たとえば 2048 ビットキーを生成した場合、作成された PEM ファイルの長さは 1679 バイトになります。パブリックキーを別の形式またはサイズで提供すると、キーの形式が無効であることを知らせるエラーメッセージが表示されます。

## AWS CLI 認証情報ヘルパーおよび CodeCommit で HTTPS を使用する

Git が CodeCommit リポジトリとの通信で AWS に対する認証を必要とする場合は、Git 認証情報を使用した HTTPS 接続の代わりに、暗号化された署名済みバージョンの IAM ユーザー認証情報または Amazon EC2 インスタンスロールを Git で使用できます。これは、IAM ユーザーを必要としない CodeCommit リポジトリと接続する唯一の方法です。また、フェデレーションアクセスおよび一時的な認証情報を使用できる唯一の方法でもあります。詳細については、以下のトピックを参照してください。

- フェデレーションアクセスの詳細については、[ID プロバイダーとフェデレーション](#) および [外部で認証されたユーザー \(ID フェデレーション\) へのアクセスの許可](#) を参照してください。
- 一時的な認証情報の詳細については、「[IAM の一時的な認証情報](#)」および「[CodeCommit リポジトリへの一時アクセス](#)」を参照してください。

AWS CLI 認証情報ヘルパーには、Keychain Access や Windows Credential Management などの他の認証情報ヘルパーシステムとの互換性はありません。認証情報ヘルパーを使用して HTTPS 接続を設定する際は、追加の設定考慮事項があります。詳細については、[AWS CLI ユーザーガイドの「AWS CLI 資格情報ヘルパーを使用した Linux、macOS、または Unix での HTTPS 接続」](#) または 「AWS CodeCommit 資格情報ヘルパーを使用した Windows での HTTPS 接続について」を参照してください。

## Amazon Keyspaces での IAM の使用 ( Apache Cassandra 用 )

Amazon Keyspaces (Apache Cassandra 用) は、スケーラブルで可用性の高い、Apache Cassandra 互換のマネージドデータベースサービスです。AWS Management Console を通じてかプログラムにより、Amazon Keyspaces にアクセスできます。サービス固有の認証情報を使用して Amazon Keyspaces にプログラムでアクセスするには、cqlsh またはオープンソースの Cassandra ドライバーを使用できます。サービス固有の認証情報は、Cassandra が認証とアクセス管理に使用するようなユーザー名とパスワードを含みます。サポート対象の各サービスでは、ユーザーごとにサービス固有の認証情報を最大 2 セット設定できます。

AWS アクセスキーを使用して Amazon Keyspaces にプログラムでアクセスするには、AWS SDK や AWS Command Line Interface (AWS CLI) またはオープンソースの Cassandra ドライバーと SigV4 プラグインを使用できます。詳細については、「Amazon Keyspaces (Apache Cassandra 向け) 開発者ガイド」の「[Amazon Keyspaces へのプログラムによる接続](#)」を参照してください。

### Note

コンソール経由で Amazon Keyspaces とやり取りする予定がある場合、サービス固有の認証情報を生成する必要はありません。詳細については、「Amazon Keyspaces (Apache Cassandra 向け) 開発者ガイド」の「[コンソールを使用した Amazon Keyspaces へのアクセス](#)」を参照してください。

Amazon Keyspaces にアクセスするために必要な権限の詳細については、「[Amazon Keyspaces \(Apache Cassandra 用\) 開発者ガイド](#)」の「Amazon キースペース (Apache Cassandra の場合) ID ベースのポリシーの例」を参照してください。

### Amazon Keyspaces 認証情報の生成 (コンソール)

AWS Management Console を使用して、IAM ユーザー用の Amazon Keyspaces (Apache Cassandra 用) 認証情報を生成できます。

Amazon Keyspaces サービス固有の認証情報を生成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザー] を選択し、認証情報を必要とするユーザーの名前を選択します。

3. [セキュリティ認証情報] タブにある [Amazon Keyspaces (Apache Cassandra 用) の認証情報] で、[認証情報を生成する] を選択します。
4. これで、サービス固有の認証情報が利用可能になりました。パスワードを表示またはダウンロードできるのはこのときだけです。後で回復することはできません。ただし、パスワードはいつでもリセットできます。ユーザーおよびパスワードは後で必要になるため、安全な場所に保存します。

## Amazon Keyspaces 認証情報の生成 ( AWS CLI )

AWS CLI を使用して、IAM ユーザー用の Amazon Keyspaces (Apache Cassandra 用) 認証情報を生成できます。

Amazon Keyspaces サービス固有の認証情報を生成するには (AWS CLI)

- 以下のコマンドを使用します。
  - [aws iam create-service-specific-credential](#)

## Amazon Keyspaces 認証情報の生成 ( AWS API )

AWS API を使用して、IAM ユーザー用の Amazon Keyspaces (Apache Cassandra 用) 認証情報を生成できます。

Amazon Keyspaces サービス固有の認証情報を生成するには (AWS API)

- 以下のオペレーションを実行します。
  - [CreateServiceSpecificCredential](#)

## IAM でのサーバー証明書の管理

ウェブサイトまたは AWS のアプリケーションへの HTTPS 接続を有効にするには、SSL/TLS サーバー証明書が必要です。AWS Certificate Manager (IAM) によってサポートされているリージョンの証明書では、ACM を使用して、サーバー証明書をプロビジョン、管理、およびデプロイすることをお勧めします。サポートされていないリージョンでは、IAM を Certificate Manager として使用する必要があります。ACM がサポートするリージョンについては、「AWS 全般のリファレンス」の「[AWS Certificate Manager エンドポイントとクォータ](#)」を参照してください。

ACM は、サーバー証明書をプロビジョン、管理、デプロイするための推奨ツールです。ACM を使用すると、証明書をリクエストしたり、既存の ACM または外部証明書を AWS リソースにデプロイしたりできます。ACM で提供された証明書は無料で自動的に更新されます。[サポートされているリージョン](#)では、ACM を使用して、コンソールまたはプログラムでサーバー証明書を管理できます。ACM の使用の詳細については、[AWS Certificate Manager ユーザーガイド](#)を参照してください。ACM 証明書のリクエストの詳細については、AWS Certificate Manager ユーザーガイドの「[パブリック証明書のリクエスト](#)」または「[プライベート証明書のリクエスト](#)」を参照してください。ACM へのサードパーティー証明書のインポートの詳細については、AWS Certificate Manager ユーザーガイドの「[証明書のインポート](#)」を参照してください。

[ACM でサポート](#)されていないリージョンで HTTPS 接続をサポートする必要があるときのみ、Certificate Manager として IAM を使用してください。IAM はプライベートキーを安全に暗号化し、暗号化されたバージョンを IAM SSL 証明書ストレージに保存します。IAM はすべてのリージョンでのサーバー証明書のデプロイをサポートしますが、AWS で使用するには、外部プロバイダーから証明書を取得する必要があります。ACM 証明書を IAM にアップロードすることはできません。また、IAM コンソールから証明書を管理することはできません。

IAM へのサードパーティー証明書のアップロードの詳細については、以下のトピックを参照してください。

## 内容

- [サーバー証明書のアップロード \(AWS API\)](#)
- [サーバー証明書の取得 \(AWS API\)](#)
- [サーバー証明書の一覧表示 \(AWS API\)](#)
- [サーバー証明書のタグ付けとタグの解除 \(AWS API\)](#)
- [サーバー証明書の名前変更またはパスの更新 \(AWS API\)](#)
- [サーバー証明書の削除 \(AWS API\)](#)
- [トラブルシューティング](#)

## サーバー証明書のアップロード (AWS API)

IAM にサーバー証明書をアップロードするには、証明書と対応するプライベートキーを提供する必要があります。証明書が自己署名されていない場合、証明書チェーンも提供する必要があります (自己署名証明書をアップロードするときに証明書チェーンは必要ありません)。証明書をアップロードする前に、これらの項目がすべてあり、以下の条件を満たしていることを確認します。

- 証明書はアップロード時に有効である必要があります。有効期間の開始 (証明書の NotBefore 日付) 前、または有効期間の終了 (証明書の NotAfter 日) 後に証明書をアップロードすることはできません。
- プライベートキーは非暗号化される必要があります。パスワードやパスフレーズで保護されたプライベートキーをアップロードすることはできません。暗号化されたプライベートキーの復号のヘルプについては、「[トラブルシューティング](#)」を参照してください。
- 証明書、プライベートキー、および証明書チェーンはすべて PEM エンコードされる必要があります。これらの項目の PEM 形式への変換のヘルプについては、「[トラブルシューティング](#)」を参照してください。

[IAM API](#) を使用して証明書をアップロードするには、[UploadServerCertificate](#) リクエストを送信します。次の例では、[AWS Command Line Interface \(AWS CLI\)](#) を使用してこのオペレーションを行う方法を示します。例では、次のように想定しています。

- PEM エンコードされた証明書は、Certificate.pem というファイルに保存されます。
- PEM エンコードされた証明書チェーンは、CertificateChain.pem というファイルに保存されます。
- PEM エンコードされ、非暗号化されたプライベートキーは、PrivateKey.pem というファイルに保存されます。
- (オプション) キーバリューペアを使ってサーバー証明書にタグを付けるとします。例えば、証明書の特定と整理を行うために、タグキー Department と タグ値 Engineering を追加することが可能です。

次のコマンドの例を使用するには、これらのファイル名を独自のファイル名に置き換えます。*ExampleCertificate* をアップロードした証明書の名前に置き換えます。証明書にタグを付ける場合は、*ExampleKey* と *ExampleValue* タグのキーバリューのペアを独自の値に置き換えます。1つの連続した行にコマンドを入力します。次の例では、読みやすくするために改行とスペースを追加しています。

```
aws iam upload-server-certificate --server-certificate-name ExampleCertificate
                                   --certificate-body file://Certificate.pem
                                   --certificate-chain file://CertificateChain.pem
                                   --private-key file://PrivateKey.pem
                                   --tags '{"Key": "ExampleKey", "Value":
"ExampleValue"}'
```



前のコマンドが成功すると、[Amazon リソースネーム \(ARN\)](#)、わかりやすい名前、識別子 (ID)、有効期限日、タグなど、アップロードされた証明書に関するメタデータを返します。

#### Note

Amazon CloudFront で使用することを目的としてサーバー証明書をアップロードする場合、`--path` オプションを使用してパスを指定する必要があります。パスの先頭に `/cloudfront` を含め、末尾にスラッシュを含める必要があります (例: `/cloudfront/test/`)。

AWS Tools for Windows PowerShell を使用して証明書をアップロードするには、[Publish-IAMServerCertificate](#) を使用します。

### サーバー証明書の取得 (AWS API)

IAM API を使用して証明書を取得するには、[GetServerCertificate](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。*ExampleCertificate* を、取得する証明書の名前と置き換えます。

```
aws iam get-server-certificate --server-certificate-name ExampleCertificate
```

前のコマンドが成功した場合、証明書、証明書チェーン (アップロードされた場合)、および証明書に関するメタデータを返します。

#### Note

アップロード後に IAM からプライベートキーをダウンロードまたは取得することはできません。

AWS Tools for Windows PowerShell を使用して証明書を取得するには、[Get-IAMServerCertificate](#) を使用します。

### サーバー証明書の一覧表示 (AWS API)

IAM API を使用して、アップロードされたサーバー証明書を一覧表示するには、[ListServerCertificates](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。



```
aws iam list-server-certificates
```

前のコマンドが成功した場合、各証明書に関するメタデータが含まれた一覧を返します。

AWS Tools for Windows PowerShell を使用して、アップロードしたサーバー証明書を一覧表示するには、[Get-IAMServerCertificates](#) を使用します。

## サーバー証明書のタグ付けとタグの解除 (AWS API)

IAM リソースにタグをアタッチすると、そのリソースへのアクセスを整理および制御できます。IAM API を使用して既存のサーバー証明書にタグ付けするには、[TagServerCertificate](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。

```
aws iam tag-server-certificate --server-certificate-name ExampleCertificate
                                --tags '{"Key": "ExampleKey", "Value":
"ExampleValue"}'
```

上記のコマンドが正常に実行されると、出力は返されません。

IAM API を使用してサーバー証明書のタグを解除するには、[UntagServerCertificate](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。

```
aws iam untag-server-certificate --server-certificate-name ExampleCertificate
                                   --tag-keys ExampleKeyName
```

上記のコマンドが正常に実行されると、出力は返されません。

## サーバー証明書の名前変更またはパスの更新 (AWS API)

IAM API を使用してサーバー証明書の名前を変更するか、パスを更新するには、[UpdateServerCertificate](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。

以下のサンプルコマンドを使用するには、古い証明書の名前、新しい証明書の名前、および証明書のパスを置き換え、1つの連続した行にコマンドを入力します。次の例では、読みやすくするために改行とスペースを追加しています。

```
aws iam update-server-certificate --server-certificate-name ExampleCertificate
                                   --new-server-certificate-name CloudFrontCertificate
```

```
--new-path /cloudfront/
```

前のコマンドが成功した場合、出力を返しません。

AWS Tools for Windows PowerShell を使用してサーバー証明書の名前を変更するか、パスを更新するには、[Update-IAMServerCertificate](#) を使用します。

## サーバー証明書の削除 (AWS API)

IAM API を使用してサーバー証明書を削除するには、[DeleteServerCertificate](#) リクエストを送信します。次の例では、AWS CLI を使用してこのオペレーションを行う方法を示します。

以下のサンプルコマンドを使用するには、*ExampleCertificate* を、削除する証明書の名前に置き換えます。

```
aws iam delete-server-certificate --server-certificate-name ExampleCertificate
```

前のコマンドが成功した場合、出力を返しません。

AWS Tools for Windows PowerShell を使用してサーバー証明書を削除するには、[Remove-IAMServerCertificate](#) を使用します。

## トラブルシューティング

証明書を IAM にアップロードする前に、証明書、プライベートキー、および証明書チェーンがすべて PEM エンコードされていることを確認する必要があります。また、プライベートキーは非暗号化されていることも確認する必要があります。次の例を参照してください。

### Example PEM エンコードされた証明書の例

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

### Example PEM エンコードされ、暗号化されていないプライベートキーの例

```
-----BEGIN RSA PRIVATE KEY-----  
Base64-encoded private key  
-----END RSA PRIVATE KEY-----
```

## Example PEM エンコードされた証明書チェーンの例

証明書チェーンには 1 つまたは複数の証明書が含まれます。テキストエディタ、Windows のコピーコマンド、または Linux の cat コマンドを使用して、ファイルをチェーンに連結します。複数の証明書を含めるときは、各証明書が、前述の証明書を認証する必要があります。そのため、証明書を連結して最後にルート CA 証明書を含めます。

次の例には 3 つの証明書が含まれていますが、証明書チェーンに含まれている証明書はそれ以上またはそれ以下である可能性があります。

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

これらの項目が IAM へのアップロードに適切な形式でない場合は、[OpenSSL](#) を使用して適切な形式に変換できます。

DER から PEM に証明書または証明書チェーンを変換するには

以下の例のように、[OpenSSL x509 コマンド](#)を使用します。次のサンプルコマンドで、*Certificate.der* を、DER エンコードされた証明書を含むファイルの名前に置き換えます。*Certificate.pem* を、希望する出力ファイル名に置き換え、PEM エンコードされた証明書を含めます。

```
openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

DER から PEM にプライベートキーを変換するには

以下の例のように、[OpenSSL rsa コマンド](#)を使用します。次のサンプルコマンドで、*PrivateKey.der* を、DER エンコードされたプライベートキーを含むファイルの名前に置き換えます。*PrivateKey.pem* を、希望する出力ファイル名に置き換え、PEM エンコードされたプライベートキーを含めます。

```
openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

暗号化されたプライベートキーを復号するには (パスワードやパスフレーズを削除)

以下の例のように、[OpenSSL rsa コマンド](#)を使用します。次のサンプルコマンドを使用するには、*EncryptedPrivateKey.pem* を、暗号化されたプライベートキーを含むファイルの名前に置き換えます。*PrivateKey.pem* を、希望する出力ファイル名に置き換え、PEM エンコードおよび非暗号化されたプライベートキーを含めます。

```
openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

証明書バンドルを PKCS#12 (PFX) から PEM に変換するには

以下の例のように、[OpenSSL pkcs12 コマンド](#)を使用します。次のサンプルコマンドで、*CertificateBundle.p12* を、PKCS#12 エンコードされた証明書バンドルを含むファイルの名前に置き換えます。*CertificateBundle.pem* を、希望する出力ファイル名に置き換え、PEM エンコードされた証明書バンドルを含めます。

```
openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

証明書バンドルを PKCS#7 から PEM に変換するには

以下の例のように、[OpenSSL pkcs7 コマンド](#)を使用します。次のサンプルコマンドで、*CertificateBundle.p7b* を、PKCS#7 エンコードされた証明書バンドルを含むファイルの名前に置き換えます。*CertificateBundle.pem* を、希望する出力ファイル名に置き換え、PEM エンコードされた証明書バンドルを含めます。

```
openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

## IAM ユーザーグループ

IAM [ユーザーグループ](#)とは、IAM ユーザーの集合です。ユーザーグループを使用すると、複数のユーザーに対してアクセス許可を指定でき、それらのユーザーのアクセス許可を容易に管理するこ

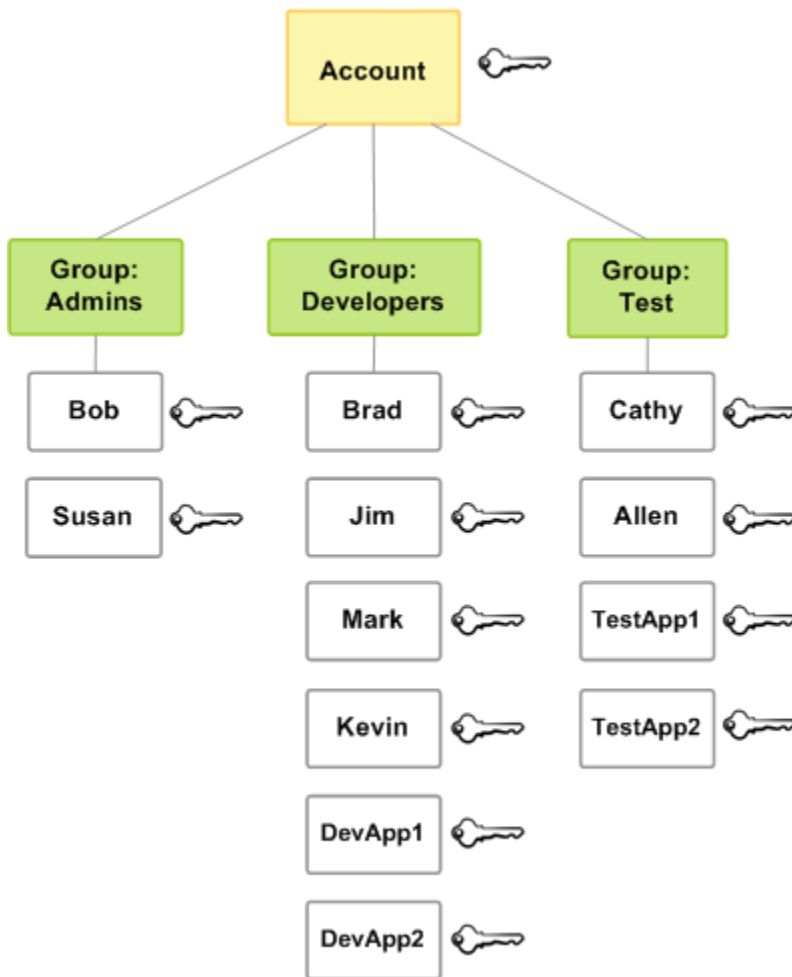
とができます。たとえば、Admins というユーザーグループを作成し、管理者が一般的に必要とするようなアクセス許可をそのユーザーグループに付与できます。そのグループのすべてのユーザーグループは、Admins グループのアクセス許可を自動的に付与されます。管理者権限を必要とする新しいユーザーが組織に参加した場合、そのユーザーを Admins ユーザーグループに追加することで、適切な権限を割り当てることができます。組織内で職務を変更したユーザーがいる場合は、そのユーザーのアクセス許可を編集する代わりに、ユーザーを古いユーザーグループから削除して適切な新しいユーザーグループに追加することができます。

グループ内のすべてのユーザーがそのポリシーのアクセス許可を受け取れるように、ID ベースのポリシーをグループにアタッチすることができます。ポリシー (リソースベースのポリシーなど) では、ユーザーグループを Principal として識別することはできません。これは、グループが認証ではなくアクセス許可に関連しており、プリンシパルが認証済みの IAM エンティティであるためです。ポリシーの詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。

次にユーザーグループの重要な特徴を示します。

- ユーザーグループは多くのユーザーを持つことができ、ユーザーは複数のグループに属することができます。
- ユーザーグループを入れ子形式にすることはできません。ユーザーグループにはユーザーのみを含めることができ、他のグループを含めることはできません。
- AWS アカウント内のユーザーすべてを自動的に含むデフォルトのユーザーグループはありません。このようなユーザーグループが必要な場合は、そのユーザーグループを作成し、新たなユーザーを 1 人 1 人割り当てる必要があります。
- AWS アカウントの IAM リソースの数とサイズは、グループの数や、ユーザーがメンバーになることができるグループの数などには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

以下の図は、小企業の簡単なサンプルを示しています。企業の所有者は、企業の成長に伴い、他のユーザーを作成して管理するための Admins ユーザーグループを作成します。Admins ユーザーグループは、Developers ユーザーグループと Test ユーザーグループを作成します。これらのユーザーグループはそれぞれ、AWS (ジム、ブラッド、DevApp1 など) とやりとりするユーザー (人員とアプリケーション) で構成されます。各ユーザーは、それぞれ一連の認証情報をもっています。この例では、各ユーザーは 1 つのユーザーグループに属しています。しかし、ユーザーは複数のユーザーグループに属することができます。



## IAM ユーザーグループの作成

### Note

[ベストプラクティス](#)として、一時的な認証情報の使用により AWS にアクセスするには、人間のユーザーに対して ID プロバイダーとのフェデレーションの使用を必須とすることをお勧めします。ベストプラクティスに従えば、IAM ユーザーやグループを管理することにはなりません。管理するのではなく、ユーザーとグループは AWS の外部で管理され、フェデレーション ID として AWS リソースにアクセスできます。フェデレーション ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、Identity Center ディレクトリのユーザー、または ID ソースから提供された認証情報を使用して AWS のサービスにアクセスするユーザーです。フェデレーション ID は、ID プロバイダーが定義したグループを使用します。AWS IAM Identity Center を使用している場合は、IAM Identity Center でのユーザーとグループの作成に関する情報について、AWS IAM


Identity Center ユーザーガイドの「[Manage identities in IAM Identity Center](#)」(IAM Identity Center での ID の管理) を参照してください。

ユーザーグループをセットアップするには、グループを作成する必要があります。次に、グループのユーザーに期待される作業のタイプに基づいて、グループにアクセス許可を付与します。最後に、ユーザーをグループに追加します。

ユーザーの作成に必要なアクセス許可の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

IAM グループを作成してポリシーをアタッチするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ]、[グループの作成] の順に選択します。
3. [ユーザーグループ名] に、グループ名を入力します。

 Note

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。グループ名は、最大 128 文字の英数字、プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (\_)、ハイフン (-) を組み合わせて指定します。名前はアカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、**ADMINS** と **admins** というロール名を両方作成することはできません。

4. ユーザーのリストで、グループに追加する各ユーザーのチェックボックスをオンにします。
5. ポリシーのリストで、グループのすべてのメンバーに適用する各ポリシーのチェックボックスをオンにします。
6. [Create group] (グループの作成) を選択します。

IAM ユーザーグループを作成するには (AWS CLI または AWS API)

以下のいずれかを使用します。

- AWS CLI: [aws iam create-group](#)
- AWS API: [CreateGroup](#)



## IAM ユーザーグループの管理

Amazon Web Services には、IAM ユーザーグループを管理するための複数のツールが用意されています。ユーザーグループ内のユーザーを追加または削除するために必要な権限の詳細については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。

### トピック

- [IAM ユーザーグループのリストの取得](#)
- [IAM グループへのユーザーの追加と削除](#)
- [IAM ユーザーグループへのポリシーのアタッチ](#)
- [IAM ユーザーグループの名前の変更](#)
- [IAM ユーザーグループの削除](#)

### IAM ユーザーグループのリストの取得

アカウント内のすべてのユーザーグループのリスト、ユーザーグループ内のユーザーのリスト、およびユーザーが属するユーザーグループのリストを取得できます。AWS CLI または AWS API を使用する場合は、特定のパスプレフィックスが付いたすべてのユーザーグループのリストを取得できます

アカウントの全ユーザーグループのリストを取得するには

次のいずれかを実行します。

- [AWS Management Console](#): ナビゲーションペインで、[ユーザーグループ] を選択します。
- AWS CLI: [aws iam list-groups](#)
- AWS API: [ListGroup](#)s

特定のユーザーグループのユーザーを一覧表示するには

次のいずれかを実行します。

- [AWS Management Console](#): ナビゲーションペインで [ユーザーグループ] を選択し、グループ名を選択してから、[ユーザー] タブを選択します。
- AWS CLI: [aws iam get-group](#)
- AWS API: [GetGroup](#)

ユーザーが所属しているすべてのユーザーグループを一覧表示するには

次のいずれかを実行します。

- [AWS Management Console](#): ナビゲーションペインで [ユーザー] を選択し、ユーザー名を選択してから、[グループ] タブを選択します。
- AWS CLI: [aws iam list-groups-for-user](#)
- AWS API: [ListGroupsWithUser](#)

## IAM グループへのユーザーの追加と削除

ユーザーグループを使用して、同じアクセス許可ポリシーを複数のユーザーに一度に適用します。適用後、IAM ユーザーグループとの間でいつでもユーザーを追加または削除できます。これは、組織で従業員の異動があるときに便利です。

### ポリシーアクセスの確認

ポリシーのアクセス許可を変更する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

### ユーザーグループのユーザーを追加または削除する (コンソール)

AWS Management Console を使用して、ユーザーグループのユーザーを追加または削除できます。

### IAM ユーザーグループにユーザーを追加するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ] を選択してから、グループ名を選択します。
3. [Users (ユーザー)] タブを選択し、[Add users (ユーザーの追加)] を選択します。追加するユーザーの横のチェックボックスをオンにします。
4. [ユーザーの追加] を選択します。

### IAM グループからユーザーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. ナビゲーションペインで、[ユーザーグループ] を選択してから、グループ名を選択します。
3. [Users] (ユーザー) タブを選択します。削除するユーザーの横にあるチェックボックスをオンにし、[ユーザーの削除]を選択します。

ユーザーグループのユーザーを追加または削除する (AWS CLI)

AWS CLI を使用して、ユーザーグループのユーザーを追加または削除できます。

IAM ユーザーグループにユーザーを追加するには (AWS CLI)

- 以下のコマンドを使用します。
  - [aws iam add-user-to-group](#)

IAM ユーザーグループからユーザーを削除するには (AWS CLI)

- 以下のコマンドを使用します。
  - [aws iam remove-user-from-group](#)

ユーザーグループのユーザーを追加または削除する (AWS API)

AWS API を使用して、ユーザーグループのユーザーを追加または削除できます。

IAM グループにユーザーを追加するには (AWS API)

- 以下のオペレーションを実行します。
  - [AddUserToGroup](#)

IAM ユーザーグループからユーザーを削除するには (AWS API)

- 以下のオペレーションを実行します。
  - [RemoveUserFromGroup](#)

## IAM ユーザーグループへのポリシーのタッチ

[AWS 管理ポリシー](#)をアタッチできます — つまり、次の手順で説明するように、AWS によって提供される事前記述されたポリシーをユーザーグループにアタッチできます。カスタマー管理ポリシー (独自に作成したカスタムのアクセス権限を使用するポリシー) をアタッチするには、まずポリシーを作成する必要があります。カスタマー管理ポリシーの作成方法については、「[IAM ポリシーの作成](#)」を参照してください。

アクセス許可とポリシーの詳細については、「[AWS リソースのアクセス管理](#)」を参照してください。

ポリシーをユーザーグループにアタッチするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[User groups (ユーザーグループ)] を選択してから、グループ名を選択します。
3. [Permissions (アクセス許可)] タブを選択します。
4. [アクセス許可を追加]、[ポリシーをアタッチ] の順に選択します。
5. ユーザーグループにアタッチされている現在のポリシーは、現在のアクセス権限ポリシーのリストに表示されます。その他のアクセス権限ポリシーのリストで、アタッチするポリシーの名前の横にあるチェックボックスをオンにします。検索ボックスを使用して、ポリシーのリストをタイプとポリシー名でフィルタリングできます。
6. IAM ユーザーグループにアタッチするポリシーを選択して、[ポリシーをアタッチ] をクリックします。

グループにポリシーをアタッチするには (AWS CLI または AWS API)

次のいずれかを実行します。

- AWS CLI: [aws iam attach-group-policy](#)
- AWS API: [AttachGroupPolicy](#)

## IAM ユーザーグループの名前の変更

ユーザーグループ名またはパスを変更すると、以下のことが起こります。

- ユーザーグループにアタッチされているポリシーは、新しい名前のグループにそのままアタッチされています。
- ユーザーグループは名前が変わるだけで、保持するユーザーは一切変わりません。
- ユーザーグループの一意の ID は変更されません。一意の ID の詳細については、「[一意の識別子](#)」を参照してください。

IAM は、新しい名前を使用するためのリソースとしてユーザーグループを参照するポリシーを自動的に更新しません。したがって、ユーザーグループの名前を変更するときには注意が必要です。ユーザーグループの名前を変更する前に、すべてのポリシーを手動でチェックし、このユーザーグループの名前を参照しているポリシーを見つける必要があります。たとえば、Bob は組織のテスト部門のマネージャーであるとします。Bob の IAM ユーザーエンティティにアタッチされたポリシーにより、Bob はテストユーザーグループに対してユーザーを追加および削除できます。管理者は、ユーザーグループの名前 (またはグループのパス) を変更する場合、Bob にアタッチされているポリシーも更新して新しい名前またはパスを反映する必要があります。この更新を行わないと、Bob はユーザーグループに対してユーザーを追加および削除できるようになりません。

ユーザーグループをリソースとして参照しているポリシーを見つけるには:

1. IAM コンソールのナビゲーションペインから、[ポリシー] を選択します。
2. [タイプ]列で並べ替えて、カスタマー管理するカスタムポリシーを見つけます。
3. [ポリシーの編集] を選択して、ポリシー内のユーザーグループの名前を変更します。
4. [アクセス許可] タブを選択し、[概要] を選択します。
5. サービスのリストに [IAM] があれば、それを選択します。
6. [リソース] 列でユーザーグループの名前を見つけます。
7. [編集] を選択して、ポリシー内のユーザーグループの名前を変更します。

IAM ユーザーグループの名前を変更するには

次のいずれかを実行します。

- [AWS Management Console](#): ナビゲーションペインで、[ユーザーグループ] を選択してから、グループ名を選択します。Edit (編集) を選択します。新しいユーザーグループ名を入力し、[変更の保存] を選択します。。
- AWS CLI: [aws iam update-group](#)
- AWS API: [UpdateGroup](#)

## IAM ユーザーグループの削除

AWS Management Consoleのユーザーグループを削除すると、コンソールは自動的にすべてのグループメンバーを削除し、アタッチされていた管理ポリシーはすべてデタッチされて、すべてのインラインポリシーが削除されます。ただし、IAM では、ユーザーグループをリソースとして参照しているポリシーは自動的に削除されません。ユーザーグループを削除する際は注意してください。ユーザーグループを削除する前に、すべてのポリシーを手動でチェックして、このグループの名前を参照しているポリシーを見つける必要があります。例えば、テストチームマネージャーの John は、IAM ユーザーエンティティにアタッチされたポリシーの権限を持ち、テストユーザーグループに対してユーザーを追加および削除できます。管理者は、ユーザーグループを削除する場合に、John にアタッチされているポリシーも削除する必要があります。それ以外の場合、管理者が削除したグループを再作成して同じ名前を付けると、テストチームを辞めても John の権限はそのまま残ります。

ユーザーグループをリソースとして参照しているポリシーを見つけるには

1. IAM コンソールのナビゲーションペインから、[ポリシー] を選択します。
2. [タイプ]列で並べ替えて、カスタマー管理するカスタムポリシーを見つけます。
3. 削除するポリシーのポリシー名を選択します。
4. [アクセス許可] タブを選択し、[概要] を選択します。
5. サービスのリストに [IAM] があれば、それを選択します。
6. [リソース] 列でユーザーグループの名前を見つけます。
7. [削除] を選択してポリシーを削除します。
8. ポリシー名を入力してポリシーの削除を確認し、[削除] を選択します。

一方、AWS CLI、Tools for Windows PowerShell、または AWS API を使用してユーザーグループを削除する場合は、最初にグループからユーザーを削除する必要があります。ユーザーグループを入れ子形式にすることはできません。ユーザーグループにはユーザーのみを含めることができ、他のグループを含めることはできません。さらに、グループにアタッチされているすべての管理ポリシーをデタッチします。その後、ユーザーグループ自体を削除できます。

### IAM ユーザーグループの削除 (コンソール)

AWS Management Console から IAM ユーザーグループを削除できます。

## IAM ユーザーグループを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ]を選択します。
3. ユーザーグループのリストで、削除するユーザーグループの名前の横にあるチェックボックスをオンにします。検索ボックスを使用して、ユーザーグループのリストをタイプ、パーミッション、およびユーザーグループ名でフィルタリングできます。
4. [Delete] (削除) をクリックします。
5. 1つのユーザーグループを削除する場合は、確認ボックスにユーザーグループ名を入力し、[削除]を選択します。複数のユーザーグループを削除する場合は、削除するユーザーグループの数を入力し、**user groups** に続いて削除を選択します。たとえば、3つのユーザーグループを削除する場合は、**3 user groups** を入力します。

## IAM ユーザーグループの削除 (AWS CLI)

AWS CLI から IAM ユーザーグループを削除できます。

### IAM ユーザーグループを削除するには (AWS CLI)

1. ユーザーグループからすべてのユーザーを削除します。
  - [aws iam get-group](#) (ユーザーグループのユーザーの一覧表示) および [aws iam remove-user-from-group](#) (ユーザーグループからのユーザーの削除)
2. ユーザーグループに組み込まれたインラインポリシーをすべて削除します。
  - [aws iam list-group-policies](#) (ユーザーグループのインラインポリシーの一覧表示) および [aws iam delete-group-policy](#) (ユーザーグループのインラインポリシーの削除)
3. ユーザーグループにアタッチされた管理ポリシーをすべてデタッチします。
  - [aws iam list-attached-group-policies](#) (ユーザーグループにアタッチされている管理ポリシーの一覧表示) および [aws iam detach-group-policy](#) (ユーザーグループからの管理ポリシーのデタッチ)
4. ユーザーグループを削除します。
  - [aws iam delete-group](#)



## IAM ユーザーグループの削除 (AWS API)

AWS API を使用して IAM ユーザーグループを削除できます。

IAM ユーザーグループを削除するには (AWS API)

1. ユーザーグループからすべてのユーザーを削除します。
  - [GetGroup](#) (ユーザーグループのユーザーの一覧表示) および [RemoveUserFromGroup](#) (ユーザーグループからのユーザーの削除)
2. ユーザーグループに組み込まれたインラインポリシーをすべて削除します。
  - [ListGroupPolicies](#) (ユーザーグループのインラインポリシーの一覧表示) および [DeleteGroupPolicy](#) (ユーザーグループからのインラインポリシーの削除)
3. ユーザーグループにアタッチされた管理ポリシーをすべてデタッチします。
  - [ListAttachedGroupPolicies](#) (ユーザーグループにアタッチされている管理ポリシーの一覧表示) および [DetachGroupPolicy](#) (ユーザーグループからの管理ポリシーのデタッチ)
4. ユーザーグループを削除します。
  - [DeleteGroup](#)

## IAM ロール

IAM ロールは、特定の許可があり、アカウントで作成できるもう 1 つの IAM アイデンティティです。IAM ロールは、アイデンティティが AWS で実行できることとできないことを決定するアクセス許可ポリシーを持つ AWS アイデンティティであるという点で IAM ユーザーと似ています。ただし、ユーザーは 1 人の特定の人に一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。また、ロールには標準の長期認証情報 (パスワードやアクセスキーなど) も関連付けられません。代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。

ロールを使用して、通常は AWS リソースへのアクセス権のないユーザー、アプリケーション、サービスにそのアクセス権を委任できます。例えば、AWS アカウントのユーザーに、通常はないリソースに対するアクセス許可を付与したり、ある AWS アカウントのユーザーに、別のアカウントのリソースに対するアクセス許可を付与したりできます。または、モバイルアプリに AWS リソースの使用を許可しても、(キーの更新が困難であり、ユーザーがキーの抽出を行える可能性がある) アプリへの AWS キーの埋め込みは禁止すべきである場合があります。AWS の外部 (社内ディレクトリなど)

に ID をすでに持っているユーザーに AWS へのアクセスを許可することが必要になる場合があります。または、リソースを監査できるように、アカウントへのアクセス権を第三者に付与することが必要になる場合もあります。

これらのシナリオでは、IAM ロールを使用して AWS リソースにアクセスを委任できます。このセクションでは、ロールの概要とさまざまな使用方法、適切なアプローチを選択するタイミングと方法、ロールの作成、管理、切り替え（または引き受け）、削除を行う方法について説明します。

### Note

AWS アカウント を初めて作成するとき、デフォルトではロールは作成されません。アカウントにサービスを追加すると、そのユースケースをサポートするためにサービスにリンクされたロールが追加される場合があります。

サービスにリンクされたロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。

サービスにリンクされたロールを削除する前に、最初に関連するリソースを削除する必要があります。これにより、リソースにアクセスするための許可を意図せず削除することが防止され、リソースが保護されます。

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

## トピック

- [ロールに関する用語と概念](#)
- [ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス](#)
- [サービスリンクロールの使用](#)
- [IAM ロールの作成](#)
- [IAM ロールを使用する](#)
- [IAM ロールの管理](#)

## ロールに関する用語と概念

ロールの操作に役立つ基本的な用語を紹介します。

### ロール

特定のアクセス権限を持ち、アカウントで作成できる IAM アイデンティティです。IAM ロールは、IAM ユーザーといくつかの類似点を持っています。ロールとユーザーは、両方とも、ID が AWS でできることとできないことを決定するアクセス許可ポリシーを持つ AWS ID です。ただし、ユーザーは 1 人の特定の一人に一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。また、ロールには標準の長期認証情報 (パスワードやアクセスキーなど) も関連付けられません。代わりに、ロールを引き受けると、ロールセッション用の一時的なセキュリティ認証情報が提供されます。

ロールを使用できるのは、以下のものです。

- ロールと同じ AWS アカウントの IAM ユーザー
- ロールとは異なる AWS アカウントの IAM ユーザー
- Amazon Elastic Compute Cloud (Amazon EC2) などの AWS が提供するウェブサービス
- SAML 2.0 または OpenID Connect と互換性のある外部 ID プロバイダー (IdP) サービスによって認証される外部ユーザー、またはカスタム作成された ID ブローカー。

### AWS サービス ロール

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#) です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。

### EC2 インスタンスの AWS サービスロール

アプリケーションが Amazon EC2 インスタンス上で実行しているサービスロールの特殊なタイプは、アカウントでアクションの実行を引き受けることができます。このロールは EC2 インスタンスにその起動時に割り当てられます。このインスタンスで実行しているアプリケーションは一時的なセキュリティ認証情報を取得でき、ロールが許可するアクションを実行できます。EC2 インスタンスのサービスロールの使用の詳細については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

### AWS サービスにリンクされたロール

サービスにリンクされたロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになりま

す。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。

#### Note

サービスにリンクされたロールのサポートを開始する時点ですでにサービスを使用している場合は、アカウントの新しいロールに関する E メールが送信されることがあります。この場合、サービスにリンクされたロールは、サービスによって自動的にアカウントに作成されています。このロールをサポートするために必要な操作はありません。また、手動でロールを削除しないでください。詳細については、「[AWS アカウントに新しいロールが表示される](#)」を参照してください。

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。詳細については、「[サービスリンクロールの使用](#)」を参照してください。

## ロールの連鎖

ロールの連鎖は、AWS CLI または API を使用して 2 つ目のロールを引き受けるロールを使用する場合に発生します。たとえば、RoleA には RoleB を引き受けるアクセス権があります。User1 は AssumeRole API オペレーションの長期的なユーザー認証情報を使用して RoleA を引き受けることができます。このオペレーションを用いて RoleA の短期的な認証情報を返します。ロールが連鎖していれば、User1 によって RoleB が引き受けられるよう RoleA の短期的な認証情報を使用できます。

ロールを引き受けるときは、セッションタグを渡して、タグを推移的に設定できます。推移的なセッションタグは、ロールの連鎖内の後続のすべてのセッションに渡されます。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

ロールの連鎖では、AWS CLI または AWS API ロールセッションは最長 1 時間に制限されます。[AssumeRole](#) API オペレーションを使用してロールを引き受ける場合は、DurationSeconds パラメータを使用してロールセッションの期間を指定できます。パラメータの値は、[ロールの最大セッション期間設定](#)によって、最大 43200 秒 (12 時間) まで指定できます。ただし、ロールの連鎖を使用してロールを引き受ける場合、DurationSeconds パラメータ値で 1 時間を超える値を指定すると、オペレーションは失敗します。

AWS では、ロールチェーンとして [EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与](#)するためのロールの使用は処理されません。

## 委任

委任により、制御するリソースへのアクセスを許可するユーザーにアクセス許可を付与できます。委任には、2つのアカウント間の信頼を設定することが含まれます。1つ目は、リソースを所有するアカウント（信頼するアカウント）です。2つ目は、リソース（信頼されたアカウント）にアクセスする必要があるユーザーを含むアカウントです。信頼するアカウントと信頼されたアカウントには、以下のいずれかを指定できます。

- 同じアカウント。
- 組織の制御下にある別々のアカウント。
- 異なる組織によって所有される2つのアカウント。

リソースにアクセスする権限を委任するには、2つの[ポリシー](#)がアタッチされた信頼されるアカウントの [IAM ロールを作成](#)します。アクセス許可ポリシーは、リソースに対して目的のタスクを実行するために必要なアクセス許可をロールのユーザーに付与します。信頼ポリシーは、信頼されたアカウントのどのメンバーにロールを割り当てることができるかを指定します。

信頼ポリシーを作成する際、プリンシパル要素およびプリンシパル要素内の ARN の一部としてワイルドカード (\*) を指定することはできません。信頼ポリシーは、信頼するアカウントのロールにアタッチされ、アクセス許可の半分です。残りの半分は、[ユーザーがロールを切り替えることができる、またはロールを引き受けることができる](#)信頼されたアカウントのユーザーにアタッチされたアクセス許可ポリシーです。ロールを引き受けるユーザーの各自のアクセス権限は一時的に無効になりますが、その代わりにロールのアクセス権限を取得します。ユーザーがロールの使用を終了または停止すると、元のユーザーアクセス権限に戻ります。[外部 ID](#) という追加パラメータは、同じ組織がコントロールしていないアカウント間でロールを安全に利用するのに役立ちます。

## フェデレーション

外部 ID プロバイダーと AWS との間に信頼関係を作成することです。ユーザーは OIDC プロバイダー (Login with Amazon、Facebook、Google など)、または OpenID Connect (OIDC) 互換の任意の IdP にサインインできます。また、Microsoft Active Directory フェデレーションサービスなど、Security Assertion Markup Language (SAML) 2.0 互換の企業の ID システムにサインインすることもできます。OIDC および SAML 2.0 を使用して、このような外部 ID プロバイダーと AWS との信頼関係を設定する場合、ユーザーは IAM ロールに割り当てられます。ユーザーは一時的な認証情報を受け取り、これを使用して AWS リソースにアクセスすることもできます。

## 委任ユーザー

IAM ユーザーを作成する代わりに、AWS Directory Service、エンタープライズユーザーディレクトリ、または OIDC プロバイダーの既存の ID を使用できます。このようなユーザーはフェデレーションユーザーと呼ばれます。AWS では、[ID プロバイダー](#)を通じてアクセスがリクエストされたとき、フェデレーションユーザーにロールを割り当てます。フェデレーテッドユーザーの詳細については、「[フェデレーテッドユーザーとロール](#)」を参照してください。

## 信頼ポリシー

[JSON ポリシードキュメント](#)では、ロールを引き受けるために信頼するプリンシパルを定義します。ロール信頼ポリシーは、IAMのロールに関連付けられている必須の[リソースベースのポリシー](#)です。信頼ポリシーで指定できる[プリンシパル](#)には、ユーザー、ロール、アカウント、およびサービスが含まれます。

## アクセス許可ポリシー

ロールで使用できるアクションやリソースを定義する [JSON](#) 形式のアクセス許可に関するドキュメント。ドキュメントは [IAM ポリシー言語](#)のルールに従って記述されます。

## アクセス許可の境界

ポリシーを使用して、アイデンティティベースのポリシーがロールに付与できるアクセス許可の上限を設定する高度な機能です。アクセス許可の境界をサービスにリンクされたロールに適用することはできません。詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。

## プリンシパル

アクションを実行してリソースにアクセスできる AWS 内のエンティティです。プリンシパルは、AWS アカウントのルートユーザー、IAM ユーザー、またはロールをにすることができます。以下の 2 つの方法のいずれかを使用して、リソースに対するアクセス許可を付与できます。

- ユーザー（直接、またはグループ経由で間接的に）またはロールに対し、アクセス権限ポリシーをアタッチすることができます。
- [リソースベースのポリシー](#)をサポートするサービスについては、リソースにアタッチされているポリシーの Principal 要素でプリンシパルを指定できます。

AWS アカウント をプリンシパルにする場合、通常はアカウント内で定義されているすべてのプリンシパルが対象となります。



**Note**

ワイルドカード (\*) を使用して、ロールの信頼ポリシー内のプリンシパル名または ARN の一部に致させることはできません。詳細については、「[AWS JSON ポリシーの要素: Principal](#)」を参照してください。

## クロスアカウントアクセスのロール

あるアカウントのリソースに対するアクセス権限を、別のアカウントの信頼されるプリンシパルに付与するロール。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の AWS サービスでは、リソースにポリシーを直接アタッチすることができます (ロールをプロキシとして使用する代わりに)。これらはリソースベースのポリシーと呼ばれ、別の AWS アカウントのプリンシパルにリソースへのアクセスを許可するために使用できます。これらのリソースには、Amazon Simple Storage Service (S3) バケット、S3 Glacier ポールト、Amazon Simple Notification Service (SNS) トピック、Amazon Simple Queue Service (SQS) キューなどがあります。リソースベースのポリシーをサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。リソースベースのポリシーの詳細については、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

## ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス

多くの AWS 機能と同様に、通常ロールを使用する方法には、IAM コンソールでインタラクティブに使用する方法と、AWS CLI、Tools for Windows PowerShell、API のいずれかを使用してプログラムで使用する方法の 2 つがあります。

- IAM コンソールを使用するアカウントの IAM ユーザーは、別のロールに切り替えて、コンソールでそのロールのアクセス許可を一時的に使用できます。ユーザーは、元のアクセス権限を返却し、そのロールに割り当てられたアクセス権限を取得します。ユーザーがそのロールを終了すると、元のアクセス権限に戻ります。
- アプリケーションまたは AWS によって提供されるサービス (Amazon EC2 など) は、AWS にプログラムによるリクエストを行うためのロールの一時的セキュリティ認証情報をリクエストすることで、ロールを引き受けることができます。ロールをこのように使用することで、リソースへのアクセスが必要なエンティティごとに長期的なセキュリティ認証情報を共有または保持する (IAM ユーザーを作成するなどして) 必要がなくなります。



**Note**

このガイドでは、ロールの切り替えおよびロールの引き受けという言葉、ほとんど同じ意味で使用しています。

ロールを使用する最もシンプルな方法は、自分のアカウントまたは別の AWS アカウント 内に作成したロールに切り替えるアクセス許可を IAM ユーザーに付与する方法です。IAM ユーザーは、コンソールを使用して簡単にロールを切り替え、通常は必要ないアクセス許可を使用することができます。その後、ロールを終了して、それらのアクセス許可を返却できます。これは、機密性の高いリソースに誤ってアクセスしたり変更したりすることを回避するのに役立ちます。

アプリケーションとサービスへのアクセスを許可する (つまり、外部フェデレーションユーザー) など、ロールを複雑な方法で使用するには、AssumeRole API を呼び出します。この API 呼び出しは、アプリケーションが後続の API 呼び出しで使用できる一時的な認証情報のセットを返します。一時的な認証情報を使用して試行されたアクションは、関連付けられたロールによって付与されたアクセス権限のみを使用します。アプリケーションは、ユーザーがコンソールで行った方法でロールを "終了" する必要はありません。アプリケーションは一時的な認証情報を使用して停止し、元の認証情報で呼び出しを再開するだけです。

フェデレーテッドユーザーは、ID プロバイダー (IdP) から提供される認証情報を使用してサインインします。その後、AWS は一時的な認証情報を信頼された IdP に提供し、後続の AWS リソースリクエストに含めることができるようにユーザーに渡します。これらの認証情報は、割り当てられたロールに付与されたアクセス権限を提供します。

このセクションは、次のシナリオの概要を提供します。

- [ユーザーが所有する別のアカウントのリソースにアクセスできるように、ユーザーが所有する 1 つの AWS アカウント の IAM ユーザーにアクセスを許可する](#)
- [AWS ワークロード以外へのアクセスを提供する](#)
- [サードパーティーが所有する AWS アカウント の IAM ユーザーへのアクセスを許可する](#)
- [AWS が提供するサービスに AWS リソースへのアクセスを許可する](#)
- [外部で認証された \(ID フェデレーション\) ユーザーにアクセスを許可する](#)

## 所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供

IAM ユーザーには、AWS アカウント 内のロール、または所有する他の AWS アカウント で定義されたロールに切り替えるアクセス許可を付与することができます。

### Note

お客様が所有または制御していないアカウントへのアクセス権を付与する場合は、このトピックの「[第三者が所有する AWS アカウント へのアクセス権を付与する](#)」を参照してください。

組織の業務に不可欠な Amazon EC2 インスタンスがあるとします。このインスタンスを終了するためのアクセス許可をユーザーに直接付与する代わりに、これらの権限を持つロールを作成できます。次に、インスタンスを終了する必要があるときに、このロールに切り替えることを管理者に許可します。これにより、インスタンスに次の保護レイヤーが追加されます。

- ユーザーにロールを引き受けるアクセス権限を明示的に付与する必要があります。
- ユーザーは、アクティブに AWS Management Console を使用してロールに切り替えるか、AWS CLI または AWS API を使用してロールを引き受ける必要があります。
- MFA デバイスでサインインしているユーザーのみがロールを引き受けることができるように、ロールに多要素認証 (MFA) 保護を追加できます。ロールを引き受けるユーザーが最初に多要素認証 (MFA) を使用して認証されるようにロールを設定する方法については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

このアプローチを使用して最小権限の原則を強制することをお勧めします。つまり、昇格されたアクセス許可の使用は、特定のタスクに必要なときのみ制限されます。ロールを使用すると、機密性の高い環境が誤って変更されるのを防ぐことができます (特に、ロールが必要なときだけ使用されるように[監査](#)と組み合わせている場合)。

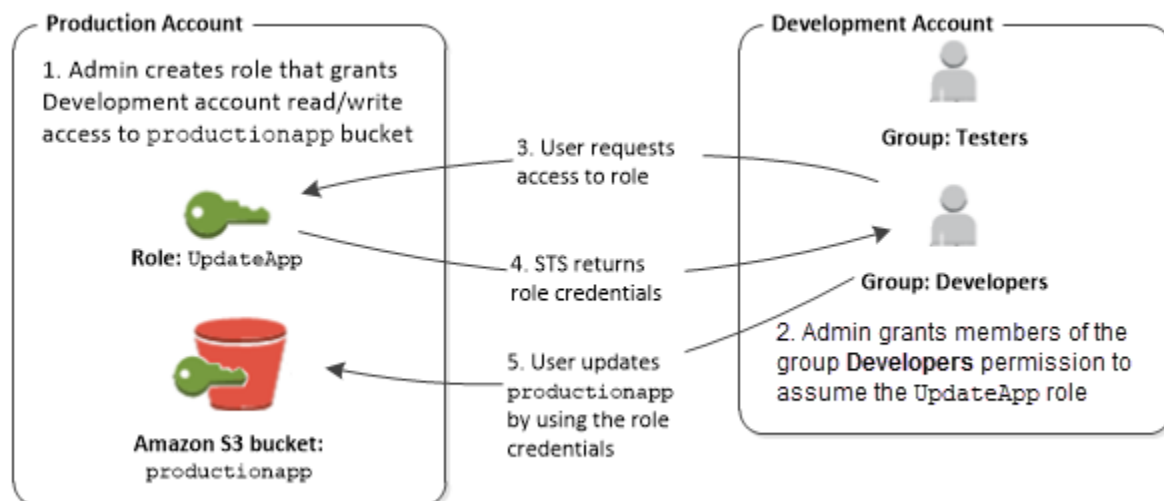
この目的でロールを作成する場合、ユーザーがロールの信頼ポリシーの Principal 要素にアクセスする必要のあるアカウントを ID で指定します。その後、その他のアカウントの特定のユーザーに、そのロールに切り替えるためのアクセス権限を付与できます。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

あるアカウントのユーザーは、同じアカウントまたは別のアカウントのロールに切り替えることができます。そのロールを使用している間、ユーザーはアクションだけを実行して、ロールによって許

可されているリソースのみにアクセスできますが、元のユーザーアクセス権限は停止されます。ユーザーがそのロールを終了すると、元のユーザーのアクセス権限に戻ります。

### 個別の開発用アカウントと本稼働用アカウントを使用したシナリオ例

本稼働環境から開発環境を分離するための複数の AWS アカウント が組織に存在するとします。開発アカウントのユーザーは、本番稼働用アカウントでリソースにアクセスすることが必要になる場合があります。たとえば、開発環境から本番稼働環境に更新を昇格させる場合、クロスアカウントアクセスが必要になることがあります。両方のアカウントで作業するユーザーにそれぞれの ID ( およびパスワード ) を作成することも可能ですが、複数のアカウントに対する複数の認証情報を管理する必要があり、ID 管理が困難になります。以下の図では、すべてのユーザーは開発用アカウントで管理されていますが、数名の開発者は本稼働用アカウントに制限付きでアクセスする必要があります。開発用アカウントにはテスターと開発者の 2 つのグループがあり、それぞれ個別のポリシーがあります。



1. 管理者は、本番稼働用アカウントで IAM を使用し、このアカウントで UpdateApp ロールを作成します。ロールでは、管理者は開発用アカウントを Principal として指定する信頼ポリシーを定義します。これは、開発用アカウントの認証されたユーザーが UpdateApp ロールを使用できることを意味します。また、管理者は、productionapp という Amazon S3 バケットに対する読み取りと書き込みを指定するロールについてのアクセス許可ポリシーを定義します。

次に、管理者は、このロールを引き受ける必要がある任意のユーザーと該当情報を共有します。この情報は、ロールのアカウント番号と名前 (AWS コンソールユーザーの場合)、または Amazon リソースネーム (ARN) (AWS CLI または AWS API アクセスの場合) です。ロールの ARN は `arn:aws:iam::123456789012:role/UpdateApp` のようになります。これは、ロールの名前が UpdateApp で、ロールがアカウント番号 123456789012 に作成されたことを意味します。

**Note**

管理者は、ロールを引き受けるユーザーが最初に多要素認証 (MFA) を使用して認証されるように、オプションでロールを設定できます。詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

- 開発アカウントでは、管理者は開発者グループのメンバーに対して、このロールに切り替えるアクセス許可を付与します。これは、Developers グループに AWS Security Token Service ( AWS STS ) UpdateApp ロールの AssumeRole API を呼び出すアクセス権限を付与することで行われます。これにより、開発用アカウントの Developers グループに所属する IAM ユーザーは、本稼働用アカウントの UpdateApp ロールに切り替えることができます。Developers グループに所属しない他のユーザーには、そのロールに切り替えるアクセス許可がないため、本番稼働用アカウントの S3 バケットにはアクセスできません。
- ユーザーは、ロールへの切り替えをリクエストします。
  - AWS コンソール: ユーザーはナビゲーションバーのアカウント名を選択してから、[ロールの切り替え] を選択します。ユーザーは、アカウント ID ( またはエイリアス ) とロール名を指定します。または、管理者からメールで送信されたリンクをクリックすることもできます。リンクをクリックすると、詳細がすでに入力された [ロールの切り替え] ページに移動します。
  - AWS API/AWS CLI: 開発用アカウントの Developers グループに所属するユーザーが AssumeRole 関数を呼び出し、UpdateApp ロールの認証情報を取得します。呼び出しの一部として、ユーザーが UpdateApp ロールの ARN を指定します。Testers グループのユーザーが同じ要求をしても、テスターは UpdateApp ロールの ARN に対して AssumeRole を呼び出すことは許可されていないため、その要求は拒否されます。
- AWS STS が一時的な認証情報を返します。
  - AWS コンソール: AWS STS はリクエストをロールの信頼ポリシーと照合し、そのリクエストが信頼されたエンティティ (開発用アカウント) からであることを確認します。照合の後、AWS STS は AWS コンソールに [一時的セキュリティ認証情報](#) を返します。
  - API/CLI: AWS STS は、リクエストをロールの信頼ポリシーと照合し、信頼されたエンティティ (Development アカウント) からであることを確認します。照合の後、AWS STS はアプリケーションに [一時的セキュリティ認証情報](#) を返します。
- 一時的な認証情報により、AWS リソースにアクセスすることができます。
  - AWS コンソール: AWS コンソールは、後続のすべてのコンソールアクション (この場合は、productionapp バケットの読み書き) でユーザーの代わりに一時的な認証情報を使用します。コンソールは、本稼働用アカウントにある他のリソースにはアクセスできません。ユー

ザーがロールを終了すると、ユーザーのアクセス権限がロールに切り替える前に持っていた元のアクセス権限に戻ります。

- API/CLI: アプリケーションは、その一時的な認証情報を使用して productionapp バケットを更新します。一時的な認証情報を使用し、アプリケーションは productionapp バケットでのみ読み書きを行うことができますが、本番稼働用アカウントにあるその他のリソースにはアクセスできません。アプリケーションは、ロールを終了する必要はありませんが、その代わりに一時的な認証情報の使用を停止し、後続の API 呼び出しで元の認証情報の使用する必要があります。

## 詳細情報

詳細については、次を参照してください。

- [IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)

## AWS ワークロード以外へのアクセスを提供する

[IAM ロール](#)は、[アクセス許可](#)が割り当てられている AWS Identity and Access Management IAM 内のオブジェクトです。IAM の ID または AWS 外部からの ID を使用して[そのロールを引き受ける](#)と、ロールセッションのための一時的なセキュリティ認証情報が提供されます。データセンターで稼働しているワークロードや、AWS の外部のインフラストラクチャで AWS リソースにアクセスする必要のあるワークロードがある場合があります。長期的なアクセスキーを作成、配布、管理する代わりに、AWS Identity and Access Management Roles Anywhere (IAM Roles Anywhere) を使用して AWS 以外のワークロードを認証することができます。IAM Roles Anywhere は、認証局 (CA) から発行された X.509 証明書を使用して ID を認証し、IAM ロールによって提供される一時的な認証情報を使用した AWS のサービス への安全なアクセスを提供します。

IAM Roles Anywhere を使用するには、[AWS Private Certificate Authority](#) を使用して CA をセットアップするか、独自の PKI インフラストラクチャの CA を使用します。CA をセットアップしたら、IAM Roles Anywhere で信頼アンカーと呼ばれるオブジェクトを作成し、IAM Roles Anywhere と CA の間で認証のための信頼を確立します。その後、既存の IAM ロールを設定するか、IAM Roles Anywhere サービスを信頼する新しいロールを作成できます。AWS 以外のワークロードが信頼アンカーを使用して IAM Roles Anywhere で認証すると、IAM ロールの一時的な認証情報を取得して AWS リソースにアクセスできるようになります。

IAM Roles Anywhere の設定の詳細については、「IAM Roles Anywhere User Guide」(IAM Roles Anywhere ユーザーガイド) の「[What is AWS Identity and Access Management Roles Anywhere](#)」(Roles Anywhere とは) を参照してください。



## 第三者が所有する AWS アカウント へのアクセス権を付与する

組織内の AWS リソースへ組織外の第三者がアクセスする必要がある場合には、ロールを使用することでアクセス許可を委任することができます。たとえば、組織内の AWS リソースの管理を第三者へ委託しているような場合が相当します。IAM ロールを使用することで、AWS セキュリティ認証情報を共有することなく第三者に AWS リソースへのアクセスを許可することができます。第三者は代わりに、AWS アカウント に作成したロールを引き受けることで、AWS リソースにアクセスできます。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

第三者は、以下の情報を提供する必要があります。これらの情報は、第三者が引き受けることのできるロールの作成に必要です。

- 第三者の AWS アカウント ID。ロールの信頼ポリシーを定義するときは、その AWS アカウント ID をプリンシパルとして指定します。
- ロールを一意に関連付けるための外部 ID。外部 ID は、ユーザーと第三者によってのみ識別される識別子です。たとえば、ユーザーとサードパーティーの間の請求書 ID は使用できますが、サードパーティーの名前や電話番号などの推測できるものは使用しないでください。ロールの信頼ポリシーを定義するときは、この ID を指定する必要があります。サードパーティーは、ロールを引き受けるときに、この ID を指定する必要があります。外部 ID の詳細については、[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#) を参照してください。
- 第三者が AWS リソースでの作業を行うのに必要なアクセス許可。ロールのアクセス許可ポリシーを定義する際に、権限を指定する必要があります。このポリシーには、第三者はどのアクションができるのか、およびどのリソースにアクセスできるのかが定義されています。

ロールの作成が完了したら、そのロールの Amazon リソースネーム (ARN) を対象の第三者に提供します。第三者がロールを担当するにあたり、このロールの ARN を必要とします。

### Important

お客様の AWS リソースへのアクセスを第三者に許可すると、第三者はポリシーで指定したすべてのリソースにアクセスできます。サードパーティーによるリソースの使用は、ユーザーに請求されます。したがって、第三者によるリソースの使用については適切な制限を設けるようにしてください。

## AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法

お客様の AWS リソースへのアクセス権を第三者に付与 (委任) することが必要になる場合があります。このシナリオで重要なのは、オプションの情報としての外部 ID です。この ID を IAM ロールの信頼ポリシーで使用することで、ロールを引き受けることができるユーザーを指定できます。

### Important

AWS は、外部 ID を機密情報として扱いません。アクセスキーペアやパスワードなどの機密情報を AWS で作成した場合、それらを再び表示することはできません。ロールの外部 ID は、ロールを表示する権限を持つすべてのユーザーが表示できます。

異なる AWS で複数の顧客をサポートするマルチテナント環境では、AWS アカウント アカウントごとに 1 つの外部 ID を使用することをお勧めします。この ID は、サードパーティーによって生成されたランダムな文字列である必要があります。

ロールを引き受けるときにサードパーティーが外部 ID を提供することを要求するには、ロールの信頼ポリシーを選択した外部 ID で更新します。

ロールを引き受けるときに外部 ID を提供するには、AWS CLI または AWS API を使用してそのロールを引き受けます。詳細については、「STS [AssumeRole](#) API オペレーション」または「STS [assume-role](#) CLI オペレーション」を参照してください。

例えば、Example Corp という第三者企業を雇って、コストを最適化するためにお客様の AWS アカウント をモニタリングする業務を依頼するとします。Example Corp がお客様の毎日の支出を追跡するには、お客様の AWS リソースにアクセスする必要があります。また、Example Corp は他の顧客について他の多くの AWS アカウントをモニタリングしています。

Example Corp に AWS アカウントの IAM ユーザーとのその長期的認証情報へのアクセスを許可しないでください。代わりに、IAM ロールとその一時的なセキュリティ認証情報を使用します。IAM ロールは、第三者が長期的認証情報 (IAM ユーザーアクセスキーなど) を共有することなくお客様の AWS リソースにアクセスできるようにするメカニズムです。

IAM ロールを使用して AWS アカウント と Example Corp アカウント と間の信頼関係を確立できます。この関係が確立されると、Example Corp アカウントのメンバーは、AWS Security Token Service [AssumeRole](#) API を呼び出して、一時的なセキュリティ認証情報を取得できます。次に、Example Corp のメンバーは、この認証情報を使用してアカウントの AWS リソースにアクセスできます。



**Note**

一時的セキュリティ認証情報を取得するために呼び出すことのできる AssumeRole とその他の AWS API オペレーションの詳細については、「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

このシナリオの詳細は以下のとおりです。

1. お客様は Example Corp を雇うとします。Example Corp はお客様の一意の顧客 ID を作成します。Example Corp はその一意の顧客 ID と Example Corp の AWS アカウント 番号をお客様に提供します。この情報は次の手順でお客様が IAM ロールを作成するために必要になります。

**Note**

Example Corp は、顧客ごとに一意である限り、任意の文字列値を ExternalId に使用できます。顧客のアカウント番号を使用することもできますし、ランダムな文字列でもかまいません。ただし、2つの顧客に同じ値を使用することはできません。「シークレット」することを意図したものではありません。Example Corp は、顧客ごとに ExternalId 値を指定する必要があります。重要なのは、各外部IDがユニークであることを確認するために、お客様によってではなく、Example Corp によって生成される必要があるということです。

2. お客様は AWS にサインインし、お客様のリソースへのアクセス権を Example Corp に付与する IAM ロールを作成します。他の IAM ロールと同様に、このロールにも2つのポリシー（アクセス許可ポリシーと信頼ポリシー）があります。ロールの信頼ポリシーでは、だれがこのロールを引き受けることができるかを指定します。このサンプルシナリオでは、ポリシーで Principal として Example Corp の AWS アカウント 番号を指定します。これにより、そのアカウントの ID はロールを引き受けられるようになります。さらに、信頼ポリシーに [Condition](#) 要素を追加します。この Condition は ExternalId コンテキストキーをテストして、その要素が Example Corp の一意の顧客 ID に一致するようにします。その例を示します。

```
"Principal": {"AWS": "Example Corp's AWS ##### ID"},  
"Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example Corp"}}
```

3. ロールのアクセス許可ポリシーでは、ロールがだれに何を許可するかを指定します。たとえば、お客様の Amazon EC2 と Amazon RDS のリソースのみを管理でき、お客様の IAM ユーザーまた

はグループは管理できないようにロールを指定できます。サンプルシナリオでは、アクセス許可ポリシーを使用して、お客様のアカウントのすべてのリソースに対する読み取り専用アクセス権を Example Corp に付与します。

4. ロールの作成が完了したら、そのロールの Amazon リソースネーム (ARN) を Example Corp に提供します。
5. Example Corp がお客様の AWS リソースに対するアクセス許可を必要とするとき、Example Corp の担当者が AWS sts:AssumeRole API を呼び出します。この呼び出しには、引き受けるロールの ARN とお客様の顧客 ID に対応する ExternalId パラメータが含まれています。

リクエスト元が Example Corp の AWS アカウント であり、ロール ARN と外部 ID が正しい場合、リクエストは成功します。次に、一時的なセキュリティ認証情報が提供されます。Example Corp はその情報を使用して、お客様のロールが許可している AWS リソースにアクセスできます。

つまり、ロールのポリシーに外部 ID が含まれている場合、そのロールを引き受けるユーザーは、ロールでプリンシパルであり、正しい外部 ID を指定する必要があります。

## 外部 ID を使用する理由

抽象的には、外部 ID により、ロールを引き受けるユーザーは、業務を遂行する環境へのアクセスを要求できます。また、アカウント所有者が、特定の環境においてのみロールを引き受けることができるようにする方法の 1 つでもあります。外部 ID の最も重要な機能は、[混乱する代理問題](#) の防止と対処です。

## 外部 ID が適している状況

外部 ID は以下の状況で使用します。

- お客様は AWS アカウント の所有者で、それ以外の AWS アカウント アカウントにアクセスするロールを第三者のために設定しました。第三者がお客様のロールを引き受けるときに含める外部 ID は、第三者に問い合わせる必要があります。その後、ロールの信頼ポリシーでその外部 ID を確認します。これにより、外部の第三者は、お客様に代わって操作を行う場合にのみ、お客様のロールを引き受けることができます。
- お客様は、前のシナリオの Example Corp のように、さまざまな顧客に代わってロールを引き受ける立場にあります。各顧客に一意の外部 ID を割り当て、ロールの信頼ポリシーに外部 ID を追加するように指示します。ロールを引き受けるには、リクエストに正しい外部 ID が常に含まれていることを確認する必要があります。

既にそれぞれの顧客に一意的 ID を割り当てている場合は、この一意の ID を外部 ID として使用できます。外部 ID は、この目的のためだけに明示的に作成したり個別に追跡したりする必要のある特別な値ではありません。

外部 ID は常に AssumeRole API 呼び出しで指定する必要があります。さらに、顧客からロールの ARN を受け取ったら、正しい外部 ID と正しくない外部 ID の両方を使用して、そのロールを引き受けることができるかどうかをテストします。正しい外部 ID がなくてもロールを引き受けることのできる場合は、システムに顧客のロール ARN を保存しないでください。顧客がロール信頼ポリシーを更新し、正しい外部 ID を要求するまで待ちます。こうすることにより、顧客による不正を防止し、「混乱した代理」問題からお客様と顧客の両方を守ることができます。

## AWS サービスへのアクセスの提供

多くの AWS のサービスでは、ロールを使用して、そのサービスがアクセスできものを制御する必要があります。サービスがお客様に代わってアクションを実行するために引き受けるロールは、[サービスロール](#)と呼ばれます。ロールにサービスに対して特殊な目的がある場合、そのロールは [EC2 インスタンスのサービスロール](#)、または [サービスにリンクされたロール](#)として分類できます。特定のサービスがロールを使用するかどうかと、使用するサービスのロールを割り当てる方法については、各サービスの [AWS ドキュメント](#)を参照してください。

AWS が提供するサービスへのアクセスを委任するロールの作成については、「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。

## 混乱する代理問題

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。これを防ぐために、サードパーティー (クロスアカウント) や他の AWS サービス (クロスサービス) に対して、お客様のアカウント内のリソースへのアクセス権を提供してしまった場合に、お客様のアカウントの保護に役立つツールを AWS が提供します。

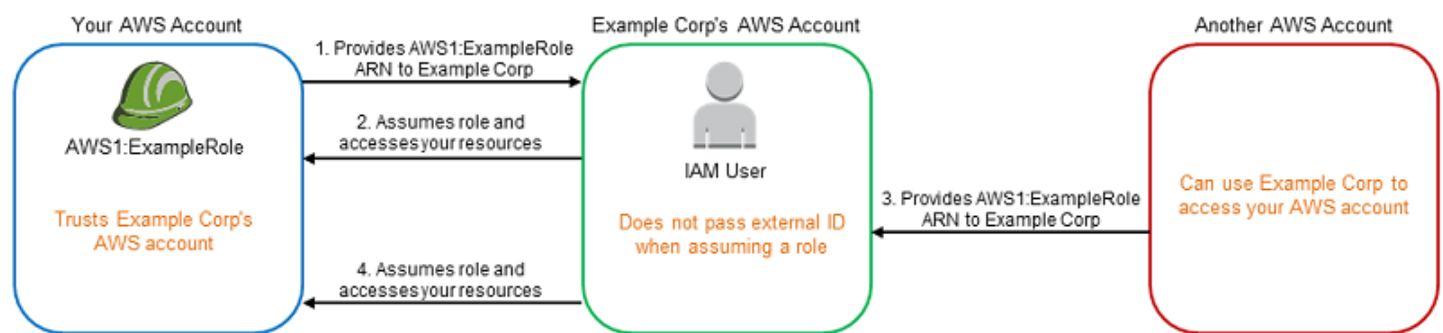
お客様の AWS リソースへのアクセス権をサードパーティーに付与 (委任) することが必要になる場合があります。例えば、Example Corp という第三者企業を雇って、コストを最適化するためにお客様の AWS アカウント をモニタリングする業務を依頼するとします。Example Corp がお客様の毎日の支出を追跡するには、お客様の AWS リソースにアクセスする必要があります。また、Example Corp は他の顧客について他の多くの AWS アカウント をモニタリングしています。IAM ロールを使用して AWS アカウント と Example Corp アカウント と間の信頼関係を確立できます。このシナリオで重要なのは、オプションの情報としての外部 ID です。この ID を IAM ロールの信頼ポリシーで使

用することで、ロールを引き受けることができるユーザーを指定できます。外部 ID の最も重要な機能は、混乱した代理問題の防止と対処です。

AWS では、サービス間でのなりすましが、混乱した代理問題を生じさせる可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。

### クロスアカウントでの混乱した代理

次の図は、クロスアカウントでの「混乱した代理」問題を示しています。



このシナリオでは、次のことを前提としています。

- AWS1 はあなたの AWS アカウント。
- AWS1:ExampleRole は、お客様のアカウントのロールである。このロールの信頼ポリシーは、Example Corp の AWS アカウントを、ロールを引き受けられるアカウントとして指定することによって、Example Corp を信頼しています。

次の状況が発生します。

1. お客様は、Example Corp のサービスの使用を開始するとき、AWS1:ExampleRole の ARN を Example Corp に提供します。
2. Example Corp はそのロールの ARN を使用して、お客様の AWS アカウント のリソースにアクセスするために必要な一時的なセキュリティ認証情報を入手します。この方法では、お客様に代わって操作を実行できる "代理" として Example Corp を信頼します。
3. 別の AWS ユーザーも Example Corp のサービスを利用し始め、このユーザーも AWS1:ExampleRole の ARN を Example Corp が使用できるように提供するとします。この別のユーザーは、秘密ではない AWS1:ExampleRole を知った、または推測した可能性があります。

4. その別のユーザーが Example Corp に自分のアカウントの AWS リソース (そう自称しているリソース) へのアクセスを依頼すると、Example Corp は `AWS1:ExampleRole` を使用してお客様のアカウントのリソースにアクセスします。

このようにして、その別のユーザーはお客様のリソースに不正にアクセスできます。Example Corp は、この別のユーザーによって操られ、無意識にお客様のリソースにアクセスしたため、「混乱した代理」になります。

Example Corp は、ロールの信頼ポリシーに `ExternalId` の条件の確認を含めることを必須とすることで、「混乱した代理」問題に対応できます。Example Corp は、顧客ごとに一意の `ExternalId` 値を生成して、ロールを引き受けるリクエストでその値を使用します。`ExternalId` 値は、Example Corp の顧客の間で一意でなければならず、Example Corp の顧客ではなく、Example Corp によって管理されます。そのため、外部 ID は Example Corp から取得するものであり、自分では用意できません。これにより、Example Corp が混乱した代理人になることを防ぎ、別のアカウントの AWS リソースへのアクセスを許可してしまうことを防ぎます。

このシナリオでは、Example Corp がお客様に割り当てた一意の ID は 12345 であり、もう一方のお客様に割り当てた ID は 67890 であるとしめます。これらの ID は、このシナリオで使用するのみを目的としているため、簡略化されています。通常、これらの ID は GUID です。これらの ID が Example Corp の顧客の間で一意であることを考慮すると、これらの値を外部 ID として使用することは賢明です。

Example Corp はお客様に 12345 という外部 ID 値を提供します。お客様は、Condition 値が 12345 であることを必須とする以下のような `sts:ExternalId` 要素をロールの信頼ポリシーに追加する必要があります。

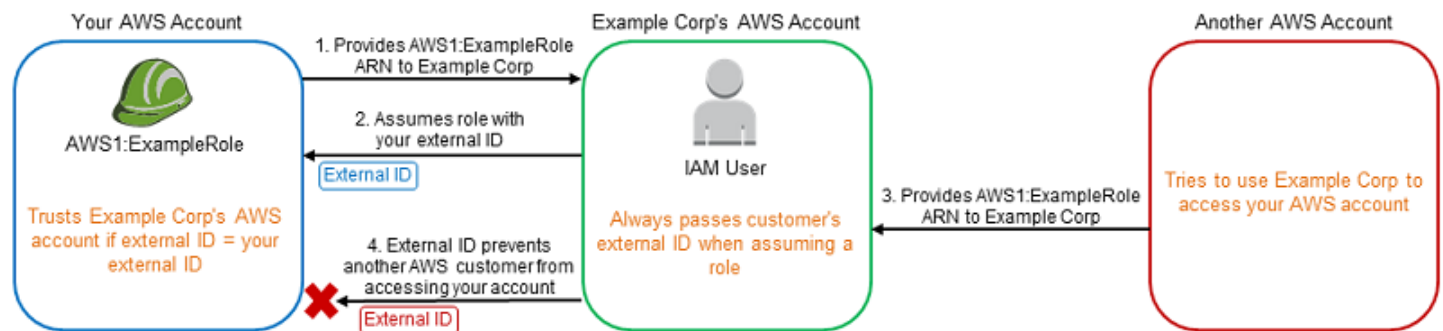
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "Example Corp's AWS Account ID"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "12345"
      }
    }
  }
}
```



}

このポリシーの Condition 要素により、AssumeRole API 呼び出しに 12345 という外部 ID 値が含まれている場合にのみ Example Corp はロールを引き受けることができます。Example Corp は、顧客に代わってロールを引き受けるたびに、その顧客の外部 ID 値を AssumeRole 呼び出しに含めます。別の顧客がお客様の ARN を Example Corp に提供した場合でも、Example Corp が AWS へのリクエストに含める外部 ID を顧客がコントロールすることはできません。これにより、権限のない顧客がお客様のリソースにアクセスすることを防止できます。

次の図は、このプロセスを示したものです。



1. 前述と同様に、お客様は、Example Corp のサービスを利用し始めるとき、AWS1:ExampleRole の ARN を Example Corp に提供します。
2. Example Corp がそのロールの ARN を使用して AWS1:ExampleRole ロールを引き受けるとき、Example Corp は AssumeRole API コールにお客様の外部 ID (12345) を含めます。この外部 ID はロールの信頼ポリシーと一致するため、AssumeRole API 呼び出しは正常に実行され、Example Corp はお客様の AWS アカウントのリソースにアクセスするための一時的なセキュリティ認証情報を取得します。
3. 別の AWS ユーザーも Example Corp のサービスを利用し始め、先ほどと同様、このユーザーも AWS1:ExampleRole の ARN を Example Corp が使用できるように提供するとします。
4. しかし、今回は、Example Corp が AWS1:ExampleRole ロールを引き受けるとき、もう一方のお客様に関連付けられている外部 ID (67890) が提供されます。その別の顧客がこの ID を変更することはできません。Example Corp がこれを行うのは、ロールを使用するリクエストがもう一方のお客様から来たからであり、67890 は Example Corp が業務を遂行する環境を示すからです。お客様は自身の外部 ID (12345) を使用する条件を AWS1:ExampleRole の信頼ポリシーに追加したため、AssumeRole API コールは失敗します。別の顧客がお客様のアカウントのリソースに不正にアクセスすることが防止されます (図の赤色の「X」で示しています)。

外部 ID により、Example Corp が他の顧客によって操られ、無意識にお客様のリソースにアクセスすることを防止します。

## サービス間の混乱した代理の防止

リソースベースのポリシー内では

[aws:SourceArn](#)、[aws:SourceAccount](#)、[aws:SourceOrgID](#)、または [aws:SourceOrgPaths](#) のグローバル条件コンテキストキーを使用して、サービスが持つ特定のリソースへのアクセス許可を制限することをお勧めします。1 つのリソースだけをクロスサービスのアクセスに関連付ける場合は、[aws:SourceArn](#) を使用します。アカウント内の任意のリソースをクロスサービスの使用に関連付ける場合は、[aws:SourceAccount](#) を使用します。組織内の任意のアカウントの任意のリソースをクロスサービスの使用に関連付ける場合は、[aws:SourceOrgID](#) を使用します。AWS Organizations パス内の任意のアカウントのリソースをクロスサービスの使用に関連付ける場合は、[aws:SourceOrgPaths](#) を使用します。パスの使用と理解の詳細については、「[AWS Organizations エンティティパスを理解する](#)」を参照してください。

混乱した代理問題から保護するための最もきめ細かな方法は、リソースベースポリシー内のリソースの完全な ARN を指定しながら、グローバル条件コンテキストキー [aws:SourceArn](#) を使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバル条件コンテキストキー [aws:SourceArn](#) で、ARN の未知部分を示すためにワイルドカード (\*) を使用します。例えば、`arn:aws:servicename:*:123456789012:*` と指定します。

[aws:SourceArn](#) の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方の [aws:SourceAccount](#) と [aws:SourceArn](#) を使用して、アクセス許可を制限する必要があります。

混乱した代理問題から保護するために、リソースベースポリシー内のリソースの組織 ID または組織パスを指定しながら、[aws:SourceOrgID](#) または [aws:SourceOrgPaths](#) のグローバル条件コンテキストキーを使用してください。[aws:SourceOrgID](#) または [aws:SourceOrgPaths](#) キーを含むポリシーには正しいアカウントが自動的に組み込まれるため、組織のアカウントを追加、削除、移動する際には手動で更新する必要はありません。

サービスにリンクされていないロールの[信頼ポリシー](#)の場合、信頼ポリシー内のすべてのサービスで `iam:PassRole` アクションを実行して、ロールが読み出し元サービスと同じアカウントにあることが検証されています。その結果、これらの信頼ポリシーと [aws:SourceAccount](#)、[aws:SourceOrgID](#)、または [aws:SourceOrgPaths](#) を併用する必要はありません。信頼ポリシーで [aws:SourceArn](#) を使用すると、Lambda 関数 ARN など、代理としてロールが引き受けることのできるリソースを指定できます。一部の AWS のサービスでは、新しく



作成されたロールに対して信頼ポリシーで `aws:SourceAccount` と `aws:SourceArn` を使用しますが、アカウント内の既存のロールにキーを使用する必要はありません。

#### Note

KMS キーグラントを使用して AWS Key Management Service と統合される AWS のサービスは、`aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID` または `aws:SourceOrgPaths` 条件キーをサポートしていません。KMS キーポリシーでこれらの条件キーを使用すると、そのキーが KMS キー付与経由で AWS のサービスでも使用されると、予期しない動作が発生します。

## AWS Security Token Service のクロスサービスでの混乱した代理の防止

AWS の多くのサービスでは、ロールを使用して、ユーザーに代わって該当サービスが他のサービスのリソースにアクセスすることを許可する必要があります。サービスがお客様に代わってアクションを実行するために引き受けるロールは、[サービスロール](#)と呼ばれます。ロールには 2 つのポリシーが必要です。ロールを引き受けることができるプリンシパルを指定する、ロールの信頼ポリシーと、ロールで実行できる操作を指定する、アクセス許可ポリシーです。IAM においては、ロールの信頼ポリシーがリソースベースのポリシーの唯一のタイプです。その他の AWS のサービスには、Amazon S3 バケットポリシーなどのリソースベースのポリシーがあります。

サービスがユーザーに代わってロールを引き受ける場合、サービスプリンシパルが [sts:AssumeRole](#) アクションを実行できるように、ロールの信頼ポリシーで許可されている必要があります。サービスが `sts:AssumeRole` を呼び出すとき、ロールのアクセス許可ポリシーで許可されているリソースに、サービスプリンシパルがアクセスするために使用する、一時的なセキュリティ認証情報のセットを、AWS STS が返します。アカウント内でサービスがロールを引き受ける場合は、ロール信頼ポリシーに `aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID` または `aws:SourceOrgPaths` のグローバル条件コンテキストキーを含めることで、期待するリソースによって生成されたリクエストのみに、ロールのアクセスを制限できます。

例えば、AWS Systems Manager Incident Manager では、お客様の代わりに Incident Manager が Systems Manager Automation ドキュメントを実行できるように、ロールを選択する必要があります。オートメーションドキュメントには、CloudWatch アラームまたは EventBridge イベントによって開始される、インシデントの自動化された応答プランを含めることができます。次のロール信頼ポリシーの例では、`aws:SourceArn` 条件キーを使用して、インシデントレコードの ARN をもとに、サービスロールへのアクセスを制限できます。応答プランリソース `myresponseplan` から作成されたインシデントレコードのみが、このロールを使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "ssm-incidents.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:ssm-incidents:*:111122223333:incident-
record/myresponseplan/*"
      }
    }
  }
}
```

#### Note

AWS STS と統合されるすべてのサービス

が、aws:SourceArn、aws:SourceAccount、aws:SourceOrgID、または aws:SourceOrgPaths 条件キーをサポートしているわけではありません。サポートされていない統合と共に IAM 信頼ポリシーでこれらのキーを使用すると、予期しない動作が発生する可能性があります。

## 外部で認証されたユーザー ( ID フェデレーション ) へのアクセスの許可

社内のディレクトリなど、AWS 以外の ID をユーザーがすでに持っているとします。これらのユーザーが AWS リソースを使用する (または、それらのリソースにアクセスするアプリケーションを使用する) 必要がある場合、これらのユーザーには AWS セキュリティ認証情報も必要です。IAM ロールを使用して、ID が組織または第三者のプロバイダー (IdP) からフェデレーションされたユーザーのアクセス許可を指定できます。

#### Note

セキュリティ上のベストプラクティスとして、IAM ユーザーを作成する代わりに、ID フェデレーションを使用して [IAM Identity Center](#) でユーザーアクセスを管理することをお勧めしま

す。IAM ユーザーが必要な特定の状況についての情報は、「[IAMユーザー \(ロールの代わりに\) を作成する場合](#)」を参照してください。

## Amazon Cognito を使用したモバイルまたはウェブベースのアプリのユーザーのフェデレーション

AWS リソースにアクセスするモバイルまたはウェブベースのアプリを作成する場合、アプリが AWS にプログラムによるリクエストを送るには認証情報が必要になります。ほとんどのモバイルアプリケーションのシナリオでは、[Amazon Cognito](#) の使用をお勧めします。このサービスを [AWS Mobile SDK for iOS](#) および [AWS Mobile SDK for Android and Fire OS](#) で使用して、ユーザーの一意の ID を作成し、AWS リソースへの安全なアクセスのためにユーザーを認証できます。Amazon Cognito では、次のセクションに示されているのと同じ ID プロバイダーがサポートされます。さらに、[開発者が認証した ID](#) と認証されていない (ゲスト) アクセスもサポートされます。また、Amazon Cognito には、ユーザーがデバイスを変えてもデータが保持されるように、ユーザーデータを同期するための API オペレーションも用意されています。詳細については、「[モバイルアプリのための Amazon Cognito の使用](#)」を参照してください。

## パブリック ID サービスプロバイダーまたは OpenID Connect を使用したユーザーのフェデレーション

可能な限り、モバイルおよびウェブベースのアプリケーションシナリオで Amazon Cognito を使用してください。Amazon Cognito は、パブリック ID プロバイダーサービスを使用する際の裏方作業をほとんど行います。同じサードパーティのサービスで機能し、また匿名サインインもサポートしています。ただし、より高度なシナリオでは、OpenID Connect (OIDC) と互換性がある Login with Amazon、Facebook、Google、その他 IdP でのログインなど、サードパーティのサービスを直接使用できます。これらのサービスを使用した OIDC フェデレーションの使用についての詳細は、「[OIDC フェデレーション](#)」を参照してください。

## SAML 2.0 を使用したユーザーのフェデレーション

組織が既に SAML 2.0 (Security Assertion Markup Language 2.0) をサポートする ID プロバイダーソフトウェアパッケージを使用している場合、ID プロバイダー (IdP) としての組織と、サービスプロバイダーとしての AWS との間に信頼を作成できます。その後、SAML を使用して、AWS Management Console へのフェデレーティッドシングルサインオン (SSO) または AWS API オペレーションを呼び出すためのフェデレーションアクセスをユーザーに許可できます。たとえば、社内で Microsoft Active Directory と Active Directory Federation Services を使用している場合は、SAML 2.0 を使用してフェデレーションが可能です。SAML 2.0 を使用したユーザーのフェデレーション方法の詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。

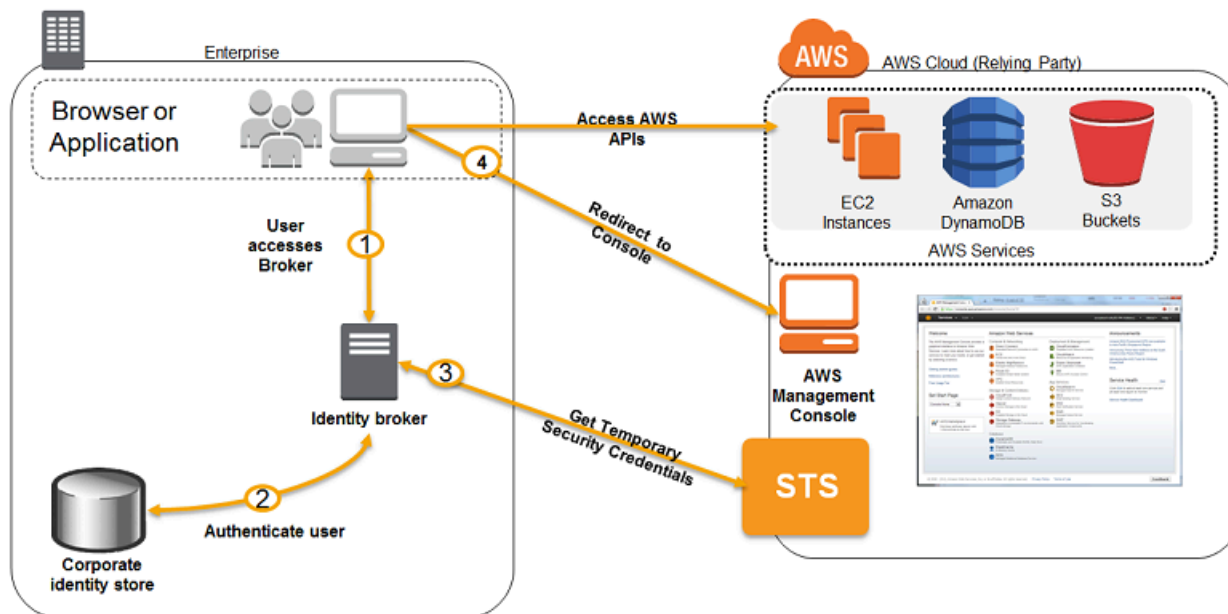
## カスタム ID ブローカーアプリケーションを作成するユーザーのフェデレーション

ID ストアに SAML 2.0 との互換性がない場合、カスタム ID ブローカーアプリケーションを作成して同じような機能を実行できます。ブローカーアプリケーションはユーザーを認証し、そのユーザー用に一時的な認証情報を AWS に要求して、それをユーザーに提供し AWS リソースにアクセスできるようにします。

たとえば、Example Corp. には、会社の AWS リソースにアクセスする社内アプリケーションを実行する必要がある従業員が多数います。従業員は、会社の ID および認証システムに既に ID があり、従業員ごとに別の IAM ユーザーを作成することは望ましくありません。

Bob は Example Corp の開発者です。Example Corp の社内アプリケーションで会社の AWS リソースにアクセスできるようにするために、カスタム ID ブローカーアプリケーションを開発しています。このアプリケーションは、従業員が Example Corp. の既存の ID および認証システムにサインインしていることを確認します。これには、LDAP や Active Directory などのシステムを使用している可能性があります。ID ブローカーアプリケーションは、従業員の一時的なセキュリティ認証情報を取得します。このシナリオは、前のシナリオ ( カスタム認証システムを使用するモバイルアプリ ) に似ています。ただし、AWS リソースにアクセスする必要があるアプリケーションはすべて企業ネットワーク内で実行され、会社には既存の認証システムがあります。

一時的なセキュリティ認証情報を取得するために、ID ブローカーアプリケーションは、ユーザーのポリシーの管理方法と一時的な認証情報の有効期限に応じて、AssumeRole または GetFederationToken を呼び出して、一時的なセキュリティ認証情報を取得します (これらの API オペレーションの違いの詳細については、「[IAM の一時的な認証情報](#)」および「[一時的なセキュリティ認証情報のアクセス権限を制御する](#)」を参照してください)。この呼び出しは、AWS アクセスキー ID、シークレットアクセスキー、およびセッショントークンで構成される一時的なセキュリティ認証情報を返します。ID ブローカーアプリケーションは、これらの一時的なセキュリティ認証情報を社内アプリケーションで使用できるようにします。アプリは、一時的な認証情報を使用して AWS を直接呼び出すことができます。アプリは、認証情報をキャッシュし、有効期限が切れると、新しい一時的な認証情報をリクエストします。このシナリオを以下に図で示します。



このシナリオには次の属性があります。

- IDブローカーアプリケーションには、一時的なセキュリティ認証情報を作成するために IAM のトークンサービス (STS) API にアクセスする権限があります。
- IDブローカーアプリケーションが、従業員が既存の認証システム内で認証されていることを確認できます。
- ユーザーは、AWS マネジメントコンソールへのアクセスを与える一時的な URL を入手できます (これはシングルサインオンと呼ばれています)。

一時的なセキュリティ認証情報の作成方法の詳細については、「[一時的なセキュリティ認証情報のリンクエントリ](#)」を参照してください。AWS マネジメントコンソールへのアクセスを取得するフェデレーションユーザーの詳細については、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする](#)」を参照してください。

## サービスリンクロールの使用

サービスにリンクされたロールは、AWS のサービスに直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、サービスによって事前定義されており、お客様の代わりにサービスから他の AWS サービスを呼び出す必要のあるアクセス権限がすべて含まれています。このリンクされたサービスでも、サービスにリンクされたロールを作成、変更、削除する方法を定義しています。サービスによって、ロールが自動的に作成または削除される場合があります。そのため、ウィザードの一部、またはサービスのプロセスとして、ロールを作成、変更、削除できる場合があ

ります。または、ロールを作成または削除するには、IAM を使用する必要がある場合があります。どの方法を使用するにしても、サービスにリンクされたロールを使用すると、サービスが自動でアクションを完了させるのに、アクセス許可を手動で追加する必要がないため、サービスの設定プロセスが簡単になります。

#### Note

サービスロールは、サービスにリンクされたロールとは異なることに注意してください。サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#) です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。サービスリンクロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは、AWS アカウントに表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集することはできません。

リンクされたサービスで、そのサービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、ロールは、そのサービスでのみ引き受けることができます。定義した許可には、信頼ポリシーと許可ポリシーが含まれます。この許可ポリシーを他のIAM エンティティにアタッチすることはできません。

ロールを削除する前に、最初に関連するリソースを削除する必要があります。これにより、リソースへの意図しないアクセスによる許可の削除が防止され、リソースは保護されます。

#### Tip

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

## サービスにリンクされたロールのアクセス許可

ユーザーまたはロールがサービスにリンクされたロールを作成または編集できるようにするには、IAM エンティティ (ユーザーまたはロール) のアクセス許可を設定する必要があります。



**Note**

サービスにリンクされたロールの ARN にはサービスプリンシパルが含まれています。以下のポリシーでは `SERVICE-NAME.amazonaws.com` として示されています。サービスプリンシパルは推量しないでください。サービスプリンシパルは、大文字と小文字が区別され、AWS のサービス間で異なる場合があります。サービスのサービスプリンシパルを表示するには、そのサービスにリンクされたロールのドキュメントを参照してください。

特定のサービスにリンクされたロールの作成を IAM エンティティに許可するには

サービスにリンクされたロールを作成する必要がある IAM エンティティに、次のポリシーを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX",
      "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-NAME.amazonaws.com"}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:PutRolePolicy"
      ],
      "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX"
    }
  ]
}
```

IAM エンティティがサービスにリンクされた任意のロールを作成することを許可するには

サービスにリンクされたロール、または必要なポリシーを含む任意のサービスロールを作成する必要がある IAM エンティティのアクセス許可ポリシーに、次のステートメントを追加します。このポリ



シーステートメントでは、IAM エンティティがポリシーをロールにアタッチすることは許可されません。

```
{
  "Effect": "Allow",
  "Action": "iam:CreateServiceLinkedRole",
  "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

IAM エンティティが任意のサービスロールの説明を編集することを許可するには

サービスにリンクされたロール、または任意のサービスロールの説明を編集する必要がある IAM エンティティのアクセス許可ポリシーに、次のステートメントを追加します。

```
{
  "Effect": "Allow",
  "Action": "iam:UpdateRoleDescription",
  "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

IAM エンティティがサービスにリンクされた特定のロールを削除することを許可するには

サービスにリンクされたロールを削除する必要がある IAM エンティティのアクセス許可ポリシーに、次のステートメントを追加します。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
}
```

IAM エンティティがサービスにリンクされた任意のロールを削除することを許可するには

サービスロールではなく、サービスにリンクされたロールを削除する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

```
{
```

```
"Effect": "Allow",
"Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
],
"Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

既存のロールをサービスに渡すことを IAM エンティティに許可するには

一部の AWS サービスでは、新しいサービスにリンクされたロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可がユーザーに必要です。ロールを渡す必要のある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。このポリシーステートメントでは、エンティティは、渡すことができるロールのリストを表示できます。詳細については、「[AWS のサービスにロールを渡すアクセス権限をユーザーに付与する](#)」を参照してください。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
},
{
  "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
  "Effect": "Allow",
  "Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account-id:role/my-role-for-XYZ"
}
```

## サービスにリンクされたロールによる間接的なアクセス許可

サービスにリンクされたロールによって付与されたアクセス許可は、他のユーザーおよびロールに間接的に転送できます。サービスにリンクされたロールが AWS サービスによって使用される場合、そのサービスにリンクされたロールは、自身のアクセス許可を使用して他の AWS サービスを呼び出すことができます。つまり、サービスにリンクされたロールを使用するサービスを呼び出すアクセス許可を持つユーザーとロールは、そのサービスにリンクされたロールがアクセスできるサービスに間接的にアクセスできる可能性があります。

たとえば、Amazon RDS DB インスタンスを作成すると、[RDS のサービスにリンクされたロール](#)は、まだ存在しない場合は自動的に作成されます。このサービスにリンクされたロールを使用

すると、RDS は、ユーザーに代わって Amazon EC2、Amazon SNS、Amazon SNS、Amazon CloudWatch Logs、Amazon Kinesis を呼び出すことを許可します。アカウントのユーザーとロールに RDS データベースの変更や作成を許可すると、RDS を呼び出すことで Amazon EC2、Amazon SNS、Amazon CloudWatch Logs ログ、Amazon Kinesis のリソースと間接的にやり取りできるようになる可能性があります。RDS はサービスにリンクされたロールを使用してこれらのリソースにアクセスするためです。

## サービスにリンクされたロールの作成

サービスにリンクされたロールを作成するメソッドは、サービスによって異なります。場合によっては、サービスにリンクされたロールを手動で作成する必要はありません。たとえば、サービス特定のアクション (リソースの作成) を完了すると、サービスによって、サービスにリンクされたロールが作成される場合があります。または、サービスにリンクされたロールのサポートを開始する前からサービスを使用していた場合は、アカウントにロールが自動的に作成される場合があります。詳細については、「[AWS アカウントに新しいロールが表示される](#)」を参照してください。

また、サービスにリンクされたロールは、サービスコンソール、API、CLI を使用して、手動で作成できる場合があります。サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールをサービスで作成できるかどうかを確認するには、「はい」リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを参照してください。

ロールの作成がサービスでサポートされていない場合は、IAM を使用して、サービスにリンクされたロールを作成できます。

### Important

サービスにリンクされたロールでは、[AWS アカウントの IAM ロール](#)の制限に向かってカウントされますが、このロールは制限を超えてもアカウントに作成することができます。この制限を超える可能性があるのは、サービスにリンクされたロールのみです。

## サービスにリンクされたロールの作成 (コンソール)

IAM のサービスにリンクされたロールを作成する前に、サービスにリンクされたロールがサービスで自動的に作成されるかどうかを確認します。さらに、サービスのコンソール、API、または CLI からロールを作成できるかどうかを確認します。

## サービスにリンクされたロールを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。続いて、[Create role] (ロールの作成) を選択します。
3. [AWS のサービス] ロールタイプを選択します。
4. サービスのユースケースを選択します。ユースケースは、サービスに必要な信頼ポリシーを含めるように定義されています。次に、[次へ] を選択します。
5. ロールにアタッチするアクセス権限ポリシーを 1 つ以上選択します。選択したユースケースに基づき、サービスで以下のいずれかを行う場合があります。
  - ロールで使用するアクセス権限を定義します。
  - 制限されたアクセス権限からの選択を許可します。
  - すべてのアクセス権限からの選択を許可します。
  - この時点でポリシーを選択できないようにし、ポリシーを作成してからロールにアタッチします。

ロールに許可する許可を割り当てるポリシーの横にあるチェックボックスを選択し、[Next] (次へ) を選択します。

### Note

設定したアクセス権限は、ロールを使用するすべてのエンティティで有効となります。デフォルトでは、ロールにはいずれのアクセス権限もありません。

6. [ロール名] で、ロール名のカスタマイズの度合いはサービスによって定義されます。サービスのロール名が定義されている場合、このオプションを変更することはできません。その他の場合、サービスはロールのプレフィックスを定義し、オプションのサフィックスを入力できるようにするかもしれません。

可能であれば、ロールのデフォルト名に追加するサフィックスを入力します。このサフィックスは、このロールの目的を識別するのに役立ちます。ロール名は AWS アカウント内で一意でなければなりません。大文字と小文字は区別されません。例えば、`<service-linked-role-name>_SAMPLE` と `<service-linked-role-name>_sample` というロール名を両方作成する

ことはできません。多くのエンティティによりロールが参照されるため、作成後にロール名を変更することはできません。

7. (オプション) [Description] (説明) で、サービスにリンクされた新しいロールの説明を編集します。
8. 作成中にサービスにリンクされたロールにタグ付けすることはできません。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
9. ロール情報を確認し、[Create role (ロールの作成)] を選択します。

### サービスにリンクされたロールの作成 (AWS CLI)

IAM でサービスにリンクされたロールを作成するには、リンクされたサービスで、サービスにリンクされたロールが自動的に作成されるかどうかと、サービスの CLI からロールを作成できるかどうかについて確認します。サービス CLI がサポートされていない場合は、IAM コマンドを使用して、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めて、サービスにリンクされたロールを作成することができます。

### サービスにリンクされたロールを作成するには (AWS CLI)

次のコマンドを実行します。

```
aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

### サービスにリンクされたロールの作成 (AWS API)

IAM でサービスにリンクされたロールを作成するには、リンクされたサービスで、サービスにリンクされたロールが自動的に作成されるかどうかと、サービスの API からロールを作成できるかどうかについて確認します。サービス API がサポートされていない場合は、AWS API を使用して、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めて、サービスにリンクされたロールを作成することができます。

### サービスにリンクされたロールを作成するには (AWS API)

[CreateServiceLinkedRole](#) API コールを使用します。リクエストで、サービス名 (**SERVICE\_NAME\_URL**.amazonaws.com) を指定します。

たとえば、サービスにリンクされたロール ([Lex Bots]) を作成するには、lex.amazonaws.com を使用します。

## サービスにリンクされたロールの編集

サービスにリンクされたロールを編集するメソッドは、サービスによって異なります。一部のサービスでは、サービスコンソール、API、CLI からサービスにリンクされたロールのアクセス権限を編集することができます。ただし、サービスにリンクされたロールを作成すると多くのエンティティによりロールが参照されるため、ロール名を変更することはできません。ロールの説明は、IAM コンソール、API、CLI から編集することができます。

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールをサービスで編集できるかどうかを確認するには、「はい」リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを参照してください。

### サービスにリンクされたロールの説明の編集 (コンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

#### サービスにリンクされたロールの説明を編集するには (コンソール)

1. IAM コンソールのナビゲーションペインで [ロール] を選択します。
2. 変更するロールの名前を選択します。
3. [Role description] の右端にある [Edit] を選択します。
4. ボックスに新しい説明を入力し、[Save] を選択します。

### サービスにリンクされたロールの説明の編集 (AWS CLI)

AWS CLI から IAM コマンドを使用して、サービスにリンクされたロールの説明を編集できます。

#### サービスにリンクされたロールの説明を変更するには (AWS CLI)

1. (オプション) ロールの現在の説明を表示するには、以下のコマンドを実行します。

```
aws iam get-role --role-name ROLE-NAME
```

CLI コマンドでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。

2. サービスにリンクされたロールの説明を更新するには、次のコマンドを実行します。

```
aws iam update-role --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
```

### サービスにリンクされたロールの説明の編集 (AWS API)

サービスにリンクされたロールの説明は、AWS API を使用して編集できます。

### サービスにリンクされたロールの説明を変更するには (AWS API)

1. (オプション) ロールの現在の説明を表示するには、次のオペレーションを呼び出し、ロールの名前を指定します。

AWS API: [GetRole](#)

2. ロール説明を更新するには、次のオペレーションを呼び出し、ロールの名前 (およびオプションの説明) を指定します。

AWS API: [UpdateRole](#)

### サービスにリンクされたロールの削除

サービスにリンクされたロールを作成するメソッドは、サービスによって異なります。場合によっては、サービスにリンクされたロールを手動で削除する必要はありません。たとえば、サービス特定のアクション (リソースの削除) を完了すると、サービスによって、サービスにリンクされたロールが削除される場合があります。

また、サービスにリンクされたロールは、サービスコンソール、API、または AWS CLI から手動で削除できる場合があります。

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールをサービスで削除できるかどうかを確認するには、「はい」リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを参照してください。

サービスがロールの削除をサポートしていない場合は、IAM コンソール、API、または AWS CLI からサービスにリンクされたロールを削除できます。サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、



使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、削除する前に、サービスにリンクされたロールをクリーンアップする必要があります。

### サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。サービスにリンクされたロールのチェックボックスではなく、ロールの名前を選択します。
3. 選択したロールの [概要] ページで、[アクセスアドバイザー] タブを選択します。
4. [アクセスアドバイザー] タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

#### Note

サービスがサービスにリンクされたロールを使用しているかどうか不明な場合は、ロールの削除を試みることができます。サービスがロールを使用している場合、削除は失敗し、ロールが使用されているリージョンを表示できます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

サービスにリンクされたロールによって使用されているリソースを削除するには

サービスにリンクされたロールを使用してサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスにリンクされたロールをサービスで削除できるかどうかを確認するには、「はい」リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを参照してください。サービスにリンクされているロールが使用しているリソースを削除する方法については、そのサービスのドキュメントを参照してください。

## サービスにリンクされたロールの削除 (コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを削除できます。

### サービスにリンクされたロールを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。ロール名または行そのものではなく、削除するロール名の横にあるチェックボックスをオンにします。
3. ページ上部にある [ロールのアクション] で [削除] を選択します。
4. 確認ダイアログボックスで、最終アクセス情報を確認します。これは、選択したそれぞれのロールの AWS サービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、[Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。
5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送信すると、削除タスクは成功または失敗する可能性があります。
  - タスクが成功した場合は、ロールがリストから削除され、成功の通知がページの上部に表示されます。
  - タスクが失敗した場合は、通知から [View details] (詳細を表示) または [View Resources] (リソースを表示) を選択して、削除が失敗した理由を知ることができます。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次に [リソースをクリーンアップ](#) してから、削除リクエストをもう一度送信できます。

#### Note

サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。たとえば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。

- タスクが失敗し、通知にリソースのリストが含まれていない場合、サービスはその情報を返さない可能性があります。サービスのリソースをクリーンアップする方法の詳細については、「[IAM と連携する AWS のサービス](#)」を参照してください。使用しているサービスをテーブルで見つけ、「はい」リンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

## サービスにリンクされたロールの削除 (AWS CLI)

AWS CLI から IAM コマンドを使用して、サービスにリンクされたロールを削除できます。

### サービスにリンクされたロールを削除するには (AWS CLI)

1. 削除するロールの名前が分からない場合、以下のコマンドを入力してお客様のアカウントにあるロールを表示します。

```
aws iam get-role --role-name role-name
```

CLI コマンドでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。

2. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `deletion-task-id` を取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、次のコマンドを入力します。

```
aws iam delete-service-linked-role --role-name role-name
```

3. 削除タスクのステータスを確認するには、次のコマンドを入力します。

```
aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

削除タスクのステータスは、`NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次に [リソースをクリーンアップ](#)してから、削除リクエストをもう一度送信できます。

**Note**

サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。たとえば、サービスにリンクされたロールが6つのリソースを使用しており、サービスはそのうち5つのリソースに関する情報を返すことがあります。5つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの1つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。リソースを報告しないサービスのリソースをクリーンアップする方法の詳細については、「[IAM と連携する AWS のサービス](#)」を参照してください。使用しているサービスをテーブルで見つけ、「はい」リンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

### サービスにリンクされたロールの削除 (AWS API)

AWS API を使用して、サービスにリンクされたロールを削除できます。

#### サービスにリンクされたロールを削除するには (AWS API)

1. サービスにリンクされたロールの削除リクエストを送信するには、[DeleteServiceLinkedRole](#) を呼び出します。リクエストで、ロール名を指定します。

サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから DeletionTaskId を取得して、削除タスクのステータスを確認する必要があります。

2. 削除タスクのステータスを確認するには、[GetServiceLinkedRoleDeletionStatus](#) を呼び出します。リクエストで DeletionTaskId を指定します。

削除タスクのステータスは、NOT\_STARTED、IN\_PROGRESS、SUCCEEDED、または FAILED となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。ロールがサービスのリソースを使用しているために削除が失敗したとき、サービスがその情報を返す場合は、通知にはリソースのリストが含まれます。次に[リソースをクリーンアップ](#)してから、削除リクエストをもう一度送信できます。

**Note**

サービスが返す情報に応じて、このプロセスを何度も繰り返す必要があります。たとえば、サービスにリンクされたロールが 6 つのリソースを使用しており、サービスはそのうち 5 つのリソースに関する情報を返すことがあります。5 つのリソースをクリーンアップして削除するロールを再度送信すると、削除は失敗し、残りの 1 つのリソースが報告されます。サービスはすべてのリソースを返しますが、そのうちいくつかはリソースを報告しない場合もあります。リソースを報告しないサービスのリソースをクリーンアップする方法の詳細については、「[IAM と連携する AWS のサービス](#)」を参照してください。使用しているサービスをテーブルで見つけ、「はい」リンクを選択すると、そのサービスのサービスにリンクされたロールに関するドキュメントが表示されます。

## IAM ロールの作成

ロールを作成するには、AWS Management Console、AWS CLI、Tools for Windows PowerShell、または IAM API を使用できます。

AWS Management Consoleを使用する場合は、ウィザードを使用し、一連のステップに従ってロールを作成できます。AWS サービス向け、AWS アカウント 向け、またはフェデレーションユーザー向けのうち、どのロールを作成するかによって、ウィザードの手順は少し異なります。

### トピック

- [IAM ユーザーにアクセス許可を委任するロールの作成](#)
- [AWS のサービスにアクセス許可を委任するロールの作成](#)
- [サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)
- [カスタム信頼ポリシーを使用したロールの作成 \(コンソール\)](#)
- [アクセス権を委任するポリシーの例](#)

## IAM ユーザーにアクセス許可を委任するロールの作成

IAM ロールを使用することで、お客様の AWS リソースに対するアクセス許可を委任できます。IAM ロールにより、お客様の信頼するアカウントと他の AWS 信頼される アカウントとの信頼関係を確立できます。信頼するアカウントは、アクセスされるリソースを所有し、信頼されるアカウントは、リソースへのアクセスを必要とするユーザーを含みます。ただし、別のアカウントがお客様のアカウントのリソースを所有する場合があります。たとえば、信頼するアカウントが、新しいリソースの

作成 (Amazon S3 バケットでの新しいオブジェクトの作成など) を、信頼されたアカウントに許可する場合があります。この場合、リソースを作成したアカウントがリソースを所有します。さらに、リソースへのアクセスを誰に許可するかも管理します。

信頼関係の作成後、IAM ユーザーまたは信頼されたアカウントのアプリケーションでは AWS Security Token Service (AWS STS) [AssumeRole](#) の API オペレーションを使用できます。このオペレーションから提供される一時的なセキュリティ認証情報を使用して、お客様のアカウントの AWS リソースにアクセスできます。

いずれものアカウントもお客様が管理できます。または、ユーザーが属するアカウントは第三者が管理できます。ユーザーが属する他のアカウントが管理対象外の AWS アカウント である場合は、externalId 属性を使用することもできます。外部 ID は、お客様とサードパーティーのアカウントの管理者との間で同意した任意の単語または数値です。このオプションにより、リクエストに正しい sts:ExternalID が含まれている場合にのみユーザーがロールを引き受けることができるという条件が、信頼ポリシーに自動的に追加されます。詳細については、「[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#)」を参照してください。

ロールを使用してアクセス権限を委任する方法の詳細については、「[ロールに関する用語と概念](#)」を参照してください。サービスロールを使用して、サービスからアカウントのリソースへのアクセスを許可する方法については、「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。

## IAM ロールの作成 (コンソール)

IAM ユーザーが引き受けるロールは、AWS Management Console を使用して作成できます。例えば、組織で複数の AWS アカウント を使用して本稼働環境から開発環境を分離しているとします。この場合、開発用アカウントのユーザーに対して、本番稼働用アカウントのリソースへのアクセスを許可するロールを作成する方法の概要情報については、「[個別の開発用アカウントと本稼働用アカウントを使用したシナリオ例](#)」を参照してください。

### ロールを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. [AWS アカウント] のロールタイプを選択します。
4. アカウントのロールを作成するには、[This account] (このアカウント) を選択します。別のアカウントのロールを作成するには、[Another AWS アカウント] (別の) を選択し、リソースへのアクセス許可を付与する [Account ID] (アカウント ID) を入力します。



指定したアカウントの管理者は、そのアカウントのすべての IAM ユーザーに、このロールを引き受ける許可を付与できます。そのためには、管理者から `sts:AssumeRole` アクションの許可を付与するユーザーまたはグループにポリシーを添付します。そのポリシーで、Resource としてロールの ARN を指定する必要があります。

5. 管理対象外のアカウントのユーザーにアクセス許可を付与し、このロールをユーザーがプログラムで引き受ける場合は、[Require external ID] (外部 ID が必要) を選択します。外部 ID は、お客様とサードパーティーのアカウントの管理者との間で同意した任意の単語または数値です。このオプションにより、リクエストに正しい `sts:ExternalID` が含まれている場合にのみユーザーがロールを引き受けることができるという条件が、信頼ポリシーに自動的に追加されます。詳細については、「[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#)」を参照してください。

#### Important

このオプションを選択すると、AWS CLI、Tools for Windows PowerShell、または AWS API からのみの、ロールへのアクセスが制限されます。これは、AWS コンソールを使用して、その信頼ポリシーに `externalId` 条件が指定されているロールに切り替えることができないためです。ただし、該当する SDK を使用してスクリプトまたはアプリケーションを作成することで、この種のアクセスをプログラムから実行することができます。詳細とサンプルスクリプトについては、AWS Management Console セキュリティログの「[AWS へのクロスアカウントアクセスを有効にする方法](#)」を参照してください。

6. 多要素認証 (MFA) を使用してサインインするユーザーにロールを制限するには、[MFA が必要] オプションを選択します。これにより、MFA によるサインインの有無を確認する条件がロールの信頼ポリシーに追加されます。ロールを引き受けるユーザーは、設定した MFA デバイスから一時的なワンタイムパスワードを使用してサインインする必要があります。MFA 認証のないユーザーはロールを引き受けることができません。MFA の詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。
7. [Next] を選択します。
8. IAM には、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。アクセス許可ポリシーとして使用するポリシーを選択するか、[ポリシーの作成] を選択して新しいブラウザタブを開き、新しいポリシーをゼロから作成します。詳細については、「[IAM ポリシーの作成](#)」を参照してください。ポリシーを作成したら、そのタブを閉じて元のタブに戻ります。ロールを引き受けるユーザーに許可するアクセス許可ポリシーの横にあるチェックボックス



をオンにします。必要に応じて、この時点でポリシーを選択せずに、後でポリシーをロールにアタッチすることもできます。デフォルトでは、ロールにはいずれのアクセス権限もありません。

9. (オプション) [アクセス許可の境界](#)を設定します。これはアドバンスド機能です。

[Set permissions boundary (アクセス許可の境界の設定)] セクションを開き、[Use a permissions boundary to control the maximum role permissions (アクセス許可の境界を使用してロールのアクセス許可の上限を設定する)] を選択します。アクセス許可の境界として使用するポリシーを選択します。

10. [Next] を選択します。
11. ロール名 に、ロールの名前を入力します。ロール名は AWS アカウント アカウント内で一意である必要があります。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。サインイン処理中など、コンソールでロール名がユーザーに表示される場合、ロール名は大文字と小文字を区別しません。さまざまなエンティティがロールを参照する可能性があるため、作成後にロール名を編集することはできません。
12. (オプション) [Description (説明)] には、新しいロールの説明を入力します。
13. [Step 1: Select trusted entities] (ステップ 1: 信頼済みエンティティの選択) または [Step 2: Add permissions] (ステップ 2: 権限の追加) のセクションで [Edit] (編集) を選択し、ロールのユースケースと権限を変更します。前のページに戻って編集を行います。
14. (オプション) タグをキーバリューのペアとしてアタッチして、メタデータをロールに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
15. ロール情報を確認し、ロールの作成 を選択します。

#### Important

上記の手順は、必要となる設定の前半であることに注意してください。信頼されたアカウントの各ユーザーに対して、コンソールでロールを切り替えるか、プログラムでロールを引き受けるためのアクセス許可も付与する必要があります。この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

## IAM ロール (AWS CLI) の作成

AWS CLI を使用したロールの作成には、複数のステップがあります。コンソールを使用してロールを作成する場合、多くのステップは自動的に行われますが、AWS CLI を使用する場合は、各ステッ

プを明示的に実行する必要があります。ロールを作成して、これにアクセス許可ポリシーを割り当てる必要があります。必要に応じて、ロールの[アクセス許可の境界](#)を設定することもできます。

クロスアカウントアクセス用のロールを作成するには (AWS CLI)

1. ロール [aws iam create-role](#) を作成します。
2. マネージドアクセス許可ポリシー [aws iam attach-role-policy](#) をロールにアタッチします。

または

ロールのインラインアクセス許可ポリシー [aws iam put-role-policy](#) を作成します。

3. (オプション) タグ ([aws iam tag-role](#)) をアタッチして、カスタム属性をロールに追加します。

詳細については、「[IAM ロールのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

4. (オプション) ロールの[アクセス許可の境界](#) [aws iam put-role-permissions-boundary](#) を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

次の例では、シンプルな環境でクロスアカウントロールを作成するための最初の 2 つのステップ (最も一般的なステップ) を示します。この例では、123456789012 アカウントの任意のユーザーに、ロールを引き受けて Amazon S3 バケット example\_bucket を表示することを許可します。また、この例では、Windows を実行しているクライアントコンピュータを使用中であり、アカウントの認証情報とリージョンを使ってコマンドラインインターフェイスを設定済みであることを前提とします。詳細については、「[AWS コマンドラインインターフェイスの設定](#)」を参照してください。

この例では、ロールの作成時に、次の信頼ポリシーを最初のコマンドに含めます。この信頼ポリシーでは、123456789012 アカウントのユーザーに対して、AssumeRole オペレーションを使用してロールを引き受けることを許可します。ただし、ユーザーが SerialNumber パラメータと TokenCode パラメータを使用して MFA 認証を提供することを条件とします。MFA の詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
    "Action": "sts:AssumeRole",
    "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }
  }
]
}
```

### ⚠ Important

Principal 要素に特定の IAM ロールまたはユーザーの ARN が含まれている場合、この ARN はポリシーの保存時に一意のプリンシパル ID に変換されます。これにより、第三者がロールやユーザーを削除して再作成することで、そのユーザーのアクセス許可をエスカレートするリスクを緩和できます。通常、この ID はコンソールには表示されません。信頼ポリシーが表示されるときに、ARN への逆変換が行われるためです。ただし、ロールまたはユーザーを削除すると、プリンシパル ID はコンソールに表示されます。これは、AWS が ARN に ID をマッピングできなくなるためです。したがって、信頼ポリシーの Principal 要素で指し示しているユーザーまたはロールを削除し、再作成する場合、ロールを編集して ARN を置き換える必要があります。

2 番目のコマンドを使用する場合、既存の管理ポリシーをロールにアタッチする必要があります。以下のアクセス許可ポリシーでは、example\_bucket Amazon S3 バケットに対して ListBucket アクションのみを実行することを、ロールを引き受ける任意のユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::example_bucket"
    }
  ]
}
```

この Test-UserAccess-Role ロールを作成するには、まず以前の信頼ポリシーを trustpolicyforacct123456789012.json という名前でローカル policies ドライブの C: フォルダに保存する必要があります。次に、以前のアクセス許可ポリシーを PolicyForRole とい

う名前のカスタマー管理ポリシーとして AWS アカウント に保存します。次に、以下のコマンドを使用してロールを作成し、管理ポリシーをアタッチできます。

```
# Create the role and attach the trust policy file that allows users in the specified
account to assume the role.
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document
file:///C:\policies\trustpolicyforacct123456789012.json

# Attach the permissions policy (in this example a managed policy) to the role to
specify what it is allowed to do.
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn
arn:aws:iam::123456789012:policy/PolicyForRole
```

### Important

上記の手順は、必要となる設定の前半であることに注意してください。信頼されたアカウントに属している個々のユーザーに、ロールを切り替えるアクセス権限を付与する必要もあります。この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

ロールを作成し、このロールに AWS タスクの実行や AWS リソースへのアクセスに必要なアクセス許可を付与すると、123456789012 アカウントのユーザーはこのロールを引き受けることができます。詳細については、「[IAM ロール \(AWS CLI\) の切り替え](#)」を参照してください。

## IAM ロール (AWS API) の作成

AWS API からロールを作成するには、複数のステップが必要です。コンソールでロールを作成する場合は多くのステップが自動的に実行されますが、API では各ステップを手動で明示的に実行する必要があります。ロールを作成して、これにアクセス許可ポリシーを割り当てる必要があります。必要に応じて、ロールの[アクセス許可の境界](#)を設定することもできます。

コードでロールを作成するには (AWS API)

1. ロールとして [CreateRole](#) を作成します。

ロールの信頼ポリシーに対して、ファイルの場所を指定できます。

2. マネージドアクセス許可ポリシー [AttachRolePolicy](#) をロールにアタッチします。

または

ロールのインラインアクセス許可ポリシー [PutRolePolicy](#) を作成します。

**⚠ Important**

上記の手順は、必要となる設定の前半であることに注意してください。信頼されたアカウントに属している個々のユーザーに、ロールを切り替えるアクセス権限を付与する必要もあります。この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

3. (オプション) タグ ([TagRole](#)) をアタッチして、カスタム属性をユーザーに追加します。

詳細については、「[IAM ユーザーのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

4. (オプション) ロールの [アクセス許可の境界 PutRolePermissionsBoundary](#) を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

ロールを作成し、このロールに AWS タスクの実行や AWS リソースへのアクセスに必要なアクセス許可を付与したら、アカウントのユーザーにアクセス許可を付与して、このロールを引き受けることを許可する必要があります。ロールの引き受けに関する詳細については、「[IAM ロール \(AWS API\) の切り替え](#)」を参照してください。

## IAM ロール (AWS CloudFormation) の作成

AWS CloudFormation での IAM ロールの作成については、[AWS CloudFormation ユーザーガイド](#)の「[リソースとプロパティのリファレンス](#)」および「例」を参照してください。

AWS CloudFormation の IAM テンプレートの詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS Identity and Access Management テンプレートスニペット](#)」を参照してください。

## AWS のサービスにアクセス許可を委任するロールの作成

AWS の多くのサービスでは、ロールを使用して、ユーザーに代わって該当サービスが他のサービスのリソースにアクセスすることを許可する必要があります。サービスがお客様に代わってアクションを実行するために引き受けるロールは、[サービスロール](#)と呼ばれます。ロールにサービスに対して特殊な目的がある場合、そのロールは [EC2 インスタンスのサービスロール](#) (この例)、または [サービスにリンクされたロール](#)として分類されます。どのサービスがサービスにリンクされたロールの使用

をサポートしているのか、またはサービスがあらゆる形式の一時的な認証情報をサポートしているのか確認するには「[IAM と連携する AWS のサービス](#)」をご覧ください。サービスがそれぞれロールをどのように使用するのか把握するには、テーブル内のサービス名を選択し、そのサービスに関するドキュメントをご覧ください。

PassRole アクセス許可を設定する場合、ユーザーがロールに必要な以上のアクセス許可があるロールを渡さないようにする必要があります。例えば、Alice が Amazon S3 アクションを実行する許可を持っていない場合があります。Alice が Amazon S3 アクションを許可するサービスにロールを渡すことができる場合、サービスはジョブの実行時に、Alice に代わって Amazon S3 アクションを実行できます。

ロールを使用してアクセス許可を委任する方法の詳細については、「[ロールに関する用語と概念](#)」を参照してください。

### サービスロールのアクセス許可

IAM エンティティ (ユーザーまたはロール) がサービスロールを作成または編集できるようにするには、アクセス許可を設定する必要があります。

#### Note

サービスにリンクされたロールの ARN にはサービスプリンシパルが含まれています。以下のポリシーでは **SERVICE-NAME**.amazonaws.com として示されています。サービスプリンシパルは推量しないでください。サービスプリンシパルは、大文字と小文字が区別され、AWS のサービス間で異なる場合があります。サービスのサービスプリンシパルを表示するには、そのサービスにリンクされたロールのドキュメントを参照してください。

IAM エンティティが特定のサービスロールを作成することを許可するには

サービスロールを作成する必要がある IAM エンティティに、以下のポリシーを追加します。このポリシーでは、指定したサービスおよび特定の名称のサービスロールの作成を許可します。管理ポリシーまたはインラインポリシーをそのロールにアタッチできます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
      "iam:AttachRolePolicy",
      "iam:CreateRole",
      "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
  }
]
}

```

IAM エンティティが任意のサービスロールを作成することを許可するには

AWS では、管理者ユーザーのみにサービスロールの作成を許可することをお勧めします。ロールの作成とポリシーのアタッチを許可されたユーザーは、自分のアクセス許可をエスカレートできます。代わりに、彼らが必要とするロールの作成のみを許可したポリシーを作成するか、彼らの代わりに、管理者にサービスロールを作成させます。

管理者が AWS アカウント 全体にアクセスできるポリシーをアタッチするには、[AdministratorAccess](#) AWS 管理ポリシーを使用します。

IAM エンティティにサービスロールの編集を許可するには

サービスロールを編集する必要がある IAM エンティティに、以下のポリシーを追加します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EditSpecificServiceRole",
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam>DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam>ListAttachedRolePolicies",
        "iam>ListRolePolicies",
        "iam:PutRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
    }
  ]
}

```



```
    },
    {
      "Sid": "ViewRolesAndPolicies",
      "Effect": "Allow",
      "Action": [
        "iam:GetPolicy",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM エンティティが特定のサービスロールを削除することを許可するには

指定したサービスロールを削除する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

```
{
  "Effect": "Allow",
  "Action": "iam:DeleteRole",
  "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
}
```

IAM エンティティがサービスロールを削除することを許可するには

AWS では、管理者ユーザーのみにサービスロールの削除を許可することをお勧めします。代わりに、彼らが必要とするロールの削除のみを許可したポリシーを作成するか、彼らの代わりに、管理者にサービスロールを削除させます。

管理者が AWS アカウント 全体にアクセスできるポリシーをアタッチするには、[AdministratorAccess](#) AWS 管理ポリシーを使用します。

### AWS のサービス用ロールの作成 (コンソール)

サービス用のロールを作成するには、AWS Management Console を使用します。一部のサービスでは、複数のサービスロールがサポートされているため、該当サービスの「[AWS ドキュメント](#)」を参照の上、選択するユースケースを確認してください。必要な信頼ポリシーとアクセス権限ポリシーを割り当て、サービスがお客様に代わってロールを引き受ける方法について説明します。ロールのアクセス許可を管理するために使用できるステップは、サービスでユースケースを定義する方法や、サービスにリンクされたロールを作成するかどうかに応じて異なります。

AWS のサービス (IAM コンソール) のロールを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
3. 信頼できるエンティティタイプで、AWS のサービス を選択します。
4. [サービスまたはユースケース] でサービスを選択し、次にユースケースを選択します。ユースケースは、サービスに必要な信頼ポリシーを含める定義になります。
5. [Next] を選択します。
6. [アクセス許可ポリシー] では、オプションは選択したユースケースによって異なります。
  - サービスがロールのアクセス許可を定義している場合、アクセス許可ポリシーを選択することはできません。
  - 制限されたアクセス許可ポリシーのセットから選択します。
  - すべてのアクセス許可ポリシーから選択します。
  - アクセス許可ポリシーを選択せずに、ロールの作成後にポリシーを作成し、そのポリシーをロールにアタッチします。
7. (オプション) [アクセス許可の境界](#)を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。
  - a. [アクセス許可の境界の設定] セクションを開き、[アクセス許可の境界を使用してロールのアクセス許可の上限を設定する] を選択します。

IAM には、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。
  - b. アクセス許可の境界として使用するポリシーを選択します。
8. [Next] を選択します。
9. [ロール名] では、オプションはサービスによって異なります。
  - サービスでロール名が定義されている場合、ロール名を編集することはできません。
  - サービスでロール名のプレフィックスが定義されている場合、オプションのサフィックスを入力できます。
  - サービスでロール名が定義されていない場合、ロールに名前を付けることができます。

**⚠ Important**

ロールに名前を付けるときは、次のことに注意してください。

- ロール名は AWS アカウント内で一意である必要があります。ただし、大文字と小文字は区別されません。

例えば、**PRODRROLE** と **prodrole** の両方の名前で作成することはできません。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。ただし、サインインプロセスなど、コンソールにロール名がユーザーに表示される場合、ロール名は大文字と小文字が区別されません。

- 他のエンティティがロールを参照する可能性があるため、ロールを作成した後にロール名を編集することはできません。

10. (オプション) [説明] にロールの説明を入力します。
11. (オプション) ロールのユースケースとアクセス許可を編集するには、[ステップ 1: 信頼されたエンティティを選択] または [ステップ 2: アクセス権限を追加] のセクションで [編集] を選択します。
12. (オプション) ロールの識別、整理、検索を簡単にするには、キーと値のペアとしてタグを追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「[IAM リソースにタグを付ける](#)」を参照してください。
13. ロールを確認したら、[Create role] (ロールを作成) を選択します。

## サービス用のロールを作成する (AWS CLI)

AWS CLI を使用したロールの作成には、複数のステップがあります。コンソールを使用してロールを作成する場合、多くのステップは自動的に行われますが、AWS CLI を使用する場合は、各ステップを明示的に実行する必要があります。ロールを作成して、これにアクセス許可ポリシーを割り当てる必要があります。使用しているサービスが Amazon EC2 の場合は、インスタンスプロファイルも作成してロールを追加する必要があります。必要に応じて、ロールの[アクセス許可の境界](#)を設定することもできます。

AWS のサービスのロールを AWS CLI で作成するには

1. 次の [create-role](#) コマンドは、Test-Role という名前のロールを作成し、それに信頼ポリシーをアタッチします。

```
aws iam create-role --role-name Test-Role --assume-role-policy-document
file://Test-Role-Trust-Policy.json
```

- マネージドアクセス許可ポリシー [aws iam attach-role-policy](#) をロールにアタッチします。

たとえば、次の `attach-role-policy` コマンドは、AWS と呼ばれる `ReadOnlyAccess` 管理ポリシーIAM ロールを `ReadOnlyRole` と呼ばれる IAM ロールにアタッチします。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
ReadOnlyAccess --role-name ReadOnlyRole
```

または

ロールのインラインアクセス許可ポリシー [aws iam put-role-policy](#) を作成します。

インラインアクセス許可ポリシーの追加については、以下の例を参照してください。

```
aws iam put-role-policy --role-name Test-Role --policy-name
ExamplePolicy --policy-document file://AdminPolicy.json
```

- (オプション) タグ ([aws iam tag-role](#)) をアタッチして、カスタム属性をロールに追加します。

詳細については、「[IAM ロールのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

- (オプション) ロールの [アクセス許可の境界](#) `aws iam put-role-permissions-boundary` を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

ロールを Amazon EC2 で使用する場合や、Amazon EC2 を使用する AWS の別のサービスで使用する場合は、ロールをインスタンスプロファイルに保存する必要があります。インスタンスプロファイルは、Amazon EC2 インスタンスの起動時にインスタンスにアタッチできるロールのコンテナです。インスタンスプロファイルに含めることができる ロールは 1 つのみであり、緩和できません。AWS Management Consoleを使用してロールを作成すると、ロールと同じ名前のインスタンスプロファイルが自動的に作成されます。インスタンスプロファイルの詳細については、「[インスタンスプロファイルの使用](#)」を参照してください。ロールを使用して EC2 インスタンスを起動する方法については、「Amazon EC2 ユーザーガイド」の [Amazon EC2 リソースへのアクセスの制御](#)に関するページを参照してください。

インスタンスプロファイルを作成し、これにロールを保存するには (AWS CLI)

1. インスタンスプロファイル [aws iam create-instance-profile](#) を作成します。
2. ロールをインスタンスプロファイル [aws iam add-role-to-instance-profile](#) に追加します。

次に示す AWS CLI のコマンド例では、ロールを作成してアクセス許可をアタッチする手順の最初の 2 つのステップを示します。また、インスタンスプロファイルを作成し、これにロールを追加する手順の最初の 2 つのステップも示します。この信頼ポリシー例では、ロールを引き受けて Amazon S3 バケット `example_bucket` を表示することを Amazon EC2 サービスに許可します。また、この例では、Windows を実行するクライアントコンピュータを使用中であり、アカウントの認証情報とリージョンを使ってコマンドラインインターフェイスを設定済みであることを前提とします。詳細については、「[AWS コマンドラインインターフェイスの設定](#)」を参照してください。

この例では、ロールの作成時に、次の信頼ポリシーを最初のコマンドに含めます。この信頼ポリシーにより、Amazon EC2 サービスはロールを引き受けることを許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

2 番目のコマンドを使用する場合、アクセス許可ポリシーをロールにアタッチする必要があります。次のアクセス許可ポリシーの例では、Amazon S3 バケット `ListBucket` に対して `example_bucket` アクションのみを実行することをロールに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

この `Test-Role-for-EC2` ロールを作成するには、まず前の信頼ポリシーを `trustpolicyforec2.json` という名前で、前のアクセス許可ポリシーを

permissionspolicyforec2.json という名前で、ローカル C: ドライブの policies ディレクトリに保存する必要があります。次に、以下のコマンドを使用して、ロールの作成、ポリシーのアタッチ、インスタンスプロファイルの作成、およびインスタンスプロファイルへのロールの追加を行います。

```
# Create the role and attach the trust policy that allows EC2 to assume this role.
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document
  file:///C:\policies\trustpolicyforec2.json

# Embed the permissions policy (in this example an inline policy) to the role to
  specify what it is allowed to do.
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-
  Policy-For-Ec2 --policy-document file:///C:\policies\permissionspolicyforec2.json

# Create the instance profile required by EC2 to contain the role
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3

# Finally, add the role to the instance profile
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --
  role-name Test-Role-for-EC2
```

EC2 インスタンスを起動するとき、AWS コンソールを使用する場合は、[インスタンスの詳細の設定] ページでインスタンスプロファイル名を指定します。aws ec2 run-instances CLI コマンドを使用する場合は、--iam-instance-profile パラメータを指定します。

### サービス用のロールを作成する (AWS API)

AWS API からロールを作成するには、複数のステップが必要です。コンソールでロールを作成する場合は多くのステップが自動的に実行されますが、API では各ステップを手動で明示的に実行する必要があります。ロールを作成して、これにアクセス許可ポリシーを割り当てる必要があります。使用しているサービスが Amazon EC2 の場合は、インスタンスプロファイルも作成してロールを追加する必要があります。必要に応じて、ロールの[アクセス許可の境界](#)を設定することもできます。

### AWS のサービスのロールを作成するには (AWS API)

1. ロールとして [CreateRole](#) を作成します。

ロールの信頼ポリシーに対して、ファイルの場所を指定できます。

2. ロールに管理アクセス許可ポリシーをアタッチします: [AttachRolePolicy](#)

または

ロールのインラインアクセス許可ポリシー [PutRolePolicy](#) を作成します。

3. (オプション) タグ ([TagRole](#)) をアタッチして、カスタム属性をユーザーに追加します。

詳細については、「[IAM ユーザーのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

4. (オプション) ロールの [アクセス許可の境界 PutRolePermissionsBoundary](#) を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

ロールを Amazon EC2 で使用する場合や、Amazon EC2 を使用する AWS の別のサービスで使用する場合は、ロールをインスタンスプロファイルに保存する必要があります。インスタンスプロファイルは、ロールのコンテナとして機能します。各インスタンスプロファイルに含めることができるロールは 1 つのみであり、増やすことはできません。AWS Management Console でロールを作成すると、ロールと同じ名前のインスタンスプロファイルが自動的に作成されます。インスタンスプロファイルの詳細については、「[インスタンスプロファイルの使用](#)」を参照してください。ロールを使用して Amazon EC2 インスタンスを起動する方法については、「Amazon EC2 ユーザーガイド」の [Amazon EC2 リソースへのアクセスの制御](#) に関するページを参照してください。

インスタンスプロファイルを作成し、これにロールを保存するには (AWS API)

1. インスタンスプロファイル [CreateInstanceProfile](#) を作成します。
2. ロールをインスタンスプロファイル [AddRoleToInstanceProfile](#) に追加します。

## サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成

AWS アカウントで IAM ユーザーを作成する代わりに、ID プロバイダーを使用できます。ID プロバイダー (IdP) を使用すると、AWS の外部のユーザー ID を管理して、これらの外部ユーザー ID にアカウント内の AWS リソースに対するアクセス許可を付与できます。フェデレーションおよび認証プロバイダーについて詳しくは、「[ID プロバイダーとフェデレーション](#)」を参照してください。

### フェデレーテッドユーザーのロールの作成 (コンソール)

フェデレーテッドユーザーのロールを作成する手順は、選択したサードパーティープロバイダーによって異なります。



- OpenID Connect (OIDC) については、「[OpenID Connect フェデレーション用のロールを作成する \(コンソール\)](#)」を参照してください。
- SAML 2.0 については、「[SAML 2.0 フェデレーション用のロールを作成する \(コンソール\)](#)」を参照してください。

### フェデレーションアクセス用のロールの作成 (AWS CLI)

サポートされている ID プロバイダー (OIDC または SAML) 用のロールを AWS CLI から作成するステップは同じです。違いは、前提条件のステップで作成する信頼ポリシーの内容です。使用するプロバイダーのタイプに合わせた前提条件セクションのステップに従って開始します。

- OIDC プロバイダーについては、「[OIDC 用のロールを作成するための前提条件](#)」を参照してください。
- SAML プロバイダーについては、「[SAML 用のロールを作成するための前提条件](#)」を参照してください。

AWS CLI を使用したロールの作成には、複数のステップがあります。コンソールを使用してロールを作成する場合、多くのステップは自動的に行われますが、AWS CLI を使用する場合は、各ステップを明示的に実行する必要があります。ロールを作成して、これにアクセス許可ポリシーを割り当てる必要があります。必要に応じて、ロールの[アクセス許可の境界](#)を設定することもできます。

### ID フェデレーションのロールを作成するには (AWS CLI)

1. ロール [aws iam create-role](#) を作成します。
2. アクセス許可ポリシー [aws iam attach-role-policy](#) をロールにアタッチします。

または

ロールのインラインアクセス許可ポリシー [aws iam put-role-policy](#) を作成します。

3. (オプション) タグ ([aws iam tag-role](#)) をアタッチして、カスタム属性をロールに追加します。

詳細については、「[IAM ロールのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

4. (オプション) ロールの[アクセス許可の境界](#) [aws iam put-role-permissions-boundary](#) を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

次の例では、シンプルな環境で ID プロバイダーのロールを作成するための最初の 2 つのステップ (最も一般的なステップ) を示します。この例では、123456789012 アカウントの任意のユーザーに、ロールを引き受けて Amazon S3 バケット `example_bucket` を表示することを許可します。また、この例では、Windows が動作しているコンピュータで AWS CLI を実行していること、さらに認証情報を使って AWS CLI を設定済みであることを前提とします。詳細については、「[Configuring the AWS Command Line Interface](#)」を参照してください。

次の例では、ユーザーが Amazon Cognito を使用してサインインする場合のモバイルアプリ用の信頼ポリシーを示します。この例では、`us-east:12345678-ffff-ffff-ffff-123456` は Amazon Cognito によって割り当てられた ID プール ID を表します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "RoleForCognito",
    "Effect": "Allow",
    "Principal": {"Federated": "cognito-identity.amazonaws.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east:12345678-ffff-ffff-ffff-123456"}}
  }
}
```

以下のアクセス許可ポリシーでは、Amazon S3 バケット `example_bucket` に対して `ListBucket` アクションのみを実行することを、ロールを引き受ける任意のユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

この `Test-Cognito-Role` ロールを作成するには、まず前の信頼ポリシーを `trustpolicyforcognitofederation.json` という名前で、前のアクセス許可ポリシーを `permpolicyforcognitofederation.json` という名前で、ローカル `policies` ドライブの `C:` フォルダに保存する必要があります。次に、以下のコマンドを使用してロールを作成し、インラインポリシーをアタッチできます。

```
# Create the role and attach the trust policy that enables users in an account to
assume the role.
$ aws iam create-role --role-name Test-Cognito-Role --assume-role-policy-document
file:///C:\policies\trustpolicyforcognitofederation.json

# Attach the permissions policy to the role to specify what it is allowed to do.
aws iam put-role-policy --role-name Test-Cognito-Role --policy-name
Perms-Policy-For-CognitoFederation --policy-document file:///C:\policies
\permspolicyforcognitofederation.json
```

## フェデレーションアクセス用のロールの作成 (AWS API)

サポートされている ID プロバイダー (OIDC または SAML) 用のロールを AWS CLI から作成するステップは同じです。違いは、前提条件のステップで作成する信頼ポリシーの内容です。使用するプロバイダーのタイプに合わせた前提条件セクションのステップに従って開始します。

- OIDC プロバイダーについては、「[OIDC 用のロールを作成するための前提条件](#)」を参照してください。
- SAML プロバイダーについては、「[SAML 用のロールを作成するための前提条件](#)」を参照してください。

## ID フェデレーションのロールを作成するには (AWS API)

1. ロールとして [CreateRole](#) を作成します。
2. アクセス許可ポリシーとして [AttachRolePolicy](#) をロールにアタッチします。

または

ロールのインラインアクセス許可ポリシー [PutRolePolicy](#) を作成します。

3. (オプション) タグ ([TagRole](#)) をアタッチして、カスタム属性をユーザーに追加します。

詳細については、「[IAM ユーザーのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

4. (オプション) ロールの [アクセス許可の境界 PutRolePermissionsBoundary](#) を設定します。

アクセス許可の境界では、ロールに許可されるアクセス許可の上限を設定します。アクセス許可の境界は AWS のアドバンスド機能です。

## OpenID Connect フェデレーション用のロールを作成する (コンソール)

AWS アカウントで AWS Identity and Access Management ユーザーを作成する代わりに、OpenID Connect (OIDC) フェデレーション ID プロバイダーを使用できます。ID プロバイダー (IdP) を使用すると、AWS の外部のユーザー ID を管理して、これらの外部ユーザー ID にアカウント内の AWS リソースに対するアクセス許可を付与できます。フェデレーションおよび IdPs (ID プロバイダー) について詳しくは、「[ID プロバイダーとフェデレーション](#)」を参照してください。

### OIDC 用のロールを作成するための前提条件

OIDC フェデレーション用のロールを作成する前に、まず次の基本的なステップを実行する必要があります。

### OIDC フェデレーション用のロールを作成する準備をするには

1. フェデレーション OIDC ID を提供する 1 つ以上のサービスにサインアップします。AWS リソースにアクセスする必要があるアプリを作成する場合は、プロバイダー情報も合わせてアプリを設定します。サインアップすると、プロバイダーからアプリに固有のアプリケーション ID または対象者 ID が提供されます。(プロバイダーが異なれば、このプロセスには異なる用語が使用されます。このガイドでは、プロバイダーでアプリを識別するプロセスに設定という用語を使用します。) プロバイダーごとに複数のアプリを設定できます。または 1 つのアプリに複数のプロバイダーを設定できます。ID プロバイダーの使用に関する情報は以下で確認してください。
  - [Login with Amazon 開発者センター](#)
  - Facebook 開発者サイトの「[アプリまたはウェブサイトに Facebook ログインを追加する](#)」
  - Google 開発者サイトの「[ログインでの OAuth 2.0 の使用 \(OpenID Connect\)](#)」
2. IdP から必要な情報を受け取ったら、IAM で IdP を作成します。詳細については、「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

#### Important

Google、Facebook、または Amazon Cognito の OIDC IdP を使用している場合は、AWS Management Console に個別の IAM IdP を作成しないでください。これらの OIDC ID プロバイダーは、すでに AWS に組み込まれており、ユーザーが使用できます。この手順をスキップして、次の手順で IdP を使用して新しいロールを作成します。

3. IdP 認証ユーザーが引き受けるロールのポリシーを準備します。すべてのロールに該当することですが、モバイルアプリ用のロールにも 2 つのポリシーがあります。1 つは、ロールの引き受け

先を指定する信頼ポリシーです。もう 1 つはアクセス許可ポリシーです。このポリシーでは、モバイルアプリにアクセスを許可または拒否する AWS アクションやリソースを指定します。

ウェブ IdP については、[Amazon Cognito](#) を使用して ID を管理することをお勧めします。この場合、次の例に示すような信頼ポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Federated": "cognito-identity.amazonaws.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east-2:12345678-abcd-abcd-abcd-123456"},
      "ForAnyValue:StringLike": {"cognito-identity.amazonaws.com:amr": "unauthenticated"}
    }
  }
}
```

*us-east-2:12345678-abcd-abcd-abcd-123456* は、Amazon Cognito が割り当てる ID プール ID に置き換えます。

OIDC IdP を手動で設定する場合は、信頼ポリシーの作成時に以下の 3 つの値を使用して当該アプリにのみロールを引き受けることを許可します。

- Action 要素で、`sts:AssumeRoleWithWebIdentity` アクションを使用します。
- Principal 要素で、`{"Federated":providerUrl/providerArn}` 文字列を使用します。
- 一部の一般的な OIDC IdP の場合、*providerUrl* は URL です。以下の例では、いくつかの一般的な IdP のプリンシパルを指定する方法を示します。

```
"Principal":{"Federated":"cognito-identity.amazonaws.com"}
```

```
"Principal":{"Federated":"www.amazon.com"}
```

```
"Principal":{"Federated":"graph.facebook.com"}
```

```
"Principal":{"Federated":"accounts.google.com"}
```

- 他の OIDC プロバイダーの場合は、次の例のように、[Step 2](#) で作成した OIDC IdP の Amazon リソースネーム (ARN) を使用します。

```
"Principal":{"Federated":{"arn:aws:iam::123456789012:oidc-provider/server.example.com"}}
```

- Condition 要素で、StringEquals 条件を使用してアクセス許可を制限します。ID プール ID (Amazon Cognito の場合) またはアプリ ID (他のプロバイダーの場合) をテストします。この ID プール ID は、IdP でアプリを設定したときに取得したアプリ ID と一致する必要があります。この ID 間の一致により、リクエストが自分のアプリからであることを確認できます。

#### Note

Amazon Cognito アイデンティティプールの IAM ロールは、サービスプリンシパル `cognito-identity.amazonaws.com` を信頼してロールを引き受けます。このタイプのロールには、ロールを引き受けるプリンシパルを制限する条件キーが少なくとも 1 つ含まれている必要があります。

[クロスアカウントの IAM ロール](#)を引き受ける Amazon Cognito アイデンティティプールには、その他の考慮事項が適用されます。これらのロールの信頼ポリシーは、`cognito-identity.amazonaws.com` サービスプリンシパルを受け入れる必要があります。ロールの引き受けを目的のアイデンティティプールのユーザーに制限するために `aud` 条件キーを含める必要があります。この条件なしで Amazon Cognito アイデンティティプールを信頼するポリシーでは、意図しないアイデンティティプールのユーザーがロールを引き受けるリスクが生じます。詳細については、「[Amazon Cognito デベロッパーガイド](#)」の「[基本 \(クラシック\) 認証の IAM ロールの信頼ポリシー](#)」を参照してください。

使用している IdP に応じて、以下の例に示すような、条件要素を作成します。

```
"Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud":  
"us-east:12345678-ffff-ffff-ffff-123456"}}
```

```
"Condition": {"StringEquals": {"www.amazon.com:app_id":  
"amzn1.application-oa2-123456"}}
```

```
"Condition": {"StringEquals": {"graph.facebook.com:app_id":  
"111222333444555"}}
```

```
"Condition": {"StringEquals": {"accounts.google.com:aud":  
"66677788899900pro0"}}
```

OIDC プロバイダーの場合は、次の例に示すように、OIDC IdP の完全修飾 URL と aud コンテキストキーを使用します。

```
"Condition": {"StringEquals": {"server.example.com:aud":  
"appid_from_oidc_idp"}}
```

#### Note

ロールの信頼ポリシーでプリンシパルとして指定する値は、IdP ごとに異なります。OIDC 用のロールで指定できるプリンシパルは 1 つだけです。したがって、モバイルアプリで複数の IdP からのサインインをユーザーに許可する場合は、サポートする IdP ごとに別個のロールを作成します。IdP ごとに別個の信頼ポリシーを作成します。

ユーザーがモバイルアプリを使用して Login with Amazon からサインインする場合、次の例の信頼ポリシーが適用されます。この例では、*amzn1.application-oa2-123456* は Login with Amazon を使用するアプリを設定したときに Amazon が割り当てるアプリ ID を表します。

```
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Sid": "RoleForLoginWithAmazon",  
    "Effect": "Allow",  
    "Principal": {"Federated": "www.amazon.com"},  
    "Action": "sts:AssumeRoleWithWebIdentity",  
    "Condition": {"StringEquals": {"www.amazon.com:app_id":  
"amzn1.application-oa2-123456"}}  ]  
}
```

ユーザーがモバイルアプリを使用して Facebook からサインインする場合、次の例の信頼ポリシーが適用されます。この例では、*111222333444555* は Facebook が割り当てるアプリ ID を表します。



```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForFacebook",
    "Effect": "Allow",
    "Principal": {"Federated": "graph.facebook.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"graph.facebook.com:app_id":
"111222333444555"}}
  ]
}
```

ユーザーがモバイルアプリを使用して Google からサインインする場合、次の例の信頼ポリシーが適用されます。この例では、**666777888999000** は Google が割り当てるアプリ ID を表します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForGoogle",
    "Effect": "Allow",
    "Principal": {"Federated": "accounts.google.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"accounts.google.com:aud":
"666777888999000"}}
  ]
}
```

ユーザーがモバイルアプリを使用して Amazon Cognito からサインインする場合、次の例の信頼ポリシーが適用されます。この例では、**us-east:12345678-ffff-ffff-ffff-123456** は Amazon Cognito が割り当てる ID プール ID を表します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForCognito",
    "Effect": "Allow",
    "Principal": {"Federated": "cognito-identity.amazonaws.com"},
  ]
}
```

```
"Action": "sts:AssumeRoleWithWebIdentity",
"Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east:12345678-ffff-ffff-ffff-123456"}}
}]
}
```

## OIDC 用のロールの作成

前提条件を満たしたら、IAM でロールを作成できます。次の手順は、AWS Management Console で OIDC フェデレーション用のロールを作成する方法を示します。AWS CLI または AWS API からロールを作成するには、「[サードパーティ ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」の手順を参照してください。

### Important

Amazon Cognito を使用する場合は、Amazon Cognito コンソールを使用してロールをセットアップします。それ以外の場合は、IAM コンソールを使用して OIDC 用のロールを作成します。

OIDC フェデレーション用の IAM ロールを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ロール] を選択した後、[ロールの作成] を選択します。
3. OIDC のロールタイプを選択します。
4. [Identity provider] (ID プロバイダー) で、ロールの IdP を選択します。
  - 個々のウェブ IdP 用のロールを作成する場合は、Login with Amazon、Facebook、Google のいずれかを選択します。

### Note

サポートする各 IdP に対して別々のロールを作成する必要があります。

- Amazon Cognito 用の高度なシナリオのロールを作成する場合は、Amazon Cognito を選択します。

**Note**

高度なシナリオで作業する場合に限り、Amazon Cognito で使用するロールを手動で作成する必要があります。それ以外の場合は、Amazon Cognito で自動的にロールを作成できます。Amazon Cognito の詳細については、[ Amazon Cognito Developer Guide ]の[\[Identity pools \(federated\) external identity providers\]](#)を参照してください。

- GitHub Actions 用のロールを作成する場合は、まず GitHub OIDC プロバイダーを IAM に追加する必要があります。GitHub OIDC プロバイダーを IAM に追加したら、`token.actions.githubusercontent.com` を選択します。

**Note**

AWS を GitHub の OIDC のフェデレーテッド ID として信頼するよう設定する方法については、「[GitHub Docs - Configuring OpenID Connect in Amazon Web Services](#)」(GitHub Docs - アマゾン ウェブ サービスの OpenID Connect 設定) を参照してください。IAM IdP for GitHub に関連するロールへのアクセスを制限するベストプラクティスについては、このページの [GitHub OIDC ID プロバイダーのロールの設定](#) を参照してください。

5. アプリケーション用の ID を入力します。ID のラベルは、選択するプロバイダーによって異なります。
  - Login with Amazon 用のロールを作成する場合は、アプリ ID を [Application ID] (アプリケーション ID) ボックスに入力します。
  - Facebook 用のロールを作成する場合は、アプリ ID を [Application ID] (アプリケーション ID) ボックスに入力します。
  - Google 用のロールを作成する場合は、対象者名を [Audience] (対象者) ボックスに入力します。
  - Amazon Cognito 用のロールを作成する場合は、Amazon Cognito アプリケーション用に作成した ID プールの ID を、[Identity Pool ID] (ID プール ID) ボックスに入力します。
  - GitHub Actions のロールを作成する場合は、以下の情報を入力します。
    - [対象者] で `[sts.amazonaws.com]` を選択します。
    - GitHub 組織の場合、GitHub の組織名を入力します。GitHub 組織名は必須で、英数字 (ダッシュ (-) を含む) でなければなりません。GitHub 組織名に、ワイルドカード文字 (\* と ?) は使用できません。

- (オプション) [GitHub リポジトリ] に、GitHub リポジトリの名前を入力します。値を指定しない場合は、デフォルトでワイルドカード (\*) になります。
  - (オプション) [GitHub ブランチ] に、GitHub ブランチ名を入力します。値を指定しない場合は、デフォルトでワイルドカード (\*) になります。
6. (オプション) [条件 (オプション)] で、[条件の追加] を選択して、ロールによって付与されたアクセス許可を使用する前にアプリケーションのユーザーが満たしておく必要がある追加条件を作成します。例えば、特定の IAM ユーザー ID にのみ AWS リソースに対するアクセス許可を付与する条件を追加できます。ロールを作成した後に、信頼ポリシーに条件を追加することもできます。詳細については、「[ロールの信頼ポリシーの変更 \(コンソール\)](#)」を参照してください。
  7. OIDC 情報を確認し、[次へ] を選択します。
  8. IAM には、あなたのアカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。アクセス許可ポリシーとして使用するポリシーを選択するか、[Create policy] (ポリシーの作成) を選択して新しいブラウザタブを開き、新しいポリシーをゼロから作成します。詳細については、「[IAM ポリシーの作成](#)」を参照してください。ポリシーを作成したら、そのタブを閉じて元のタブに戻ります。OIDC ユーザーに割り当てるアクセス許可ポリシーの横にあるチェックボックスをオンにします。必要に応じて、この時点でポリシーを選択せずに、後でポリシーをロールにアタッチすることもできます。デフォルトでは、ロールにはいずれのアクセス権限もありません。
  9. (オプション) [アクセス許可の境界](#)を設定します。これはアドバンスド機能です。

[Permissions boundary] (アクセス許可の境界) セクションを開き、[Use a permissions boundary to control the maximum role permissions] (アクセス許可の境界を使用してロールのアクセス許可の上限を設定する) を選択します。アクセス許可の境界として使用するポリシーを選択します。
  10. [Next] を選択します。
  11. [Role name] (ロール名) に、ロールの名前を入力します。ロール名は AWS アカウント アカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、**PRODRole** と **prodrole** というロール名を両方作成することはできません。他の AWS リソースがロールを参照している場合があるため、作成後はロールの名前を編集できません。
  12. (オプション) [Description] (説明) には、新しいロールの説明を入力します。
  13. [Step 1: Select trusted entities] (ステップ 1: 信頼済みエンティティの選択) または [Step 2: Add permissions] (ステップ 2: 権限の追加) のセクションで [Edit] (編集) を選択し、ロールのユースケースと権限を変更します。
  14. (オプション) ロールにメタデータを追加するには、キーバリューのペアとしてタグをアタッチします。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。

15. ロール情報を確認し、ロールの作成 を選択します。

### GitHub OIDC ID プロバイダーのロールの設定

GitHub を OpenID Connect (OIDC) ID プロバイダー (IdP) として使用する場合、ベストプラクティスは、IAM IdP に関連付けられたロールを引き受けることができるエンティティを制限することです。信頼ポリシーに条件ステートメントを含めると、ロールを特定の GitHub Organization、リポジトリ、またはブランチに制限できます。条件キー `token.actions.githubusercontent.com:sub` を文字列条件演算子と共に使用してアクセスを制限できます。条件を GitHub 組織内の特定のリポジトリまたはブランチのセットに制限することをお勧めします。AWS を GitHub の OIDC のフェデレーテッド ID として信頼するよう設定する方法については、「[GitHub Docs - Configuring OpenID Connect in Amazon Web Services](#)」(GitHub Docs - アマゾン ウェブ サービスの OpenID Connect 設定) を参照してください。

アクションワークフローまたは OIDC ポリシーで GitHub 環境を使用する場合は、セキュリティを強化するために保護ルールを環境に追加することを強くお勧めします。デプロイブランチとタグを使用して、環境にデプロイできるブランチとタグを制限します。保護ルールを使用して環境を設定する方法については、GitHub の記事「[Using environments for deployment](#)」(デプロイ用の環境の使用) に記載されている「[Deployment branches and tags](#)」(デプロイブランチとタグ) を参照してください。

GitHub の OIDC IdP がロールの信頼できるプリンシパルである場合、IAM はロールの信頼ポリシー条件をチェックして条件キー `token.actions.githubusercontent.com:sub` が存在していて、その値がワイルドカード文字 (\* と ?) または null だけではないことを確認します。IAM は、信頼ポリシーが作成または更新されたときにこのチェックを実行します。条件キー `token.actions.githubusercontent.com:sub` が存在しない場合、またはキー値が上記の値基準を満たさない場合、リクエストは失敗し、エラーが返されます。

#### Important

条件キー `token.actions.githubusercontent.com:sub` を特定の組織またはリポジトリに制限しない場合、管理外の組織またはリポジトリの GitHub Actions は、AWS アカウントで GitHub IAM IdP に関連付けられたロールを引き受けることができます。

以下の信頼ポリシー例は、定義済みの GitHub Organization、リポジトリ、ブランチへのアクセスを制限します。次の例では条件キー `token.actions.githubusercontent.com:sub` の値は、GitHub が文書化しているデフォルトのサブジェクト値の形式です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::012345678910:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub":
"repo:GitHubOrg/GitHubRepo:ref:refs/heads/GitHubBranch"
        }
      }
    }
  ]
}
```

次の例の条件は、定義された GitHub Organization とリポジトリへのアクセスを制限しますが、リポジトリ内の任意のブランチへのアクセスを許可します。

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:*"
  }
}
```

以下の条件例は、定義された GitHub Organization 内の任意のリポジトリまたはブランチへのアクセスを制限します。条件キー `token.actions.githubusercontent.com:sub` は、アクセスを GitHub 組織内からの GitHub Actions へのアクセスに制限する特定の値に制限することをお勧めします。

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
}
```

```
},  
"StringLike": {  
  "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/*"  
}  
}
```

ポリシーの条件チェックで利用可能な OIDC フェデレーションのキーの詳細については、「[OIDC AWS フェデレーションで使用できるキー](#)」を参照してください。

## SAML 2.0 フェデレーション用のロールを作成する (コンソール)

AWS アカウントで IAM ユーザーを作成する代わりに、SAML 2.0 フェデレーションを使用できます。ID プロバイダー (IdP) を使用すると、AWS の外部のユーザー ID を管理して、これらの外部ユーザー ID にアカウント内の AWS リソースに対するアクセス許可を付与できます。フェデレーションおよび認証プロバイダーについては、「[ID プロバイダーとフェデレーション](#)」を参照してください。

### Note

フェデレーションの耐障害性を高めるには、IdP と AWS フェデレーションを、複数の SAML サインインエンドポイントをサポートするように設定することをお勧めします。詳細については、AWS セキュリティブログの記事「[フェイルオーバーにリージョン SAML エンドポイントを使用する方法](#)」を参照してください。

## SAML 用のロールを作成するための前提条件

SAML 2.0 フェデレーション用のロールを作成する前に、まず次の基本的なステップを実行する必要があります。

### SAML 2.0 フェデレーション用のロールを作成する準備をするには

1. SAML ベースのフェデレーション用のロールを作成する前に、IAM で SAML プロバイダーを作成する必要があります。詳細については、「[IAM で SAML ID プロバイダーを作成する](#)」を参照してください。
2. SAML 2.0 認証ユーザーが引き受けるロールのポリシーを準備します。すべてのロールに該当することですが、SAML フェデレーション用のロールにも 2 つのポリシーが含まれています。1 つは、ロールの引き受け先を指定するロール信頼ポリシーです。もう 1 つは IAM アクセス許可ポリシーです。このポリシーでは、フェデレートッドユーザーにアクセスを許可または拒否する AWS アクションやリソースを指定します。



ロールの信頼ポリシーを作成する場合は、以下の 3 つの値を使用して当該アプリケーションにのみロールの引き受けを許可する必要があります。

- Action 要素で、sts:AssumeRoleWithSAML アクションを使用します。
- Principal 要素で、{"Federated":*ARNofIdentityProvider*} 文字列を使用します。*ARNofIdentityProvider* を、[Step 1](#) で作成した [SAML ID プロバイダー](#) の ARN と置き換えます。
- Condition 要素では StringEquals 条件を使用して、SAML レスポンスからの saml:aud 属性が AWS の SAML フェデレーションエンドポイントと一致することをテストします。

次の例では、SAML フェデレーテッドユーザー用に設計された信頼ポリシーを示します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRoleWithSAML",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/PROVIDER-NAME"},
    "Condition": {"StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}}
  }
}
```

プリンシパル ARN は、IAM で作成した SAML プロバイダー用の実際の ARN に置き換えます。これは、独自のアカウント ID およびプロバイダー名になります。

## SAML 用のロールの作成

前提条件のステップを完了すると、SAML ベースのフェデレーション用のロールを作成できます。

SAML ベースのフェデレーション用のロールを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. [SAML 2.0 フェデレーション] ロールタイプを選択します。

4. [Select a SAML provider] (SAML プロバイダーの選択) で、ロールのプロバイダーを選択します。
5. SAML 2.0 のアクセスレベルメソッドを選択します。
  - [プログラムによるアクセスのみを許可する] を選択して、AWS API または AWS CLI からプログラムで引き受けることができるロールを作成します。
  - [プログラムによるアクセスと AWS Management Console によるアクセスを許可する] を選択して、AWS Management Console からプログラムで引き受けることのできるロールを作成します。

作成されたロールはいずれも似ていますが、コンソールから引き受けられるロールにも、特定の条件を含む信頼ポリシーがあります。その条件では明示的に、SAML 対象者 (SAML:aud 属性) が SAML の AWS サインインエンドポイント (<https://signin.aws.amazon.com/saml>) に設定されます。

6. プログラムによるアクセス用のロールを作成する場合は、[属性] リストから属性を選択します。次に、[Value] (値) ボックスで、ロールに追加する値を入力します。これにより、ロールアクセスは、指定した属性を SAML 認証応答 (アサーション) に含んでいる ID プロバイダーのユーザーのみに制限されます。ロールが組織のユーザーのサブセットに限定されるように、少なくとも 1 つの属性を指定する必要があります。

プログラムによるアクセスとコンソールによるアクセス用のロールを作成する場合、SAML:aud 属性が自動的に追加され、AWS SAML エンドポイントの URL (<https://signin.aws.amazon.com/saml>) に設定されます。

7. 信頼ポリシーに属性関連の条件をさらに追加するには、[Condition](optional) (条件) (オプション) を選択し、続いて追加の条件を選択して、値を指定します。

#### Note

最も一般的に使用される SAML 属性がリストに表示されます。IAM では、条件の作成に使用できる追加の属性をサポートしています。サポートされる属性のリストについては、「[SAML ベースのフェデレーションに利用可能なキー](#)」を参照してください。リストに含まれないサポート対象の SAML 属性の条件が必要な場合、その条件は次のステップで手動で追加できます。これを行うには、ロールを作成後に信頼ポリシーを編集します。

8. SAML 2.0 の信頼情報を確認し、[Next] (次へ) を選択します。

9. IAM には、あなたのアカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。アクセス許可ポリシーとして使用するポリシーを選択するか、[Create policy] (ポリシーの作成) を選択して新しいブラウザタブを開き、新しいポリシーをゼロから作成します。詳細については、「[IAM ポリシーの作成](#)」を参照してください。ポリシーを作成したら、そのタブを閉じて元のタブに戻ります。OIDC フェデレーションユーザーに割り当てるアクセス許可ポリシーの横にあるチェックボックスをオンにします。必要に応じて、この時点でポリシーを選択せずに、後でポリシーをロールにアタッチすることもできます。デフォルトでは、ロールにはいずれのアクセス権限もありません。
10. (オプション) [アクセス許可の境界](#)を設定します。これはアドバンスド機能です。

[Permissions boundary] (アクセス許可の境界) セクションを開き、[Use a permissions boundary to control the maximum role permissions] (アクセス許可の境界を使用してロールのアクセス許可の上限を設定する) を選択します。アクセス許可の境界として使用するポリシーを選択します。
11. [Next] を選択します。
12. [次へ: レビュー] を選択します。
13. [Role name] (ロール名) に、ロールの名前を入力します。ロール名は AWS アカウント アカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、**PRODRROLE** と **prodrole** というロール名を両方作成することはできません。他の AWS リソースがロールを参照している場合があるため、作成後はロールの名前を変更できません。
14. (オプション) [Description] (説明) には、新しいロールの説明を入力します。
15. [Step 1: Select trusted entities] (ステップ 1: 信頼済みエンティティの選択) または [Step 2: Add permissions] (ステップ 2: 権限の追加) のセクションで [Edit] (編集) を選択し、ロールのユースケースと権限を変更します。
16. (オプション) タグをキーバリューのペアとしてアタッチして、メタデータをロールに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
17. ロール情報を確認し、ロールの作成 を選択します。

ロールの作成後に、AWS に関する情報を使用して ID プロバイダーソフトウェアを設定し、SAML 信頼を確立します。この情報には、フェデレーテッドユーザーで使用するロールが含まれます。これは、IdP と AWS との証明書利用者信頼の設定と呼ばれます。詳細については、「[証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する](#)」を参照してください。

## カスタム信頼ポリシーを使用したロールの作成 (コンソール)

カスタムの信頼ポリシーを作成して、アクセスを委任し、他のユーザーに自分の AWS アカウントでアクションの実行を許可することができます。詳細については、「[IAM ポリシーの作成](#)」を参照してください。

ロールを使用してアクセス権限を委任する方法の詳細については、「[ロールに関する用語と概念](#)」を参照してください。

### カスタム信頼ポリシーを使用した IAM ロールの作成 (コンソール)

IAM ユーザーが引き受けるロールは、AWS Management Console を使用して作成できます。例えば、組織で複数の AWS アカウント を使用して本稼働環境から開発環境を分離しているとします。開発用アカウントのユーザーに対して本番用アカウントのリソースへのアクセスを許可するロールを作成するための高レベルの情報については、「[個別の開発用アカウントと本稼働用アカウントを使用したシナリオ例](#)」を参照してください。

### カスタム信頼ポリシーを使用してロールを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. [Custom trust policy] (カスタム信頼ポリシー) ロールタイプを選択してください。
4. [Custom trust policy] (カスタム信頼ポリシー) セクションで、ロールのカスタム信頼ポリシーを入力または貼り付けます。詳細については、「[IAM ポリシーの作成](#)」を参照してください。
5. [ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。
6. 作成したカスタム信頼ポリシーの横にあるチェックボックスをオンにします。
7. (オプション) [アクセス許可の境界](#)を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。

[Permissions boundary] (アクセス許可の境界) セクションを開き、[Use a permissions boundary to control the maximum role permissions] (アクセス許可の境界を使用してロールのアクセス許可の上限を設定する) を選択します。IAM には、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。アクセス許可の境界として使用するポリシーを選択します。

8. [Next] (次へ) をクリックします。
9. [ロール名] で、ロール名のカスタマイズの度合いはサービスによって定義されます。サービスのロール名が定義されている場合、このオプションを変更することはできません。それ以外の場

合、サービスでロールのプレフィックスが定義され、オプションのサフィックスを入力できる場合があります。一部のサービスでは、ロールの名前全体を指定することができます。

可能な場合は、ロール名またはロール名のサフィックスを入力します。ロール名は AWS アカウント アカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、**PRODROLE** と **prodrole** というロール名を両方作成することはできません。他の AWS リソースがロールを参照している場合があるため、作成後はロールの名前を変更できません。

10. (オプション) [Description] (説明) には、新しいロールの説明を入力します。
11. [Step 1: Select trusted entities] (ステップ 1: 信頼済みエンティティの選択) または [Step 2: Add permissions] (ステップ 2: 権限の追加) セクションで [Edit] (編集) を選択し、ロールのカスタムポリシーと権限を編集します。
12. (オプション) タグをキーバリューのペアとしてアタッチして、メタデータをロールに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
13. ロール情報を確認し、[Create role (ロールの作成)] を選択します。

## アクセス権を委任するポリシーの例

以下の例では、AWS アカウント に別の AWS アカウント のリソースへのアクセスを許可する方法を示しています。これらの例の JSON ポリシードキュメントを使用して IAM ポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

### トピック

- [ロールを使用して他の AWS アカウント のリソースへのアクセス権を委任する](#)
- [ポリシーを使用してサービスへのアクセスを委任する](#)
- [リソースベースのポリシーを使用して他のアカウントの Amazon S3 バケットへのアクセス権を委任する](#)
- [リソースベースのポリシーを使用して他のアカウントの Amazon SQS キューへのアクセス権を委任する](#)
- [アカウントがアクセスを拒否されているとアクセス権を委任できない](#)

## ロールを使用して他の AWS アカウント のリソースへのアクセス権を委任する

IAM ロールを使用し、あるアカウントに属しているユーザーに、他のアカウントに属している AWS リソースへのアクセスを許可する方法のチュートリアルについては、「[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)」を参照してください。

### Important

特定のロールまたはユーザーの ARN を、ロールの信頼ポリシーの Principal 要素に含めることができます。ポリシーを保存すると、AWS は ARN を一意のプリンシパル ID に変換します。これにより、ロールまたはユーザーを削除して再作成することにより、誰かがそのユーザーの特権をエスカレートするリスクを緩和できます。通常、この ID はコンソールには表示されません。これは、信頼ポリシーが表示されるときに、ARN への逆変換が行われるためです。ただし、ロールまたはユーザーを削除すると、関係が壊れます。ポリシーは、ユーザーまたはロールを再作成しても適用されません。これは、信頼ポリシーに保存されているプリンシパル ID に一致しないためです。この場合、プリンシパル ID はコンソールに表示されます。これは、AWS が ARN に ID をマッピングできなくなるためです。その結果、信頼ポリシーの Principal 要素で指し示しているユーザーまたはロールを削除し、再作成する場合、ロールを編集して ARN を置き換える必要があります。ポリシーを保存するときに、ARN は新しいプリンシパル ID に変換されます。

## ポリシーを使用してサービスへのアクセスを委任する

次の例では、ロールにアタッチできるポリシーを示します。ポリシーは、Amazon EMR および AWS Data Pipeline の 2 つのサービスを有効にして、ロールを想定します。これらのサービスは、ロールに割り当てられた権限ポリシー (表示されていない) によって付与されたタスクを実行できます。複数のサービスプリンシパルを指定する場合に、2 つの Service 要素は指定できません。1 つのみ指定できます。代わりに、1 つの Service 要素の値として複数のサービスのプリンシパルのアレイを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "elasticmapreduce.amazonaws.com",
          "datapipeline.amazonaws.com"
        ]
      }
    }
  ]
}
```



```
    ]
  },
  "Action": "sts:AssumeRole"
}
]
```

リソースベースのポリシーを使用して他のアカウントの Amazon S3 バケットへのアクセス権を委任する

この例では、アカウント A はリソースベースのポリシー (Amazon S3 [バケットポリシー](#)) を利用して、アカウント A の S3 バケットへのフルアクセスをアカウント B に許可します。次に、アカウント B が IAM ユーザーポリシーを作成して、アカウント A のバケットへのアクセス権をアカウント B のいずれかのユーザーに委任します。

アカウント A の S3 バケットポリシーは以下のポリシーのようになります。この例では、アカウント A の S3 バケットの名前を mybucket とし、アカウント B のアカウント番号は 111122223333 とします。この番号は、アカウント B の個々のユーザーまたはグループではなく、アカウント自体のみを指定します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBAccess1",
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

また、アカウント A は Amazon S3 [アクセスコントロールリスト \(ACL\)](#) を使用して、アカウント B に S3 バケットまたはバケット内の 1 つのオブジェクトへのアクセスを許可することもできます。この場合、唯一変更する点は、アカウント A がアカウント B にアクセス権を付与する方法です。ただし、この例の 2 番目の部分で説明したように、アカウント B はポリシーを使用して、アカウント B の IAM グループにアクセス権を委任することになります。S3 バケットとオブジェクトのアクセス制御の詳細については、[Amazon Simple Storage Service ユーザーガイド](#)のアクセス制御を参照してください。



アカウント B の管理者が、次のサンプルポリシーを作成するとします。このポリシーでは、アカウント B のグループまたはユーザーに読み取りアクセスが許可されます。前のポリシーでは、アカウント B へのアクセスが許可されます。ただし、アカウント B の個々のグループとユーザーは、グループまたはユーザーポリシーでリソースへの明示的なアクセス許可が付与されるまで、リソースにアクセスできません。このポリシーのアクセス許可は、前のクロスアカウントポリシーのアクセス許可のサブセットとしてのみ付与できます。アカウント B はそのグループとユーザーに、最初のポリシーでアカウント A から付与されたものよりも多くのアクセス権限を付与することはできません。このポリシーでは、Action アクションのみを許可するように List 要素が明示的に定義されます。このポリシーの Resource 要素は、アカウント A が実装するバケットポリシーの Resource に一致します。

このポリシーを実装するには、アカウント B は IAM を使用して、アカウント B の該当するユーザー（またはグループ）にそのポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:List*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

リソースベースのポリシーを使用して他のアカウントの Amazon SQS キューへのアクセス権を委任する

以下の例では、アカウント A には Amazon SQS キューがあり、キューにアタッチされているリソースベースのポリシーを使用して、キューへのアクセスをアカウント B に許可します。その後で、アカウント B が IAM グループポリシーを使用して、アカウント B のグループにアクセス許可を委任します。

下記の例のキューポリシーでは、アカウント B にアカウント A の queue1 というキューで SendMessage および ReceiveMessage のアクションを実行するアクセス許可が付与されます。ただし、この許可が有効なのは、2014 年 11 月 30 日の正午から午後 3 時の間だけです。アカウント B のアカウント番号は 1111-2222-3333 です。アカウント A は、Amazon SQS を使用して、このポリシーを実装します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": ["arn:aws:sqs:*:123456789012:queue1"],
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
    }
  }
}
```

アカウント B のグループにアクセス権を委任するアカウント B のポリシーは、以下の例のようになります。アカウント B は、IAM を使用して、このポリシーをグループ（またはユーザー）にアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:queue1"
  }
}
```

上記の IAM ユーザーポリシーの例では、アカウント B はワイルドカードを使用して、属しているユーザーに、アカウント A のキューに対するすべての Amazon SQS アクションへのアクセスを許可しています。ただし、アカウント B は、アカウント B にアクセス許可が付与されている範囲においてのみ、アクセスを委任できます。2 つ目のポリシーを持つアカウント B グループは、2014 年 11 月 30 日の正午から午後 3 時の間だけキューにアクセスできます。ユーザーが実行できるのは、アカウント A の Amazon SQS キューポリシーで定義されている SendMessage および ReceiveMessage アクションのみです。

## アカウントがアクセスを拒否されているとアクセス権を委任できない

AWS アカウントは、自リソースへのアクセスをユーザーの親アカウントに対して明示的に拒否している場合、自リソースへのアクセス権をそれらのユーザーに委任することはできません。この拒否は、アクセス権を付与する既存のポリシーをユーザーが持っているかどうかにかかわらず、そのアカウントのユーザーにも反映されます。

たとえば、アカウント A は自身の S3 バケットについて、アカウント B からのアクセスを明示的に拒否するバケットポリシーを作成するとします。ただし、アカウント B は、アカウント B のユーザーにアカウント A のバケットへのアクセス権を付与する IAM ユーザーポリシーを作成しています。アカウント A の S3 バケットに適用される明示的な拒否は、アカウント B のユーザーにも反映され、アカウント B のユーザーにアクセス権を付与する IAM ユーザーポリシーよりも優先されます (アクセス許可の評価方法については、「[ポリシーの評価論理](#)」を参照してください)。

アカウント A のバケットポリシーは、以下のポリシーになります。この例では、アカウント A の S3 バケットの名前を mybucket とし、アカウント B のアカウント番号は 1111-2222-3333 とします。アカウント A は、Amazon S3 を使用して、このポリシーを実装します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBDeny",
    "Effect": "Deny",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::mybucket/*"
  }
}
```

この明示的な拒否は、アカウント A の S3 バケットにアクセスする権限を提供するアカウント B のポリシーをオーバーライドします。

## IAM ロールを使用する

作成したロールをユーザー、アプリケーション、またはサービスが使用できるようにするには、ロールを切り替えるアクセス許可を付与する必要があります。グループまたはユーザーにアタッチされたポリシーを使用して、必要なアクセス許可を付与することができます。このセクションでは、ロールを使用するアクセス許可をユーザーに付与する方法について説明します。また、ユーザーが、AWS Management Console、Tools for Windows PowerShell、AWS Command Line Interface (AWS CLI)、および [AssumeRole](#) API からロールに切り替える方法についても説明します。

**⚠ Important**

ロールの作成を IAM コンソールではなくプログラムで行う場合は、最大 64 文字までの Path に加えて最大 512 文字までの RoleName を追加できます。ただし、AWS Management Console の [ロールの切り替え] 機能でロールを使用する場合は、Path と RoleName の合計が 64 文字を超えることはできません。

ロールの切り替えは AWS Management Console から行うことができます。ロールを引き受けるには、AWS CLI または API オペレーションを呼び出すか、カスタム URL を使用します。使用方法によって、誰がロールを引き受けることができるか、およびロールセッションの持続期間が決定されます。AssumeRole\* API オペレーションを使用する場合、引き受ける IAM ロールはリソースです。AssumeRole\* API オペレーションを呼び出す ユーザーまたはロールはプリンシパルです。

## ロールを使用するためのメソッドの比較

ロールを引き受ける方法	だれがロールを引き受けるか	認証情報の有効期間を指定する方法	認証情報の有効期間 (最小   最大   デフォルト)
AWS Management Console	ユーザー ( <a href="#">ロールの切り替え</a> による)	最大セッション期間でロールの [概要] ページ	15 分   最大セッション期間設定 <sup>2</sup>   1 時間
<a href="#">assume-role</a> CLI または <a href="#">AssumeRole</a> API オペレーション	ユーザーまたはロール <sup>1</sup>	duration-seconds CLI または DurationSeconds API パラメータ	15 分   最大セッション期間設定 <sup>2</sup>   1 時間
<a href="#">assume-role-with-saml</a> CLI または <a href="#">AssumeRoleWithSAML</a> API オペレーション	SAML を使用して認証されたユーザー	duration-seconds CLI または DurationSeconds API パラメータ	15 分   最大セッション期間設定 <sup>2</sup>   1 時間

ロールを引き受ける方法	だれがロールを引き受けるか	認証情報の有効期間を指定する方法	認証情報の有効期間 (最小   最大   デフォルト)
<a href="#">assume-role-with-web-identity</a> CLI または <a href="#">AssumeRoleWithWebIdentity</a> API オペレーション	OIDC プロバイダーを使用して認証されたユーザー	duration-seconds CLI または DurationSeconds API パラメータ	15 分   最大セッション期間設定 <sup>2</sup>   1 時間
<a href="#">URLAssumeRole</a> を使用して構築されたコンソール	ユーザーまたはロール	URL の SessionDuration HTML パラメータ	15 分   12 時間   1 時間
<a href="#">URLAssumeRoleWithSAML</a> を使用して構築されたコンソール	SAML を使用して認証されたユーザー	URL の SessionDuration HTML パラメータ	15 分   12 時間   1 時間
<a href="#">URLAssumeRoleWithWebIdentity</a> を使用して構築されたコンソール	OIDC プロバイダーを使用して認証されたユーザー	URL の SessionDuration HTML パラメータ	15 分   12 時間   1 時間

<sup>1</sup> 1 つのロールの認証情報を使用して別のロールを引き受けることを [ロールの連鎖](#) と呼びます。ロールの連鎖を使用すると、新しい認証情報は最長期間である 1 時間に制限されます。ロールを使用して [EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与する](#) 場合、これらのアプリケーションにはこの制限が適用されません。

<sup>2</sup> この設定の値は 1 時間 ~ 12 時間です。最大セッション期間設定の修正の詳細については、「[ロールの修正](#)」を参照してください。この設定は、ロールの認証情報を取得したときにリクエストできる最大セッション期間設定を決定します。たとえば、[AssumeRole\\*](#) API オペレーションを使用してロールを引き受ける場合は、DurationSeconds パラメータの値を使用してセッションの期間を指定できます。このパラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。コンソールでロールを切り替える IAM ユーザーには、最大セッション期間、またはユーザーのセッションの残り時間のいずれかが短い方が付与されます。ロールに 5 時間の最大期間を設定することを想定しています。コンソールに 10 時間 (デフォルトの最大 12 時間) サインインした IAM ユーザーがロールに切り替わります。使用可能なロールセッション期間は 2 時間です。ロールの最大値を確認する方法については、このページで後述する「[ロールの最大セッション期間設定の表示](#)」を参照してください。

### 📌 メモ

- 最大セッション期間の設定では、AWS サービスが引き受けるセッションは制限されません。
- Amazon EC2 IAM ロールの認証情報は、ロールで設定された最大セッション期間の対象にはなりません。
- ロールセッション内でユーザーが現在のロールを再び引き受けることができるようにするには、ロール信頼ポリシーでロール ARN または AWS アカウント ARN をプリンシパルとして指定します。Amazon EC2、Amazon ECS、Amazon EKS、Lambda などのコンピューティングリソースを提供する AWS のサービスは、一時的な認証情報を提供し、これらの認証情報を自動的に更新します。これにより、常に有効な認証情報セットを確保できます。これらのサービスでは、一時的な認証情報を取得するために現在のロールを再度引き受ける必要はありません。ただし、[セッションタグ](#)または[セッションポリシー](#)を渡す場合は、現在のロールを再度引き受ける必要があります。ロールの信頼ポリシーを変更してプリンシパルロールの ARN または AWS アカウント ARN を追加する方法については、[ロールの信頼ポリシーの変更 \(コンソール\)](#) を参照してください。

### トピック

- [ロールの最大セッション期間設定の表示](#)
- [ロールを切り替えるアクセス許可をユーザーに付与する](#)
- [AWS のサービスにロールを渡すアクセス権限をユーザーに付与する](#)
- [ロールの切り替え \(コンソール\)](#)

- [IAM ロール \(AWS CLI\) の切り替え](#)
- [IAM ロールへの切り替え \(Tools for Windows PowerShell\)](#)
- [IAM ロール \(AWS API\) の切り替え](#)
- [Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)
- [IAM ロールの一時的なセキュリティ認証情報の取り消し](#)

## ロールの最大セッション期間設定の表示

AWS Management Console を使用するか、AWS CLI または AWS API を使用して、ロールの最大セッション期間を指定できます。AWS CLI または API オペレーションを使用してロールを引き受ける場合は、DurationSeconds パラメータの値を指定できます。このパラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。パラメータを指定する前に、ロールのこの設定を表示する必要があります。DurationSeconds パラメータに最大値設定よりも大きい値を指定した場合は、オペレーションが失敗します。

ロールの最大セッション期間を表示するには (コンソール)

1. IAM コンソールのナビゲーションペインで [ロール] を選択します。
2. 表示するロールの名前を選択します。
3. [最大セッション期間] の横に、ロールに付与されている最大セッション長を表示します。これは、AWS CLI または API オペレーションで指定できる最大セッション期間です。

ロールの最大セッション継続時間設定を表示するには (AWS CLI)

1. 引き受けるロールの名前がわからない場合は、次のコマンドを実行してアカウントのロールを一覧表示します。
  - [aws iam list-roles](#)
2. ロールの最大セッション継続時間を表示するには、次のコマンドを実行します。次に、最大セッション継続時間パラメータを確認します。
  - [aws iam get-role](#)



## ロールの最大セッション継続時間設定を表示するには (AWS API)

1. 引き受けるロールの名前がわからない場合は、次のオペレーションを呼び出してアカウントのロールを一覧表示します。
  - [ListRoles](#)
2. ロールの最大セッション継続時間を表示するには、次のオペレーションを実行します。次に、最大セッション継続時間パラメータを確認します。
  - [GetRole](#)

## ロールを切り替えるアクセス許可をユーザーに付与する

管理者が[クロスアカウントアクセス用のロールを作成する](#)場合、ロールを所有するアカウント、リソース (信頼するアカウント)、およびユーザーを含むアカウント (信頼されるアカウント) の間で信頼を確立します。これを行うには、信頼するアカウントの管理者が、ロールの信頼ポリシーで信頼できるアカウント番号を Principal として指定します。これにより、信頼されたアカウント内のすべてのユーザーがロールを引き受けることができるようになる可能性があります。設定を完了するには、信頼されたアカウントの管理者がそのアカウント内の特定のグループまたはユーザーにロールを切り替えるアクセス権限を付与する必要があります。

### ロールを切り替えるアクセス許可を付与するには

1. 信頼されたアカウントの管理者として、ユーザーの新しいポリシーを作成するか、既存のポリシーを編集して必要な要素を追加します。詳細については、「[ポリシーの作成または編集](#)」を参照してください。
2. 次に、ロール情報を共有する方法を次の中から選択します。
  - **ロールリンク:** 詳細がすべて既に入力されている [Switch Role] (ロールの切り替え) ページへのリンクをユーザーに送信します。
  - **アカウント ID またはエイリアス:** ロール名とアカウント ID 番号またはアカウントのエイリアスを各ユーザーに提供します。これにより、ユーザーは [ロールの切り替え] ページに移動し、詳細を手動で追加できます。

詳細については、「[ユーザーへの情報の提供](#)」を参照してください。

ロールを切り替えできるのは、IAM ユーザー、SAML フェデレーションロール、またはウェブ ID フェデレーションロールとしてサインインしている場合のみです。AWS アカウントのルートユーザーとしてサインインすると、ロールを切り替えることはできません。

### Important

AWS Management Console で、[ExternalId](#) 値を必要とするロールに切り替えることはできません。ExternalId パラメータをサポートする [AssumeRole](#) API を呼び出すことにより、このようなロールに切り替えることができます。

### メモ

- タスクを達成するために最終的にユーザーにアクセス許可を付与するので、このトピックでは、ユーザーのポリシーについて説明します。ただし、個々のユーザーに直接アクセス許可を付与することはお勧めしません。ユーザーがロールを引き受けると、そのロールに関連付けられたアクセス許可が割り当てられます。
- AWS Management Console でロールを切り替えると、コンソールは常に元の認証情報を使用して切り替えを認証します。これは、IAM ユーザー、SAML フェデレーションロール、またはウェブ ID フェデレーションロールとしてサインインする際に適用されます。例えば、RoleA に切り替える場合は、IAM では元のユーザーまたはフェデレーションロールの認証情報を使用して、RoleA の引き受けが許可されているかどうか判断されます。その後、RoleA を使用中に RoleB への切り替えを試みると、元のユーザーまたはフェデレーションロールの認証情報を使用して、RoleB への切り替えが承認されます。RoleA の資格情報は、このアクションには使用されません。

## トピック

- [ポリシーの作成または編集](#)
- [ユーザーへの情報の提供](#)


## ポリシーの作成または編集

ロールを引き受けるアクセス許可をユーザーに付与するポリシーには、次の Allow 効果を伴うステートメントを含める必要があります。

- sts:AssumeRole アクション

- Resource 要素でのロールの Amazon リソースネーム (ARN)


このポリシーを取得するユーザーは、(グループメンバーとして、または直接アタッチされて) 一覧表示されたリソース上のロールに切り替えることができます。

 Note

Resource が \* に設定されている場合、ユーザーは、ユーザーのアカウントを信頼する任意のアカウントで任意のロールを引き受けることができます。(つまり、ロールの信頼ポリシーは、ユーザーのアカウントを Principal として指定します)。ベストプラクティスとして、[最小権限の原則](#)に従って、ユーザーが必要とするロールに限って完全な ARN を指定することをお勧めします。

次の例に示すポリシーでは、ユーザーは 1 つのアカウントに限ってロールを引き受けることができます。さらに、このポリシーではワイルドカード (\*) を使用し、ロール名が文字列 Test で始まる場合に限り、ユーザーがロールに切り替えることができることを指定します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account-id:role/Test*"
  }
}
```

 Note

ロールをユーザーに付与するアクセス権限は、既にユーザーに付与されているアクセス権限に追加されるわけではありません。ユーザーがロールに切り替えると、そのユーザーはロールによって付与されたアクセス権限と引き換えに元々付与されていたアクセス権限を一時的に喪失します。ユーザーがロールを終了すると、元のユーザーアクセス権限が自動的に復元します。例えば、ユーザーのアクセス許可では Amazon EC2 インスタンスの操作が許可されるが、これらのアクセス許可はロールのアクセス許可ポリシーで付与されないとします。この場合、ロールの使用中に、ユーザーはコンソールで Amazon EC2 インスタンスを操作で

きません。さらに、AssumeRole を介して取得された一時的な認証情報は、プログラムでは Amazon EC2 インスタンスで機能しません。

## ユーザーへの情報の提供

ロールを作成し、このロールに切り替えるアクセス許可をユーザーに付与した後で、ユーザーに以下を提供する必要があります。

- ロールの名前は
- ID またはロールが含まれているアカウントエイリアス

アカウント ID とロール名が事前設定されたリンクを送信すると、ユーザーのアクセスが簡素化されます。ロールのリンクは、[ロールの作成] ウィザードの完了後に [ロールを表示] バナーを選択すると表示されます。また、クロスアカウントが有効なロールの場合は [ロールの概要] ページに表示されます。

または、次の形式を使用して手動でリンクを作成することもできます。次の例の 2 つのパラメータのアカウント ID またはエイリアスとロール名を置き換えます。

```
https://signin.aws.amazon.com/switchrole?  
account=your_account_ID_or_alias&roleName=optional_path/role_name
```

ユーザーにトピック「[ロールの切り替え \(コンソール\)](#)」を参照してプロセスに従って進めてもらうことをお勧めします。ロールを引き受ける際に遭遇する可能性がある一般的な問題のトラブルシューティングを行うには、「[ロールを引き受けることができない](#)」を参照してください。

## 考慮事項

- プログラムでロールを作成する場合は、パスと名前を持ったロールを作成できます。、ユーザーが AWS Management Console の[ロールの切り替え]ページで入力できるように、完全なパスとロール名をユーザーに提供する必要があります。例: `division_abc/subdivision_efg/role_XYZ`。
- プログラムでロールを作成する場合は、512 文字までの Path と RoleName を追加できます。名前の長さは最大 64 文字です。ただし、AWS Management Console の [ロールの切り替え] 機能でロールを使用するには、Path と RoleName の合計が 64 文字を超えることはできません。

- セキュリティ上の理由から、[AWS CloudTrail ログを確認](#)して、AWS でアクションを実行したユーザーを調べることができます。ロール信頼ポリシーで `sts:SourceIdentity` 条件キーを使用すると、ユーザーがロールを引き受けるときに ID を指定するように要求できます。例えば、IAM ユーザーがセッション名として自分のユーザー名を指定するように要求できます。これにより、AWS の特定のアクションを実行したユーザーを特定できます。詳細については、「[sts:SourceIdentity](#)」を参照してください。[sts:RoleSessionName](#) 条件キーを使用すると、ユーザーがロールを引き受けるときにセッション名を指定するように要求できます。これは、ロールが異なるプリンシパルによって使用される場合に、ロールセッションを区別するのに役立ちます。

## AWS のサービスにロールを渡すアクセス権限をユーザーに付与する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これで、その後サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。大半のサービスでは、サービスにロールを渡さなければならないのはセットアップ時の 1 回のみで、サービスがロールを引き受けるときには行いません。例えば、Amazon EC2 インスタンスで実行しているアプリケーションがあるとします。このアプリケーションには認証に使う一時的な認証情報と、AWS でアクションを実行するためのアプリケーション認証のアクセス許可が必要です。アプリケーションをセットアップする場合、こうした認証情報を提供するインスタンスで使用するよう、Amazon EC2 にロールを渡す必要があります。そのロールに IAM ポリシーをアタッチして、インスタンスで実行するアプリケーションのアクセス許可を定義します。アプリケーションは、このロールが許可しているアクションを実行する必要があるたびに、ロールを引き受けます。

ユーザーがロール (とそのアクセス許可) を AWS サービスに渡すには、そのサービスにロールを渡すアクセス許可が必要になります。これにより、承認済みのユーザーのみが、アクセス許可を付与するロールを使用してサービスを設定できるようになります。ユーザーが AWS サービスにロールを渡すには、IAM ユーザー、ロール、またはグループに `PassRole` アクセス許可を付与する必要があります。

### Warning

- 同じ AWS アカウントを共有するサービスに IAM ロールを渡すには、`PassRole` のアクセス許可のみ使用できます。アカウント A のロールをアカウント B のサービスに渡すには、まずアカウント A からロールを引き受けることができる IAM ロールをアカウント B に作成する必要があります。その後、アカウント B のロールをサービスに渡すことができます。詳細については、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

- ロールをタグ付けした後に、iam:PassRole アクションでポリシー内の ResourceTag 条件キーを使用してロールを渡せるユーザーを制御しないようにしてください。このアプローチでは信頼できる結果は得られません。

PassRole アクセス許可を設定する場合、ユーザーがロールに必要な以上のアクセス許可があるロールを渡さないようにする必要があります。例えば、Alice が Amazon S3 アクションを実行する許可を持っていない場合があります。Alice が Amazon S3 アクションを許可するサービスにロールを渡すことができる場合、サービスはジョブの実行時に、Alice に代わって Amazon S3 アクションを実行できます。

サービスにリンクされたロールを特定する場合は、サービスにそのロールを渡すためのアクセス許可も必要になります。一部のサービスでは、そのサービスでアクションを実行する際にアカウント内にサービスにリンクされたロールが自動的に作成されます。例えば、Amazon EC2 Auto Scaling では Auto Scaling グループが最初に作成されたときに、AWSServiceRoleForAutoScaling のサービスにリンクされたロールを作成します。のアクセス許可がない状態で Auto Scaling グループの作成時にサービスにリンクされたのアクセス許可がない状態で、のアクセス許可がない状態で、iam:PassRole のアクセス許可がない状態で、エラーが表示されます。ロールを明示的に指定しない場合、iam:PassRole のアクセス許可は不要で、デフォルトではそのグループで実行されるすべての操作に AWSServiceRoleForAutoScaling ロールを使用します。サービスにリンクされたロールをサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。サービスにリンクされたロールがサービスでアクションの実行時に自動的に作成されるかどうかを確認するには、「はい」リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを参照してください。

ユーザーは、ロールを使用してサービスにアクセス許可を割り当てる任意の API オペレーションで、パラメータとして ARN を渡すことができます。次に、そのサービスはユーザーに iam:PassRole 権限があるかどうか確認します。ユーザーが渡せるロールを承認済みのロールだけに制限するには iam:PassRole のアクセス許可と IAM のポリシーステートメントの Resources 要素をフィルターに掛けることができます。

JSON ポリシーの Condition 要素を使用して、すべての AWS リクエストのリクエストコンテキストに含まれるキーの値をテストできます。ポリシーでの条件キーの使用の詳細については、「[IAM JSON ポリシー要素Condition](#)」をご参照ください。iam:PassedToService 条件キーを使用して、ロールを渡すことができるサービスのサービスプリンシパルを指定できます。ポリシーでの iam:PassedToService 条件キーの使用の詳細については、「[iam:PassedToService](#)」をご参照ください。

## 例 1

インスタンスの起動時に、承認済みの一連のロールを Amazon EC2 サービスに渡すための権限をユーザーに付与したいと考えているとします。その場合、3つの要素が必要です。

- ロールに許可されている内容を定義し、ロールにアタッチしている IAM のアクセス許可ポリシーロールが実行しなければならない操作やロールが操作を実行する上で必要とするリソースのみにアクセス許可を制限します。AWS のマネージドまたはカスタマー定義の IAM アクセス権ポリシーが使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [ "A list of the permissions the role is allowed to use" ],
    "Resource": [ "A list of the resources the role is allowed to access" ]
  }
}
```

- サービスにロールの引き受けを許可する、ロールの信頼ポリシー。例えば、次の信頼ポリシーを UpdateAssumeRolePolicy アクションを使用するロールにアタッチできます。この信頼ポリシーは Amazon EC2 がロールを使用し、そのロールにアタッチしているアクセス許可の使用を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
    "Effect": "Allow",
    "Principal": { "Service": "ec2.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

- 承認済みのロールのみをユーザーが渡せるように許可する、IAM ユーザーにアタッチしている IAM のアクセス許可ポリシー。ユーザーが渡すロールの詳細を得るため、通常 iam:GetRole を iam:PassRole に追加します。この例でユーザーが渡すことができるのは、指定されたアカウントに存在し、EC2-roles-for-XYZ- で始まる名前を持つロールに限ります。

```
{
  "Version": "2012-10-17",
```



```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "iam:GetRole",
    "iam:PassRole"
  ],
  "Resource": "arn:aws:iam::account-id:role/EC2-roles-for-XYZ-*"
}]
}
```

これでユーザーは割り当てられたロールで Amazon EC2 インスタンスを起動することができます。インスタンスで実行しているアプリケーションはインスタンスプロファイルのメタデータのロールで一時的な認証情報にアクセスすることができます。ロールにアタッチされたアクセス権限ポリシーは、インスタンスに許可する操作を定義します。

## 例 2

Amazon Relational Database Service (Amazon RDS) は、拡張モニタリングと呼ばれる機能をサポートしています。この機能により、Amazon RDS はエージェントを使用してデータベースインスタンスをモニタリングできます。また、Amazon RDS は Amazon CloudWatch Logs にメトリクスを記録することもできます。この機能を有効にするには、サービスロールを作成して、メトリクスをモニタリングしログに書き込む権限を Amazon RDS に付与する必要があります。

Amazon RDS 拡張モニタリング用のロールを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [ロール]、[ロールの作成] の順に選択します。
3. [AWS Service] ロールタイプを選択し、[Use cases for other AWS のサービス] で [RDS] サービスを選択します。[RDS – Enhanced Monitoring] (RDS – 拡張モニタリング)、[Next] (次へ) の順に選択します。
4. AmazonRDSEnhancedMonitoringRole アクセス権ポリシーを選択します。
5. [Next] を選択します。
6. [Role name] (ロール名) に、このロールの目的を識別しやすくするロール名を入力します。ロール名は AWS アカウント アカウント内で一意である必要があります。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。サインイン処理中など、コンソールでロール名がユーザーに表示される場合、ロール名は大文字と小文字を

区別しません。さまざまなエンティティがロールを参照する可能性があるため、作成後にロール名を編集することはできません。

7. (オプション) [Description (説明)] には、新しいロールの説明を入力します。
8. (オプション) タグをキーバリューペアとしてアタッチして、メタデータをユーザーに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
9. ロール情報を確認し、ロールの作成 を選択します。

ロールを引き受ける `monitoring.rds.amazonaws.com` サービスのアクセス許可を付与する信頼ポリシーをロールが自動的に取得します。その後、Amazon RDS は `AmazonRDSEnhancedMonitoringRole` ポリシーが許可するすべての操作を実行できるようになります。

拡張モニタリングへのアクセスを許可するユーザーは、次に示すように、ユーザーが RDS をリストすることを許可するステートメントとユーザーがロールを渡すことを許可するステートメントを含むポリシーが必要になります。アカウント番号を自分のものに置き換え、ロールの名前をステップ 6 で指定した名前に置き換えます。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
},
{
  "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
  "Effect": "Allow",
  "Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account-id:role/RDS-Monitoring-Role"
}
```

別のポリシーのステートメントとこのステートメントを組み合わせたり、独自のポリシーで使用することができます。代わりにユーザーが RDS- で始まるロールを渡せるように指定するには、以下のようリソース ARN にあるロールの名前をワイルドカードに置き換えできます。

```
"Resource": "arn:aws:iam::account-id:role/RDS-*
```

## AWS CloudTrail ログ内の `iam:PassRole` アクション

PassRole は API 呼び出しではありません。PassRole はアクセス許可です。つまり、IAM PassRole には CloudTrail ログは生成されません。CloudTrail でどのロールがどの AWS のサービスに渡されるかを確認するには、ロールを受け取る AWS のリソースを作成または変更した CloudTrail ログを確認する必要があります。例えば、ロールが作成されると AWS Lambda 関数に渡されます。CreateFunction アクションのログには、関数に渡されたロールの記録が表示されます。

## ロールの切り替え (コンソール)

ロールは、必要な AWS リソースへのアクセスに使用できる一連のアクセス許可を指定します。その点では、[AWS Identity and Access Management \( IAM \) のユーザー](#)に似ています。ユーザーとしてサインインすると、特定の一連のアクセス許可が付与されます。ただし、ロールにはサインインされませんが、一度サインインするとロールを切り替えることもできます。こうすると、元のユーザーアクセス権限が一時的に無効になり、そのロールに割り当てられたアクセス権限が代わりに付与されます。ロールは、自身のアカウントのロールでも、他の AWS アカウント のロールでもかまいません。ロール、その利点、作成方法の詳細については、「[IAM ロール](#)」と「[IAM ロールの作成](#)」を参照してください。

### Important

ユーザーと切り替え後のロールのアクセス許可は、累積されません。同時に有効になるアクセス権限のセットは 1 つのみです。ロールを切り替えると、ユーザーアクセス権限が一時的に無効になり、切り替え後のロールに割り当てられたアクセス権限が有効になります。そのロールを終了すると、ユーザーアクセス権限が自動的に復元されます。

AWS Management Console でロールを切り替えると、コンソールは常に元の認証情報を使用して切り替えを認証します。これは、IAM ユーザー、IAM Identity Center でのユーザー、SAML フェデレーションロール、またはウェブ ID フェデレーションロールとしてサインインする際に適用されます。たとえば、RoleA に切り替える場合は、IAM は元のユーザーまたはフェデレーションロールの認証情報を使用して、RoleA の引き受けが許可されているかどうかを判断します。その後、RoleA を使用中に RoleB への切り替えを試みると、AWS は引き続き RoleA の認証情報ではなく元のユーザーまたはフェデレーテッドロールの認証情報を使用して切り替えを承認します。

## コンソールでのロールの切り替えについて知っておくべきこと

このセクションでは、IAM コンソールを使用してロールを切り替えるための追加情報を提供します。

**i** 注記 :

- AWS アカウントのルートユーザーとしてサインインすると、ロールを切り替えることはできません。IAM ユーザー、IAM Identity Center でのユーザー、SAML フェデレーションロール、またはウェブ ID フェデレーションロールとしてサインインするときにロールを切り替えられます。
- AWS Management Console で、[ExternalId](#) 値を必要とするロールに切り替えることはできません。ExternalId パラメータをサポートする [AssumeRole](#) API を呼び出すことにより、このようなロールに切り替えることができます。

- 管理者によりリンクが提供されている場合、リンクを選択して、以下の手順のステップ「[Step 5](#)」に進んでください。リンクをクリックすると該当するウェブページに移動し、アカウント ID (またはエイリアス) とロール名が自動的に入力されます。
- 手動でリンクを作成して、次の手順のステップ [Step 5](#) にスキップできます。リンクを作成するには、次の形式を使用します。

```
https://signin.aws.amazon.com/switchrole?  
account=account_id_number&roleName=role_name&displayName=text_to_display
```

ここで、次のテキストに置き換えます。

- *account\_id\_number* – 12 桁のアカウント ID は、管理者から提供されます。または、管理者がアカウントエイリアスを作成して、URL にアカウント ID の代わりにアカウント名を含めることもできます。詳細については、AWS サインイン ユーザーガイドの「[ユーザータイプ](#)」を参照してください。
- *role\_name* – 引き受けるロールの名前。これはロールの ARN の末尾から取得できます。たとえば、ロールの ARN が `arn:aws:iam::123456789012:role/TestRole` である場合、ロール名は `TestRole` になります。
- (オプション) *text\_to\_display* – このロールがアクティブなときにユーザー名の代わりにナビゲーションバーに表示するテキストを入力できます。
- 次の手順を使用して、管理者が提供する情報を使用して手動でロールを切り替えることができます。

デフォルトでは、ロールを切り替えると、AWS Management Console セッションは 1 時間続きます。IAM ユーザーセッションは、デフォルトで 12 時間です。コンソールでロールを切り替える IAM

ユーザーには、ロールに設定された最大セッション期間、またはユーザーのセッションの残り時間のいずれか短い方が付与されます。たとえば、ロールに対して最大セッション期間が 10 時間に設定されているとします。IAM ユーザーがロールに切り替えることを決定すると、8 時間コンソールにサインインしています。ユーザーセッションには残り時間が 4 時間あるため、許可されるロールセッション期間は 4 時間です。次の表は、コンソールでロールを切り替えるときの IAM ユーザーのセッション期間を決定する方法を示しています。

#### IAM ユーザーコンソールロールセッション期間

IAM ユーザーセッションの残り時間は..。	ロールセッションの期間は..。		
ロールの最大セッション時間より小さい	ユーザーセッションでの残り時間		
ロールの最大セッション時間の長さを超えています	最大セッション期間値		
ロールの最大セッション時間と同等	最大セッション継続時間値 (概算)		

#### Note

ある程度 AWS サービスコンソールは、ロールセッションの有効期限が切れたときに、アクションを実行せずにロールセッションを自動更新できます。セッションを再認証するために、ブラウザページをリロードするように求めるメッセージが表示されることがあります。

ロールを引き受ける際に遭遇する可能性がある一般的な問題のトラブルシューティングを行うには、[「ロールを引き受けることができない」](#)を参照してください。

## ロールを切り替えるには (コンソール)

1. IAM として AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールで、右上のナビゲーションバーのユーザー名を選択します。通常は **username@account\_ID\_number\_or\_alias** のように表示されます。
3. [Switch Role] (ロールの切り替え) を選択します。このオプションを選択するのが初めての場合、詳細な情報がページに表示されます。情報を読んだ後、[ロールの切り替え] を選択します。ブラウザの Cookie をオフにすると、このページが再び表示されることがあります。
4. [ロールの切り替え] ページで、アカウント ID 番号またはアカウントエイリアスと、管理者により提供されたロールの名前を入力します。

### Note

管理者が `division_abc/subdivision_efg/roleToDoX` などのパスを使用してロールを作成した場合は、[ロール] ボックスに完全なパスと名前を入力する必要があります。ロール名のみを入力した場合、または Path と RoleName の組み合わせの合計が 64 文字を超える場合は、ロールの切り替えは失敗します。これは、ロール名を保存するブラウザクッキーの制約です。この場合は、管理者に連絡して、パスとロール名のサイズを減らすことを依頼してください。

5. (オプション) 表示名を選択します。このロールがアクティブな場合にユーザー名の代わりにナビゲーションバーに表示するテキストを入力します。名前の候補がアカウントとロールの情報に基づいて表示されますが、わかりやすい名前に変更できます。表示名の強調表示に使用される色を選択することもできます。名前と色から、ロールがアクティブになってアクセス権限が変わることがわかりやすくなります。たとえば、テスト環境にアクセスできるロールには、[表示名] に「**Test**」と入力し、[色] で緑を選択できます。本稼働環境にアクセスできるロールには、[表示名] に「**Production**」と入力し、[色] で赤を選択できます。
6. [Switch Role] (ロールの切り替え) を選択します。表示名と色によってナビゲーションバーのユーザー名が置き換えられ、そのロールにより付与されたアクセス許可を使用し始めることができます。

### ヒント

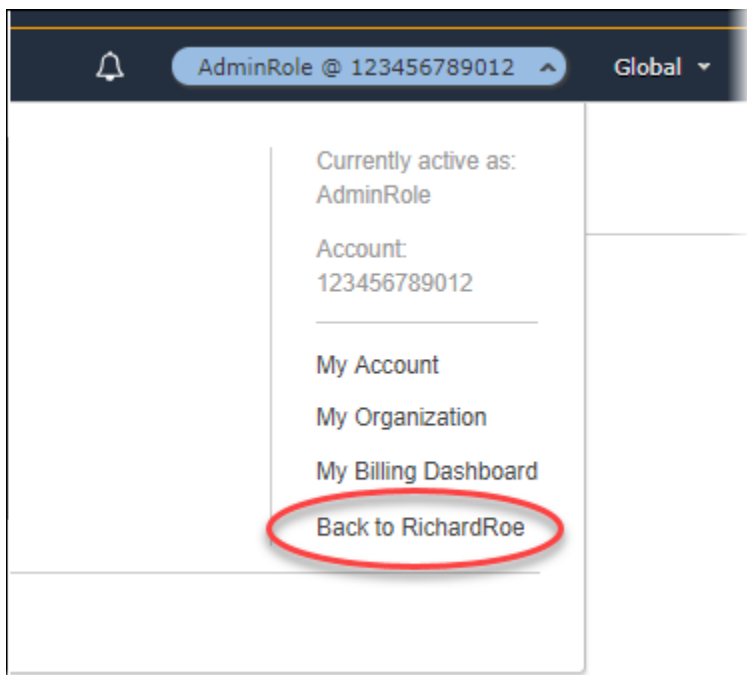
最近使用したいいくつかのロールがメニューに表示されます。次回いずれかのロールに切り替える必要があるときは、そのロールを選択するだけで切り替えることができます。ロールが

メニューに表示されていない場合は、アカウントとロールの情報を入力するだけでかまいません。

ロールの使用を停止するには (コンソール)

1. IAM コンソールで、右上のナビゲーションバーでロールの [表示名] を選択します。通常、表示名は「**rolename@account\_ID\_number\_or\_alias**」のように表示されます。
2. [#####に戻る] を選択します。ロールとそのアクセス権限が無効になり、IAM ユーザーおよびグループに関連付けられたアクセス権限が自動的に復元されます。

たとえば、アカウント番号 123456789012 にユーザー名 RichardRoe を使用してサインインしているとします。AdminRole ロールの使用後に、ロールの使用を停止して元のアクセス許可に戻ります。ロールの使用を停止するには、[AdminRole @ 123456789012]、[RichardRoe に戻る] の順に選択します。



## IAM ロール (AWS CLI) の切り替え

ロールは、必要な AWS リソースへのアクセスに使用できる一連のアクセス許可を指定します。その点では、[AWS Identity and Access Management \( IAM \) のユーザー](#)に似ています。ユーザーとしてサインインすると、特定の一連のアクセス許可が付与されます。ただし、ロールにはサインインされませんが、ユーザーとしてサインインした後でロールを切り替えることができます。こうすると、元



のユーザーアクセス権限が一時的に無効になり、そのロールに割り当てられたアクセス権限が代わりに付与されます。ロールは、自身のアカウントのロールでも、他の AWS アカウント のロールでもかまいません。ロールとその利点、およびロールを作成して設定する方法については、「[IAM ロール](#)」および「[IAM ロールの作成](#)」を参照してください。ロールを引き受ける別の方法については、「[IAM ロールを使用する](#)」を参照してください。

#### Important

IAM ユーザーのアクセス許可および引き受けるロールは、累積されません。同時に有効になるアクセス権限のセットは 1 つのみです。ロールを引き受けると、以前のユーザーまたはロールのアクセス許可が一時的に無効になり、切り替え後のロールに割り当てられたアクセス許可が有効になります。そのロールを終了すると、ユーザーアクセス権限が自動的に復元されます。

IAM ユーザーとしてサインインしている場合、ロールを使用して AWS CLI コマンドを実行できます。また、[外部で認証されたユーザー \(SAML または OIDC\)](#) としてサインインしている場合にも、ロールを使用して AWS CLI コマンドを実行できます。また、インスタンスプロファイルを経由して、ロールにアタッチされた Amazon EC2 インスタンス内部から AWS CLI コマンドを実行するロールを使用できます。AWS アカウントのルートユーザーとしてサインインしているときに、ロールを引き受けることはできません。

**ロールの連鎖** — ロールの連鎖を使用することもできます。この連鎖は、ロールのアクセス許可を使用して 2 つ目のロールにアクセス許可を使用します。

デフォルトでは、ロールセッションは 1 時間です。assume-role\* CLI オペレーションを使用してこのロールを引き受ける場合は、duration-seconds パラメータの値を指定できます。この値は 900 秒 (15 分) からロールの最大セッション期間設定までの範囲を指定できます。コンソールでロールを変更した場合、セッションの所要時間は最大 1 時間に制限されます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。

ロールの連鎖を使用すると、セッション期間は最長である 1 時間に制限されます。この場合 duration-seconds パラメータを使用して 1 時間より大きい値を指定すると、オペレーションは失敗します。

シナリオ例: 本稼働ロールに切り替える

開発環境で作業している IAM ユーザーであるとしてします。このシナリオでは、[AWS CLI](#) でコマンドラインを使用して実稼働環境を操作する必要が生じることがあります。1 つのアクセスキー認証情報の

セットがすでに使用可能です。このセットは、標準の IAM ユーザーに割り当てられたアクセスキーペアである場合があります。または、フェデレーテッドユーザーとしてサインインしている場合は、最初に割り当てられたロールのアクセスキーペアである場合があります。現在のアクセス許可で特定の IAM ロールを引き受けることができるなら、AWS CLI 設定ファイルの「プロファイル」でそのロールを特定できます。このコマンドは、元のアイデンティティではなく、指定された IAM ロールのアクセス権限を使用して実行されます。このプロファイルを AWS CLI コマンドで指定すると、新しいロールを使用することになります。この場合、開発用アカウントの元のアクセス許可を同時に使用することはできません。同時に有効にできるアクセス許可のセットは 1 つのみであるためです。

### Note

セキュリティ上の理由から、管理者は [AWS CloudTrail ログを確認](#)して、AWS でアクションを実行したユーザーを調べることができます。管理者は、ロールを引き受けるときに、ソース ID またはロールセッション名の指定を要求する場合があります。詳細については、[sts:SourceIdentity](#) および [sts:RoleSessionName](#) を参照してください。

本稼働ロールに切り替えるには (AWS CLI)

1. AWS CLI をはじめて使用する場合は、まず、デフォルトの CLI プロファイルを設定する必要があります。コマンドプロンプトを開き、IAM ユーザーまたはフェデレーテッドロールからのアクセスキーを使用するように、AWS CLI を設定します。詳細については、『[AWS Command Line Interface ユーザーガイド](#)』の「AWS Command Line Interface の設定。」を参照してください。

以下のように、[aws configure](#) コマンドを実行します。

```
aws configure
```

プロンプトが表示されたら、次の情報を入力します。

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-east-2
Default output format [None]: json
```

2. Unix または Linux の `.aws/config` ファイル、または Windows の `C:\Users\USERNAME\.aws\config` ファイルに、ロールの新しいプロファイルを作成します。次の例で

は、123456789012 アカountの *ProductionAccessRole* ロールへの切り替えをする「prodaccess」というプロファイルを作成します。ロール ARN は、ロールを作成したアカウント管理者から入手します。このプロファイルが呼び出されると、AWS CLI は source\_profile の認証情報を使用してロールのための認証情報をリクエストします。そのため、source\_profile として参照されるアイデンティティは、role\_arn で指定されたロールの sts:AssumeRole アクセス権限がなければなりません。

```
[profile prodaccess]
role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
source_profile = default
```

- 新しいプロファイルを作成した後、AWS CLI パラメータを指定した --profile prodaccess コマンドは、デフォルトのユーザーではなく、IAM ロールである ProductionAccessRole にアタッチされたアクセス権限により実行されます。

```
aws iam list-users --profile prodaccess
```

このコマンドが機能するのは、ProductionAccessRole に割り当てられたアクセス許可の下で現在の AWS アカountのユーザーを一覧表示できる場合です。

- 元の認証情報によって付与されるアクセス許可に戻すには、--profile パラメータなしでコマンドを実行します。AWS CLI は、「[Step 1](#)」で設定したデフォルトのプロファイルの認証情報の使用に戻ります。

詳細については、AWS Command Line Interface ユーザーガイドの「[ロールを引き受ける](#)」を参照してください。

シナリオ例: インスタンスプロファイルロールが他のアカウントのロールに切り替えることを許可する

2 つの AWS アカount を使用していて、Amazon EC2 インスタンスで実行されているアプリケーションが両方のアカウントで [AWS CLI](#) コマンドを実行できるようにするとします。アカウント 111111111111 に EC2 インスタンスが存在すると仮定します。このインスタンスには、同じ abcd アカount内の Amazon S3 バケットで読み取り専用の my-bucket-1 タスクをアプリケーションが実行する 111111111111 インスタンスプロファイルのロールが含まれています。ただし、アプリケーションは、アカウント 222222222222 でタスクを実行するために efgh クロスアカウントロールを引き受けることも許可されている必要があります。これを行うには、abcd EC2 インスタンスプロファイルのロールは次のアクセス許可ポリシーを持っている必要があります。

アカウント 111111111111 **abcd** ロールのアクセス許可ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3::*:*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-1/*",
        "arn:aws:s3:::my-bucket-1"
      ]
    },
    {
      "Sid": "AllowIPToAssumeCrossAccountRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::222222222222:role/efgh"
    }
  ]
}
```

efgh クロスアカウントのロールが、同じ my-bucket-2 アカウント内の 222222222222 バケットで読み取り専用 Amazon S3 タスクを許可すると仮定します。これを行うには、efgh クロスアカウントのロールは、以下のアクセス許可ポリシーを持っている必要があります。

アカウント 222222222222 **efgh** ロールのアクセス許可ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-2/*",
        "arn:aws:s3:::my-bucket-2"
      ]
    }
  ]
}
```

efgh ロールは abcd インスタンスプロファイルのロールがそれを引き受けることを許可する必要があります。これを行うには、efgh ロールは次の信頼ポリシーを持っている必要があります。

アカウント 222222222222 **efgh** ロールの信頼ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "efghTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
    }
  ]
}
```

```
]
}
```

次にアカウント 222222222222 で AWS CLI コマンドを実行するには、CLI 設定ファイルを更新する必要があります。AWS CLI 設定ファイルで、efgh ロールを「プロファイル」として、abcd EC2 インスタンスプロファイルロールを「認証情報ソース」として識別します。次に、元の abcd ロールではなく、efgh ロールのアクセス許可で CLI コマンドが実行されます。

#### Note

セキュリティ上の目的で、AWS CloudTrail を使用して、アカウントのロールの使用を監査することができます。CloudTrail ログで異なるプリンシパルのロールが使用されている場合にロールセッションを区別するには、ロールセッション名を使用できます。このトピックで説明しているように、AWS CLI がユーザーに代わってロールを引き受けると、ロールセッション名が自動的に `AWS-CLI-session-nnnnnnnn` の形式で作成されます。ここでは *nnnnnnnn* は [Unix エポック時刻](#) (1970 年 1 月 1 日午前 0 時 UTC からの秒数) 形式の時刻を表す整数です。詳細については、[AWS CloudTrail ユーザーガイド](#) の「CloudTrail イベントリファレンス」を参照してください。

EC2 インスタンスプロファイルのロールをクロスアカウントのロール (AWS CLI) に切り替えることができるようにするには

1. デフォルトの CLI プロファイルを設定する必要はありません。代わりに、EC2 インスタンスプロファイルのメタデータから認証情報を読み込むことができます。ロールの新しいプロファイルを `.aws/config` ファイルに作成します。次の例では、222222222222 アカウントのロール *efgh* に切り替える `instancecrossaccount` プロファイルを作成します。このプロファイルが呼び出されると、AWS CLI は EC2 インスタンスプロファイルのメタデータの認証情報を使用してロールの認証情報をリクエストします。そのため、EC2 インスタンスプロファイルのロールには、`role_arn` で指定されたロールに対する `sts:AssumeRole` アクセス許可が必要です。

```
[profile instancecrossaccount]
role_arn = arn:aws:iam::222222222222:role/efgh
credential_source = Ec2InstanceMetadata
```

2. 新しいプロファイルを作成した後、パラメータ `--profile instancecrossaccount` を指定するすべての AWS CLI コマンドは、アカウント 222222222222 の `efgh` ロールにアタッチされたアクセス許可で実行されます。

```
aws s3 ls my-bucket-2 --profile instancecrossaccount
```

このコマンドは、`efgh` ロールに割り当てられたアクセス許可で現在の AWS アカウントのユーザーを一覧表示できる場合に実行されます。

3. アカウント 111111111111 の元の EC2 インスタンスプロファイルのアクセス許可に戻るには、`--profile` パラメータなしで CLI コマンドを実行します。

詳細については、AWS Command Line Interface ユーザーガイドの「[ロールを引き受ける](#)」を参照してください。

## IAM ロールへの切り替え (Tools for Windows PowerShell)

ロールは、必要な AWS リソースへのアクセスに使用できる一連のアクセス許可を指定します。その点では、[AWS Identity and Access Management \( IAM \) のユーザー](#) に似ています。ユーザーとしてサインインすると、特定の一連のアクセス許可が付与されます。ただし、ロールにはサインインされませんが、一度サインインするとロールを切り替えることもできます。こうすると、元のユーザーアクセス権限が一時的に無効になり、そのロールに割り当てられたアクセス権限が代わりに付与されます。ロールは、自身のアカウントのロールでも、他の AWS アカウントのロールでもかまいません。ロールとその利点、およびロールを作成して設定する方法については、「[IAM ロール](#)」および「[IAM ロールの作成](#)」を参照してください。

### Important

IAM ユーザーと切り替え後のロールのアクセス許可は、累積されません。同時に有効になるアクセス権限のセットは 1 つのみです。ロールを切り替えると、ユーザーアクセス権限が一時的に無効になり、切り替え後のロールに割り当てられたアクセス権限が有効になります。そのロールを終了すると、ユーザーアクセス権限が自動的に復元されます。

このセクションでは、AWS Tools for Windows PowerShell のコマンドラインで作業するときロールを切り替える方法について説明します。

開発環境にアカウントがあり、場合によっては、本稼働環境で [Tools for Windows PowerShell](#) のコマンドラインで作業する必要があるとします。1 つのアクセスキー認証情報のセットがすでに使用可能



です。このセットは、標準の IAM ユーザーに割り当てられたアクセスキーペアです。または、フェデレーテッドユーザーとしてサインインしている場合は、最初に割り当てられたロールのアクセスキーペアです。これらの認証情報を使用して、パラメータとして新しいロールの ARN を渡す `Use-STSRole` コマンドレットを実行します。このコマンドは、リクエストされたロールの一時的なセキュリティ証明書を返します。それらの認証情報は、以降の PowerShell コマンドで、本稼働環境のリソースにアクセスするためのロールのアクセス権限と共に使用します。ロールの使用中に、開発アカウントのユーザーアクセス許可を使用することはできません。同時に有効になるアクセス許可のセットは 1 つに限られるためです。

### Note

セキュリティ上の理由から、管理者は [AWS CloudTrail ログを確認](#)して、AWS でアクションを実行したユーザーを調べることができます。管理者は、ロールを引き受けるときに、ソース ID またはロールセッション名の指定を要求する場合があります。詳細については、[sts:SourceIdentity](#) および [sts:RoleSessionName](#) を参照してください。

すべてのアクセスキーとトークンは例にすぎず、実際にはそのように使用できないことに注意してください。ライブ環境の適切な値に置き換えてください。

ロールに切り替えるには (Tools for Windows PowerShell)

1. PowerShell コマンドプロンプトを開き、現在の IAM ユーザーまたはフェデレーションロールのアクセスキーを使用するように、デフォルトのプロファイルを設定します。Tools for Windows PowerShell を以前に使用した場合、この設定はすでに完了している可能性があります。AWS アカウントのルートユーザーではなく、IAM ユーザーとしてサインインしている場合にのみ、ロールを切り替えることができます。

```
PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -  
SecretKey wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY -StoreAs MyMainUserProfile  
PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
```

詳細については、[AWS ユーザーガイド](#)の「AWS Tools for Windows PowerShell 認証情報の指定」を参照してください。

2. 新しいロールの認証情報を取得するには、以下のコマンドを実行して 123456789012 アカウントの `RoleName` ロールに切り替えます。ロール ARN は、ロールを作成したアカウント管理者から入手します。コマンドには、セッション名も指定する必要があります。その名前には任意の

テキストを選択できます。以下のコマンドは、認証情報をリクエストした後、返されたオブジェクトから Credentials プロパティオブジェクトを取得して、\$Creds 変数に格納します。

```
PS C:\> $Creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoLeName" -  
RoleSessionName "MyRoleSessionName").Credentials
```

\$Creds オブジェクトにはこの時点で、以降の手順に必要な AccessKeyId、SecretAccessKey、SessionToken 要素が格納されています。以下のサンプルコマンドに示しているのは、一般的な値です。

```
PS C:\> $Creds.AccessKeyId  
AKIAIOSFODNN7EXAMPLE
```

```
PS C:\> $Creds.SecretAccessKey  
wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

```
PS C:\> $Creds.SessionToken  
AQoDYXdzEGcaEXAMPLE2gsYULo  
+Im5ZEXAMPLEEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvSRyh0FW7jEXAMPLEW+vE/7s1HRp  
XviG7b+qYf4nD00EXAMPLEmj4wxS04L/uZEXAMPLECiHzFB51TYLto9dyBgSDyEXAMPLE9/  
g7QRUhZp4bqbEXAMPLENwGPY  
Oj59pFA41NKCikVgkREXAMPLEj1zxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEiYw  
C  
s8EXAMPLEEpZg0s+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==
```

```
PS C:\> $Creds.Expiration  
Thursday, June 18, 2018 2:28:31 PM
```

- 以降のコマンドでこれらの認証情報を使用するには、-Credential パラメータに指定します。たとえば、以下のコマンドでロールの認証情報を使用すると、そのロールに iam:ListRoles アクセス許可が付与されている場合にのみ、Get-IAMRoles コマンドレットを実行できます。

```
PS C:\> get-iamroles -Credential $Creds
```

- 元の認証情報に戻すには、-Credentials \$Creds パラメータの使用を止めるだけです。PowerShell ではデフォルトのプロファイルに保存された認証情報が再び使用されるようになります。

## IAM ロール (AWS API) の切り替え

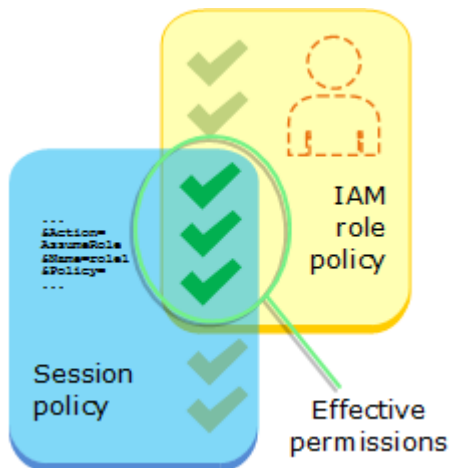
ロールは、AWS リソースへのアクセスに使用できる一連のアクセス許可を指定します。その点では、[IAM ユーザー](#)に似ています。ロールを引き受けることで、必要なタスクを実行したり、AWS リソースを操作したりするアクセス許可がプリンシパル (ユーザーまたはアプリケーション) に付与されます。ロールは、自身のアカウントのロールでも、他の AWS アカウント のロールでもかまいません。ロールとその利点、およびロールを作成して設定する方法については、「[IAM ロール](#)」および「[IAM ロールの作成](#)」を参照してください。ロールを引き受ける別の方法については、「[IAM ロールを使用する](#)」を参照してください。

### Important

IAM ユーザーのアクセス許可および引き受けるロールは、累積されません。同時に有効になるアクセス権限のセットは 1 つのみです。ロールを引き受けると、以前のユーザーまたはロールのアクセス許可が一時的に無効になり、切り替え後のロールに割り当てられたアクセス許可が有効になります。そのロールを終了すると、自動的に元のアクセス許可が復元されます。

ロールを引き受けるため、アプリケーションは AWS STS [AssumeRole](#) API オペレーションを呼び出して、使用するロールの ARN を渡します。このオペレーションでは、一時的な認証情報で新しいセッションを作成します。このセッションのアクセス許可は、そのロールのアイデンティティベースのポリシーと同じです。

[AssumeRole](#) を呼び出すと、必要に応じてインラインまたは管理[セッションポリシー](#)を渡すことができます。セッションポリシーは、ロールまたはフェデレーテッドユーザーの一時認証情報セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。Policy パラメータを使用して単一の JSON インラインセッションポリシードキュメントを渡すことができます。PolicyArns パラメータを使用して、最大 10 個の管理セッションポリシーを指定できます。結果として得られるセッションのアクセス許可は、エンティティの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーは、ロールの一時的な認証情報を他のユーザーに提供する必要があるときに役立ちます。他のユーザーは、ロールの一時的な認証情報を以降の AWS API コールで使用し、ロールを所有するアカウント内のリソースにアクセスできます。セッションポリシーを使用して、ID ベースのポリシーで許可されているよりも多くのアクセス許可を付与することはできません。AWS でロールの有効なアクセス許可を決定する方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。



IAM ユーザーまたはすでにロールを使用している [外部で認証されたユーザー \(SAML または OIDC\)](#) としてサインインすると、`AssumeRole` を呼び出すことができます。ロールを使用して 2 つ目のロールを引き受ける、[ロールの連鎖](#)を使用することもできます。AWS アカウントのルートユーザーとしてサインインしているときに、ロールを引き受けることはできません。

デフォルトでは、ロールセッションは 1 時間です。AWS STS [AssumeRole\\*](#) API オペレーションを使用してこのロールを引き受ける場合は、`DurationSeconds` パラメータの値を指定できます。この値は 900 秒 (15 分) からロールの最大セッション期間設定までの範囲を指定できます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。

ロールの連鎖を使用すると、セッションは最長である 1 時間に制限されます。この場合 `DurationSeconds` パラメータを使用して 1 時間より大きい値を指定すると、オペレーションは失敗します。

#### Note

セキュリティ上の理由から、管理者は [AWS CloudTrail ログを確認](#)して、AWS でアクションを実行したユーザーを調べることができます。管理者は、ロールを引き受けるときに、ソース ID またはロールセッション名の指定を要求する場合があります。詳細については、[sts:SourceIdentity](#) および [sts:RoleSessionName](#) を参照してください。

次のコード例は、ユーザーを作成しロールを割り当てる方法を示しています。

**⚠ Warning**

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#)などの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- 指定したアカウントに Amazon S3 バケットへのアクセス権限を付与するロールを作成します。
- ユーザーにロールを引き受けさせるポリシーを追加します。
- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示しリソースをクリーンアップします。

**.NET****AWS SDK for .NET****i Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;

namespace IAMActions;

public class IAMWrapper
{
```

```
private readonly IAMIdentityManagementService _IAMService;

/// <summary>
/// Constructor for the IAMWrapper class.
/// </summary>
/// <param name="IAMService">An IAM client object.</param>
public IAMWrapper(IAMIdentityManagementService IAMService)
{
    _IAMService = IAMService;
}

/// <summary>
/// Add an existing IAM user to an existing IAM group.
/// </summary>
/// <param name="userName">The username of the user to add.</param>
/// <param name="groupName">The name of the group to add the user to.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
{
    var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
    {
        GroupName = groupName,
        UserName = userName,
    });

    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Attach an IAM policy to a role.
/// </summary>
/// <param name="policyArn">The policy to attach.</param>
/// <param name="roleName">The role that the policy will be attached to.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AttachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
    {
```

```
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Create an IAM access key for a user.
/// </summary>
/// <param name="userName">The username for which to create the IAM access
/// key.</param>
/// <returns>The AccessKey.</returns>
public async Task<AccessKey> CreateAccessKeyAsync(string userName)
{
    var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
    {
        UserName = userName,
    });

    return response.AccessKey;
}

/// <summary>
/// Create an IAM group.
/// </summary>
/// <param name="groupName">The name to give the IAM group.</param>
/// <returns>The IAM group that was created.</returns>
public async Task<Group> CreateGroupAsync(string groupName)
{
    var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
{ GroupName = groupName });
    return response.Group;
}

/// <summary>
/// Create an IAM policy.
/// </summary>
/// <param name="policyName">The name to give the new IAM policy.</param>
```



```
    /// <param name="policyDocument">The policy document for the new policy.</  
param>  
    /// <returns>The new IAM policy object.</returns>  
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string  
policyDocument)  
    {  
        var response = await _IAMService.CreatePolicyAsync(new  
CreatePolicyRequest  
        {  
            PolicyDocument = policyDocument,  
            PolicyName = policyName,  
        });  
  
        return response.Policy;  
    }  
  
    /// <summary>  
    /// Create a new IAM role.  
    /// </summary>  
    /// <param name="roleName">The name of the IAM role.</param>  
    /// <param name="rolePolicyDocument">The name of the IAM policy document  
    /// for the new role.</param>  
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>  
    public async Task<string> CreateRoleAsync(string roleName, string  
rolePolicyDocument)  
    {  
        var request = new CreateRoleRequest  
        {  
            RoleName = roleName,  
            AssumeRolePolicyDocument = rolePolicyDocument,  
        };  
  
        var response = await _IAMService.CreateRoleAsync(request);  
        return response.Role.Arn;  
    }  
  
    /// <summary>  
    /// Create an IAM service-linked role.  
    /// </summary>  
    /// <param name="serviceName">The name of the AWS Service.</param>  
    /// <param name="description">A description of the IAM service-linked role.</  
param>
```

```
/// <returns>The IAM role that was created.</returns>
public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
string description)
{
    var request = new CreateServiceLinkedRoleRequest
    {
        AWSServiceName = serviceName,
        Description = description
    };

    var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
    return response.Role;
}

/// <summary>
/// Create an IAM user.
/// </summary>
/// <param name="userName">The username for the new IAM user.</param>
/// <returns>The IAM user that was created.</returns>
public async Task<User> CreateUserAsync(string userName)
{
    var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
    return response.User;
}

/// <summary>
/// Delete an IAM user's access key.
/// </summary>
/// <param name="accessKeyId">The Id for the IAM access key.</param>
/// <param name="userName">The username of the user that owns the IAM
/// access key.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
{
    var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
    {
        AccessKeyId = accessKeyId,
        UserName = userName,
    });
};
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
        { GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
    policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
```

```
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeletePolicyAsync(string policyArn)
{
    var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRoleAsync(string roleName)
{
    var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM role policy.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="policyName">The name of the IAM role policy to delete.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
{
    var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
    });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
```

```
/// Delete an IAM user.
/// </summary>
/// <param name="userName">The username of the IAM user to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserAsync(string userName)
{
    var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM user policy.
/// </summary>
/// <param name="policyName">The name of the IAM policy to delete.</param>
/// <param name="userName">The username of the IAM user.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
{
    var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Detach an IAM policy from an IAM role.
/// </summary>
/// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
/// <param name="roleName">The name of the IAM role.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
```

```
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Gets the IAM password policy for an AWS account.
/// </summary>
/// <returns>The PasswordPolicy for the AWS account.</returns>
public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
{
    var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
    return response.PasswordPolicy;
}

/// <summary>
/// Get information about an IAM policy.
/// </summary>
/// <param name="policyArn">The IAM policy to retrieve information for.</
param>
/// <returns>The IAM policy.</returns>
public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
{
    var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
    return response.Policy;
}

/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
{
        RoleName = roleName,
```

```
    });

    return response.Role;
}

/// <summary>
/// Get information about an IAM user.
/// </summary>
/// <param name="userName">The username of the user.</param>
/// <returns>An IAM user object.</returns>
public async Task<User> GetUserAsync(string userName)
{
    var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
    return response.User;
}

/// <summary>
/// List the IAM role policies that are attached to an IAM role.
/// </summary>
/// <param name="roleName">The IAM role to list IAM policies for.</param>
/// <returns>A list of the IAM policies attached to the IAM role.</returns>
public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
{
    var attachedPolicies = new List<AttachedPolicyType>();
    var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

    await foreach (var response in attachedRolePoliciesPaginator.Responses)
    {
        attachedPolicies.AddRange(response.AttachedPolicies);
    }

    return attachedPolicies;
}

/// <summary>
/// List IAM groups.
/// </summary>
```



```
/// <returns>A list of IAM groups.</returns>
public async Task<List<Group>> ListGroupsAsync()
{
    var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
    var groups = new List<Group>();

    await foreach (var response in groupsPaginator.Responses)
    {
        groups.AddRange(response.Groups);
    }

    return groups;
}

/// <summary>
/// List IAM policies.
/// </summary>
/// <returns>A list of the IAM policies.</returns>
public async Task<List<ManagedPolicy>> ListPoliciesAsync()
{
    var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
    var policies = new List<ManagedPolicy>();

    await foreach (var response in listPoliciesPaginator.Responses)
    {
        policies.AddRange(response.Policies);
    }

    return policies;
}

/// <summary>
/// List IAM role policies.
/// </summary>
/// <param name="roleName">The IAM role for which to list IAM policies.</
param>
/// <returns>A list of IAM policy names.</returns>
public async Task<List<string>> ListRolePoliciesAsync(string roleName)
{
```

```
        var listRolePoliciesPaginator =
_IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }

    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }

    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
}
```

```
/// <summary>
/// List IAM users.
/// </summary>
/// <returns>A list of IAM users.</returns>
public async Task<List<User>> ListUsersAsync()
{
    var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
    var users = new List<User>();

    await foreach (var response in listUsersPaginator.Responses)
    {
        users.AddRange(response.Users);
    }

    return users;
}

/// <summary>
/// Remove a user from an IAM group.
/// </summary>
/// <param name="userName">The username of the user to remove.</param>
/// <param name="groupName">The name of the IAM group to remove the user
from.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
{
    // Remove the user from the group.
    var removeUserRequest = new RemoveUserFromGroupRequest()
    {
        UserName = userName,
        GroupName = groupName,
    };

    var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
```

```
/// Add or update an inline policy document that is embedded in an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
{
    var request = new PutGroupPolicyRequest
    {
        GroupName = groupName,
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutGroupPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Update the inline policy document embedded in a role.
/// </summary>
/// <param name="policyName">The name of the policy to embed.</param>
/// <param name="roleName">The name of the role to update.</param>
/// <param name="policyDocument">The policy document that defines the role.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutRolePolicyAsync(string policyName, string
roleName, string policyDocument)
{
    var request = new PutRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutRolePolicyAsync(request);
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

```
/// <summary>
/// Add or update an inline policy document that is embedded in an IAM user.
/// </summary>
/// <param name="userName">The name of the IAM user.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
{
    var request = new PutUserPolicyRequest
    {
        UserName = userName,
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutUserPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Wait for a new access key to be ready to use.
/// </summary>
/// <param name="accessKeyId">The Id of the access key.</param>
/// <returns>A boolean value indicating the success of the action.</returns>
public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
{
    var keyReady = false;

    do
    {
        try
        {
            var response = await _IAMService.GetAccessKeyLastUsedAsync(
                new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
            if (response.UserName is not null)
            {
                keyReady = true;
            }
        }
    }
}
```

```
        catch (NoSuchEntityException)
        {
            keyReady = false;
        }
    } while (!keyReady);

    return keyReady;
}
}

using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<IAMWrapper>()
                    .AddTransient<UIWrapper>()
                )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();

        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
    }
}
```

```
.AddJsonFile("settings.local.json",
    true) // Optionally load local settings.
.Build();

// Values needed for user, role, and policies.
string userName = configuration["UserName"]!;
string s3PolicyName = configuration["S3PolicyName"]!;
string roleName = configuration["RoleName"]!;

var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

uiWrapper.DisplayBasicsOverview();
uiWrapper.PressEnter();

// First create a user. By default, the new user has
// no permissions.
uiWrapper.DisplayTitle("Create User");
Console.WriteLine($"Creating a new user with user name: {userName}.");
var user = await iamWrapper.CreateUserAsync(userName);
var userArn = user.Arn;

Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

// Define a role policy document that allows the new user
// to assume the role.
string assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
            "\"AWS\": \"{userArn}\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

// Permissions to list all buckets.
string policyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
```



```
        " \"Statement\" : [{\" +
            \" \"Action\" : [\"s3:ListAllMyBuckets\"],\" +
            \" \"Effect\" : \"Allow\",\" +
            \" \"Resource\" : \"*\"]\" +
        \"}]" +
    "};

// Create an AccessKey for the user.
uiWrapper.DisplayTitle("Create access key");
Console.WriteLine("Now let's create an access key for the new user.");
var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

var accessKeyId = accessKey.AccessKeyId;
var secretAccessKey = accessKey.SecretAccessKey;

Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

// Now try listing the Amazon Simple Storage Service (Amazon S3)
// buckets. This should fail at this point because the user doesn't
// have permissions to perform this task.
uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
var buckets = await s3Wrapper.ListMyBucketsAsync();

Console.WriteLine(buckets is null
    ? "As expected, the call to list the buckets has returned a null
list."
    : "Something went wrong. This shouldn't have worked.");

uiWrapper.PressEnter();

uiWrapper.DisplayTitle("Create IAM role");
Console.WriteLine($"Creating the role: {roleName}");
```

```
// Creating an IAM role to allow listing the S3 buckets. A role name
// is not case sensitive and must be unique to the account for which it
// is created.
var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

uiWrapper.PressEnter();

// Create a policy with permissions to list S3 buckets.
uiWrapper.DisplayTitle("Create IAM policy");
Console.WriteLine($"Creating the policy: {s3PolicyName}");
Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

// Wait 15 seconds for the IAM policy to be available.
uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

// Attach the policy to the role you created earlier.
uiWrapper.DisplayTitle("Attach new IAM policy");
Console.WriteLine("Now let's attach the policy to the role.");
await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

// Wait 15 seconds for the role to be updated.
Console.WriteLine();
uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

// Use the AWS Security Token Service (AWS STS) to have the user
// assume the role we created.
var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

// Wait for the new credentials to become valid.
uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

// Try again to list the buckets using the client created with
// the new user's credentials. This time, it should work.
var s3Client2 = new AmazonS3Client(assumedRoleCredentials);
```

```
s3Wrapper.UpdateClients(s3Client2, stsClient2);

buckets = await s3Wrapper.ListMyBucketsAsync();

uiWrapper.DisplayTitle("List Amazon S3 buckets");
Console.WriteLine("This time we should have buckets to list.");
if (buckets is not null)
{
    buckets.ForEach(bucket =>
    {
        Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
    });
}

uiWrapper.PressEnter();

// Now clean up all the resources used in the example.
uiWrapper.DisplayTitle("Clean up resources");
Console.WriteLine("Thank you for watching. The IAM Basics demo is
complete.");
Console.WriteLine("Please wait while we clean up the resources we
created.");

await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

await iamWrapper.DeletePolicyAsync(policy.Arn);

await iamWrapper.DeleteRoleAsync(roleName);

await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

await iamWrapper.DeleteUserAsync(userName);

uiWrapper.PressEnter();

Console.WriteLine("All done cleaning up our resources. Thank you for your
patience.");
}
}

namespace IamScenariosCommon;
```

```
using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }
}
```

```
}

/// <summary>
/// Delete an S3 bucket.
/// </summary>
/// <param name="bucketName">Name of the S3 bucket to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
    return result.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// List the buckets that are owned by the user's account.
/// </summary>
/// <returns>Async Task.</returns>
public async Task<List<S3Bucket>?> ListMyBucketsAsync()
{
    try
    {
        // Get the list of buckets accessible by the new user.
        var response = await _s3Service.ListBucketsAsync();

        return response.Buckets;
    }
    catch (AmazonS3Exception ex)
    {
        // Something else went wrong. Display the error message.
        Console.WriteLine($"Error: {ex.Message}");
        return null;
    }
}

/// <summary>
/// Create a new S3 bucket.
/// </summary>
/// <param name="bucketName">The name for the new bucket.</param>
/// <returns>A Boolean value indicating whether the action completed
/// successfully.</returns>
public async Task<bool> PutBucketAsync(string bucketName)
{
```

```
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}

namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
    }
}
```

```
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
```



```
{
    var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
    var leftPad = new string(' ', padAmount);
    return $"{leftPad}{strToCenter}";
}

/// <summary>
/// Display a line of hyphens, the centered text of the title, and another
/// line of hyphens.
/// </summary>
/// <param name="strTitle">The string to be displayed.</param>
public void DisplayTitle(string strTitle)
{
    Console.WriteLine(SepBar);
    Console.WriteLine(CenterString(strTitle));
    Console.WriteLine(SepBar);
}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }

    PressEnter();
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)

- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####  
# function iam_create_user_assume_role  
#  
# Scenario to create an IAM user, create an IAM role, and apply the role to the  
# user.  
#  
# "IAM access" permissions are needed to run this code.  
# "STS assume role" permissions are needed to run this code. (Note: It might  
# be necessary to  
# create a custom policy).  
#  
# Returns:  
# 0 - If successful.  
# 1 - If an error occurred.  
#####
```

```
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}
```

```
echo "Created access key named $key_name"

echo "Wait 10 seconds for the user to be ready."
sleep 10
echo_repeat "*" 88
echo

local iam_role_name
iam_role_name=$(generate_random_name "test-role")
echo "Creating a role named $iam_role_name with user $user_name as the
principal."

local assume_role_policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Principal\": {\"AWS\": \"$user_arn\"},
    \"Action\": \"sts:AssumeRole\"
  }]
}"

local role_arn
role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

# shellcheck disable=SC2181
if [ $? == 0 ]; then
  echo "Created IAM role named $iam_role_name"
else
  errecho "The role failed to create. This demo will exit."
  clean_up "$user_name" "$key_name"
  return 1
fi

local policy_name
policy_name=$(generate_random_name "test-policy")
local policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Action\": \"s3:ListAllMyBuckets\",
    \"Resource\": \"arn:aws:s3:::*\"}]}"
```

```
local policy_arn
policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
# shellcheck disable=SC2181
if [[ $? == 0 ]]; then
    echo "Created IAM policy named $policy_name"
else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
fi

if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
fi

local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"sts:AssumeRole\",
        \"Resource\": \"$role_arn\"}]}"

local assume_role_policy_name
assume_role_policy_name=$(generate_random_name "test-assume-role-")

# shellcheck disable=SC2181
local assume_role_policy_arn
assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
# shellcheck disable=SC2181
if [ $? == 0 ]; then
    echo "Created IAM policy named $assume_role_policy_name for sts assume role"
else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
    return 1
fi
```

```
echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
sleep 10
echo_repeat "*" 88
echo

echo "Try to list buckets without the new user assuming the role."
echo_repeat "*" 88
echo

# Set the environment variables for the created user.
# bashsupport disable=BP2001
export AWS_ACCESS_KEY_ID=$key_name
# bashsupport disable=BP2001
export AWS_SECRET_ACCESS_KEY=$key_secret

local buckets
buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
else
    errecho "Because the role with permissions has not been assumed, listing
buckets failed."
fi

echo
echo_repeat "*" 88
echo "Now assume the role $iam_role_name and list the buckets."
echo_repeat "*" 88
echo

local credentials
credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
else
    errecho "Failed to assume role."
```

```
export AWS_ACCESS_KEY_ID=""
export AWS_SECRET_ACCESS_KEY=""
clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
return 1
fi

IFS=$'\t ' read -r -a credentials <<<"$credentials"

export AWS_ACCESS_KEY_ID=${credentials[0]}
export AWS_SECRET_ACCESS_KEY=${credentials[1]}
# bashsupport disable=BP2001
export AWS_SESSION_TOKEN=${credentials[2]}

buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
succeeded because of "
    echo "the assumed role."
else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
    return 1
fi

local result=0
export AWS_ACCESS_KEY_ID=""
export AWS_SECRET_ACCESS_KEY=""

echo
echo_repeat "*" 88
echo "The created resources will now be deleted."
echo_repeat "*" 88
echo
```



```

clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    result=1
fi

return $result
}

```

このシナリオで使用される IAM 関数。

```

#####
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
# (IAM) user already exists.
#
# Parameters:
#     $1 - The name of the IAM user to check.
#
# Returns:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_user_exists() {
    local user_name
    user_name=$1

    # Check whether the IAM user already exists.
    # We suppress all output - we're interested only in the return code.

    local errors
    errors=$(aws iam get-user \
        --user-name "$user_name" 2>&1 >/dev/null)

    local error_code=${?}

    if [[ $error_code -eq 0 ]]; then
        return 0 # 0 in Bash script means true.
    else

```

```

    if [[ $errors != *"error"*(NoSuchEntity)* ]]; then
        aws_cli_error_log $error_code
        errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
fi
}

#####
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#     -u user_name  -- The name of the user to create.
#
# Returns:
#     The ARN of the user.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user() {
    local user_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user"
        echo "Creates an WS Identity and Access Management (IAM) user. You must
supply a username:"
        echo "  -u user_name    The name of the user. It must be unique within the
account."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            h)
                usage

```

```
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    User name:  $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
fi

echo "$response"

return 0
}
```

```
#####
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#   -u user_name -- The name of the IAM user.
#   [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#   [access_key_id access_key_secret]
#   And:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_create_user_access_key() {
    local user_name file_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) key pair."
        echo "  -u user_name   The name of the IAM user."
        echo "  [-f file_name] Optional file name for the access key output."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:f:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            f) file_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}
```

```
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
fi

if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
fi

local key_id key_secret
# shellcheck disable=SC2086
key_id=$(echo $response | cut -f 2 -d ' ')
# shellcheck disable=SC2086
key_secret=$(echo $response | cut -f 4 -d ' ')

echo "$key_id $key_secret"

return 0
}

#####
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_json -- The assume role policy document.
#
```

```
# Returns:
#     The ARN of the role.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_role() {
    local role_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_json  -- The assume role policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$role_name" ]]; then
        errecho "ERROR: You must provide a role name with the -n parameter."
        usage
        return 1
    fi

    if [[ -z "$policy_document" ]]; then
```

```

    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#     -n policy_name -- The name of the IAM policy.
#     -p policy_json -- The policy document.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_policy() {
    local policy_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_policy"
    }

```



```
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name    The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) policy_name="${OPTARG}" ;;
        p) policy_document="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

local error_code=${?}
```

```

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
fi

echo "$response"
}

#####
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_arn -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_attach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_attach_role_policy"
        echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_arn -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done

```

```

        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a role.
#

```

```
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_detach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_detach_role_policy"
        echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_ARN -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$role_name" ]]; then
        errecho "ERROR: You must provide a role name with the -n parameter."
        usage
        return 1
    fi
}
```

```

fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#     -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_policy() {
    local policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_policy"
    }

```

```
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "  Policy arn: $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
  return 1
fi

iecho "delete-policy response:$response"
```

```

iecho

return 0
}

#####
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_role() {
    local role_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_role"
        echo "Deletes an WS Identity and Access Management (IAM) role"
        echo "  -n role_name -- The name of the IAM role."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```



```

export OPTIND=1

echo "role_name:$role_name"
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Role name:  $role_name"
iecho ""

response=$(aws iam delete-role \
    --role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
fi

iecho "delete-role response:$response"
iecho

return 0
}

#####
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user.
#     -k access_key -- The access key to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_access_key() {

```

```
local user_name access_key response
local option OPTARG # Required to use getopt command in a function.

# bashsupport disable=BP5008
function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key    The access key to delete."
    echo ""
}

# Retrieve the calling parameters.
while getopt "u:k:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
```

```

iecho " Username: $user_name"
iecho " Access key: $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi

iecho "delete-access-key response:$response"
iecho

return 0
}

#####
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#   -u user_name -- The name of the user to create.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopt command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
supply a username:"
    echo "  -u user_name    The name of the user."
  }
}

```

```
    echo ""
}

# Retrieve the calling parameters.
while getopts "u:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    User name:  $user_name"
iecho ""

# If the user does not exist, we don't want to try to delete it.
if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
fi

response=$(aws iam delete-user \
    --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
```

```
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」で以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
namespace AwsDoc {
```

```
namespace IAM {

    //! Cleanup by deleting created entities.
    /*!
        \sa DeleteCreatedEntities
        \param client: IAM client.
        \param role: IAM role.
        \param user: IAM user.
        \param policy: IAM policy.
    */
    static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                     const Aws::IAM::Model::Role &role,
                                     const Aws::IAM::Model::User &user,
                                     const Aws::IAM::Model::Policy &policy);
}

static const int LIST_BUCKETS_WAIT_SEC = 20;

static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
necessary to
//   create a custom policy).
/*!
    \sa iamCreateUserAssumeRoleScenario
    \param clientConfig: Aws client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
```

```
Aws::String userName = "iam-demo-user-" +
    Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetUserName(userName);

Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
if (!outcome.IsSuccess()) {
    std::cout << "Error creating IAM user " << userName << ":" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
else {
    std::cout << "Successfully created IAM user " << userName <<
std::endl;
}

    user = outcome.GetResult().GetUser();
}

// 2. Create a role.
{
    // Get the IAM user for the current client in order to access its ARN.
    Aws::String iamUserArn;
    {
        Aws::IAM::Model::GetUserRequest request;
        Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error getting Iam user. " <<
                outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully retrieved Iam user "
                << outcome.GetResult().GetUser().GetUserName()
                << std::endl;
        }

        iamUserArn = outcome.GetResult().GetUser().GetArn();
    }

    Aws::IAM::Model::CreateRoleRequest request;

    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
```



```
Aws::String roleName = "iam-demo-role-" +
    Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetRoleName(roleName);

// Build policy document for role.
Aws::Utils::Document jsonStatement;
jsonStatement.WithString("Effect", "Allow");

Aws::Utils::Document jsonPrincipal;
jsonPrincipal.WithString("AWS", iamUserArn);
jsonStatement.WithObject("Principal", jsonPrincipal);
jsonStatement.WithString("Action", "sts:AssumeRole");
jsonStatement.WithObject("Condition", Aws::Utils::Document());

Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Setting policy for role\n  "
    << policyDocument.View().WriteCompact() << std::endl;

// Set role policy document as JSON string.
request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a role with name " << roleName
        << std::endl;
}

role = outcome.GetResult().GetRole();
}
```

```
// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3::*");

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array<Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    std::cout << "Creating a policy.\n    " <<
policyDocument.View().WriteCompact()
        << std::endl;

    // Set IAM policy document as JSON string.
    request.SetPolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully created a policy with name, " <<
policyName <<
            "." << std::endl;
    }

    policy = outcome.GetResult().GetPolicy();
}
```

```
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSClient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtil::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);

    Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

    // Repeatedly call AssumeRole, because there is often a delay
    // before the role is available to be assumed.
    // Repeat at most 20 times when access is denied.
    int count = 0;
    while (true) {
        assumeRoleOutcome = stsClient.AssumeRole(request);
        if (!assumeRoleOutcome.IsSuccess()) {
            if (count > 20 ||
                assumeRoleOutcome.GetError().GetErrorType() !=
                Aws::STS::STSErrors::ACCESS_DENIED) {
                std::cerr << "Error assuming role after 20 tries. " <<
                    assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            std::cout << "Successfully assumed the role after " << count
                << " seconds." << std::endl;
            break;
        }
        count++;
    }
}
```

```
        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }

    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
            Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                      credentials.GetSecretAccessKey(),
                                      credentials.GetSessionToken()),
            Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
            clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                    listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                    << "Access to list buckets denied because privileges have
not been applied."
                    << std::endl;
            }
        }
        else {
            std::cerr
                << "Successfully retrieved bucket lists when this should not
happen."
                << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
```

```
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully attached the policy with name, "
            << policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." <<
std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the
role.
// Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
while (true) {
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if ((count > LIST_BUCKETS_WAIT_SEC) ||
            listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                listBucketsOutcome.GetError().GetMessage() <<
std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
```

```
    }
    else {

        std::cout << "Successfully retrieved bucket lists after " << count
                  << " seconds." << std::endl;

        break;
    }
    count++;
}

// 8. Delete all the created resources.
return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {

    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
            request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                        outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
```

```
        Aws::IAM::Model::DeletePolicyRequest request;
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting policy. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the policy with arn "
                << policy.GetArn() << std::endl;
        }
    }
}

if (role.RoleIdHasBeenSet()) {
    // Delete the role.
    Aws::IAM::Model::DeleteRoleRequest request;
    request.SetRoleName(role.GetRoleName());

    Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting role. " <<
            outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Successfully deleted the role with name "
            << role.GetRoleName() << std::endl;
    }
}

if (user.ArnHasBeenSet()) {
    // Delete the user.
    Aws::IAM::Model::DeleteUserRequest request;
    request.WithUserName(user.GetUserName());

    Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting user. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
}
```


```
        result = false;
    }
    else {
        std::cout << "Successfully deleted the user with name "
                  << user.GetUserName() << std::endl;
    }
}

return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。



コマンドプロンプトからインタラクティブのシナリオを実行します。

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
// (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
    sdkConfig aws.Config
    accountWrapper actions.AccountWrapper
    policyWrapper actions.PolicyWrapper
    roleWrapper actions.RoleWrapper
    userWrapper actions.UserWrapper
    questioner demotools.IQuestioner
    helper IScenarioHelper
    isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
// configuration.
// It uses the specified config to get an IAM client and create wrappers for the
// actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner,
    helper IScenarioHelper) AssumeRoleScenario {
    iamClient := iam.NewFromConfig(sdkConfig)
    return AssumeRoleScenario{
        sdkConfig:    sdkConfig,
        accountWrapper: actions.AccountWrapper{IamClient: iamClient},
        policyWrapper: actions.PolicyWrapper{IamClient: iamClient},
        roleWrapper:   actions.RoleWrapper{IamClient: iamClient},
        userWrapper:   actions.UserWrapper{IamClient: iamClient},
        questioner:    questioner,
        helper:        helper,
    }
}
```

```
// addTestOptions appends the API options specified in the original configuration
to
// another configuration. This is used to attach the middleware stubber to
clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
    if scenario.isTestRun {
        scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
scenario.sdkConfig.APIOptions...)
    }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong with the demo.\n")
            log.Println(r)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
demo.")
    log.Println(strings.Repeat("-", 88))

    user := scenario.CreateUser()
    accessKey := scenario.CreateAccessKey(user)
    role := scenario.CreateRoleAndPolicies(user)
    noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
    scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
    scenario.Cleanup(user, role)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
    log.Println("Let's create an example user with no permissions.")
    userName := scenario.questioner.Ask("Enter a name for the example user:",
demotools.NotEmpty{})
}
```

```
user, err := scenario.userWrapper.GetUser(userName)
if err != nil {
    panic(err)
}
if user == nil {
    user, err = scenario.userWrapper.CreateUser(userName)
    if err != nil {
        panic(err)
    }
    log.Printf("Created user %v.\n", *user.UserName)
} else {
    log.Printf("User %v already exists.\n", *user.UserName)
}
log.Println(strings.Repeat("-", 88))
return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
*types.AccessKey {
    accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
    if err != nil {
        panic(err)
    }
    log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
    log.Println("Waiting a few seconds for your user to be ready...")
    scenario.helper.Pause(10)
    log.Println(strings.Repeat("-", 88))
    return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
// buckets for
// the current account and attaches the policy to a newly created role. It also
// adds an
// inline policy to the specified user that grants the user permission to assume
// the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
*types.Role {
    log.Println("Let's create a role and policy that grant permission to list S3
buckets.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    listBucketsRole, err :=
scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
```

```
if err != nil {panic(err)}
log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
    scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
if err != nil {panic(err)}
log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
    *listBucketsRole.RoleName)
if err != nil {panic(err)}
log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
    *listBucketsRole.RoleName)
err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
    scenario.helper.GetName(),
    []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
if err != nil {panic(err)}
log.Printf("Created an inline policy for user %v that lets the user assume the
role.\n",
    *user.UserName)
log.Println("Let's give AWS a few seconds to propagate these new resources and
connections...")
scenario.helper.Pause(10)
log.Println(strings.Repeat("-", 88))
return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
access key
// credentials and tries to list buckets for the account. Because the user does
not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
    *types.AccessKey) *aws.Config {
    log.Println("Let's try to list buckets without permissions. This should return
an AccessDenied error.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
    config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
        *accessKey.AccessKeyId, *accessKey.SecretAccessKey, "")),
    ))
    if err != nil {panic(err)}

    // Add test options if this is a test run. This is needed only for testing
purposes.
    scenario.addTestOptions(&noPermsConfig)
```

```
s3Client := s3.NewFromConfig(noPermsConfig)
_, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
if err != nil {
    // The SDK for Go does not model the AccessDenied error, so check ErrorCode
    directly.
    var ae smithy.APIError
    if errors.As(err, &ae) {
        switch ae.ErrorCode() {
        case "AccessDenied":
            log.Println("Got AccessDenied error, which is the expected result because\n"
+
            "the ListBuckets call was made without permissions.")
        default:
            log.Println("Expected AccessDenied, got something else.")
            panic(err)
        }
    }
} else {
    log.Println("Expected AccessDenied error when calling ListBuckets without
permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
}
log.Println(strings.Repeat("-", 88))
return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
//    created from
//    the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
//    list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
//    generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
*aws.Config, role *types.Role) {
    log.Println("Let's assume the role that grants permission to list buckets and
try again.")
    scenario.questioner.Ask("Press Enter when you're ready.")
}
```

```
stsClient := sts.NewFromConfig(*noPermsConfig)
tempCredentials, err := stsClient.AssumeRole(context.TODO(),
&sts.AssumeRoleInput{
    RoleArn:          role.Arn,
    RoleSessionName: aws.String("AssumeRoleExampleSession"),
    DurationSeconds:  aws.Int32(900),
})
if err != nil {
    log.Printf("Couldn't assume role %v.\n", *role.RoleName)
    panic(err)
}
log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
    config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
        *tempCredentials.Credentials.AccessKeyId,
        *tempCredentials.Credentials.SecretAccessKey,
        *tempCredentials.Credentials.SessionToken),
    ),
)
if err != nil {panic(err)}

// Add test options if this is a test run. This is needed only for testing
purposes.
scenario.addTestOptions(&assumeRoleConfig)

s3Client := s3.NewFromConfig(assumeRoleConfig)
result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
if err != nil {
    log.Println("Couldn't list buckets with assumed role credentials.")
    panic(err)
}
log.Println("Successfully called ListBuckets with assumed role credentials, \n"
+
    "here are some of them:")
for i := 0; i < len(result.Buckets) && i < 5; i++ {
    log.Printf("\t%v\n", *result.Buckets[i].Name)
}
log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
    if scenario.questioner.AskBool(
        "Do you want to delete the resources created for this example? (y/n)", "y",
```

```
) {
    policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
    if err != nil {panic(err)}
    for _, policy := range policies {
        err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
*policy.PolicyArn)
        if err != nil {panic(err)}
        err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
        if err != nil {panic(err)}
        log.Printf("Detached policy %v from role %v and deleted the policy.\n",
            *policy.PolicyName, *role.RoleName)
    }
    err = scenario.roleWrapper.DeleteRole(*role.RoleName)
    if err != nil {panic(err)}
    log.Printf("Deleted role %v.\n", *role.RoleName)

    userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
    if err != nil {panic(err)}
    for _, userPol := range userPols {
        err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
        if err != nil {panic(err)}
        log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
    }
    keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
    if err != nil {panic(err)}
    for _, key := range keys {
        err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
        if err != nil {panic(err)}
        log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
*user.UserName)
    }
    err = scenario.userWrapper.DeleteUser(*user.UserName)
    if err != nil {panic(err)}
    log.Printf("Deleted user %v.\n", *user.UserName)
    log.Println(strings.Repeat("-", 88))
}
}
```

アカウントアクションをラップする構造体を定義します。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
// actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
    iamClient *iam.Client
}

// GetAccountPasswordPolicy gets the account password policy for the current
// account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
error) {
    var pwPolicy *types.PasswordPolicy
    result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
        &iam.GetAccountPasswordPolicyInput{})
    if err != nil {
        log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
    } else {
        pwPolicy = result.PasswordPolicy
    }
    return pwPolicy, err
}

// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
error) {
    var providers []types.SAMLProviderListEntry
    result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
        &iam.ListSAMLProvidersInput{})
    if err != nil {
        log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
    } else {
        providers = result.SAMLProviderList
    }
    return providers, err
}
```



ポリシーアクションをラップする構造体を定義します。

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect string
    Action []string
    Principal map[string]string `json:",omitempty"`
    Resource *string `json:",omitempty"`
}

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
// actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    IamClient *iam.Client
}

// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
error) {
    var policies []types.Policy
    result, err := wrapper.IamClient.ListPolicies(context.TODO(),
&iam.ListPoliciesInput{
    MaxItems: aws.Int32(maxPolicies),
})
    if err != nil {
        log.Printf("Couldn't list policies. Here's why: %v\n", err)
    } else {
```

```
    policies = result.Policies
  }
  return policies, err
}

// CreatePolicy creates a policy that grants a list of actions to the specified
// resource.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
  var policy *types.Policy
  policyDoc := PolicyDocument{
    Version:  "2012-10-17",
    Statement: []PolicyStatement{{
      Effect: "Allow",
      Action: actions,
      Resource: aws.String(resourceArn),
    }},
  }
  policyBytes, err := json.Marshal(policyDoc)
  if err != nil {
    log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
      resourceArn, err)
    return nil, err
  }
  result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
    &iam.CreatePolicyInput{
      PolicyDocument: aws.String(string(policyBytes)),
      PolicyName:     aws.String(policyName),
    })
  if err != nil {
    log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
  } else {
    policy = result.Policy
  }
  return policy, err
}
```

```
// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
    var policy *types.Policy
    result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}

// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
    _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
    }
    return err
}
```

ロールアクションをラップする構造体を定義します。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    IamClient *iam.Client
}

// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
```

```
var roles []types.Role
result, err := wrapper.IamClient.ListRoles(context.TODO(),
    &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
)
if err != nil {
    log.Printf("Couldn't list roles. Here's why: %v\n", err)
} else {
    roles = result.Roles
}
return roles, err
}

// CreateRole creates a role that trusts a specified user. The trusted user can
// assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
(*types.Role, error) {
    var role *types.Role
    trustPolicy := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Principal: map[string]string{"AWS": trustedUserArn},
            Action: []string{"sts:AssumeRole"},
        }},
    }
    policyBytes, err := json.Marshal(trustPolicy)
    if err != nil {
        log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
            trustedUserArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreateRole(context.TODO(),
    &iam.CreateRoleInput{
        AssumeRolePolicyDocument: aws.String(string(policyBytes)),
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
    }
}
```

```
    } else {
        role = result.Role
    }
    return role, err
}

// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.GetRole(context.TODO(),
        &iam.GetRoleInput{RoleName: aws.String(roleName)})
    if err != nil {
        log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// CreateServiceLinkedRole creates a service-linked role that is owned by the
// specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
    description string) (*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
        &iam.CreateServiceLinkedRoleInput{
            AWSServiceName: aws.String(serviceName),
            Description:    aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
            serviceName, err)
    } else {
        role = result.Role
    }
    return role, err
}
```

```
// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
        &iam.DeleteServiceLinkedRoleInput{
            RoleName: aws.String(roleName)},
    )
    if err != nil {
        log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
            roleName, err)
    }
    return err
}

// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
    error {
    _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
        &iam.AttachRolePolicyInput{
            PolicyArn: aws.String(policyArn),
            RoleName: aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
            roleName, err)
    }
    return err
}

// ListAttachedRolePolicies lists the policies that are attached to the specified
    role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
    ([]types.AttachedPolicy, error) {
    var policies []types.AttachedPolicy
    result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
        &iam.ListAttachedRolePoliciesInput{
            RoleName: aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
            roleName, err)
    }
}
```

```
} else {
    policies = result.AttachedPolicies
}
return policies, err
}

// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
    error {
    _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
        &iam.DetachRolePolicyInput{
            PolicyArn: aws.String(policyArn),
            RoleName:  aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
            err)
    }
    return err
}

// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
        &iam.ListRolePoliciesInput{
            RoleName: aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
            err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}

// DeleteRole deletes a role. All attached policies must be detached before a
```

```
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
    }
    return err
}
```

ユーザーアクションをラップする構造体を定義します。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    IamClient *iam.Client
}

// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
    var users []types.User
    result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
        MaxItems: aws.Int32(maxUsers),
    })
    if err != nil {
        log.Printf("Couldn't list users. Here's why: %v\n", err)
    } else {
        users = result.Users
    }
    return users, err
}

// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
```



```
var user *types.User
result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
    UserName: aws.String(userName),
})
if err != nil {
    var apiError smithy.APIError
    if errors.As(err, &apiError) {
        switch apiError.(type) {
        case *types.NoSuchEntityException:
            log.Printf("User %v does not exist.\n", userName)
            err = nil
        default:
            log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
        }
    }
} else {
    user = result.User
}
return user, err
}

// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.CreateUser(context.TODO(),
        &iam.CreateUserInput{
            UserName: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    } else {
        user = result.User
    }
    return user, err
}

// CreateUserPolicy adds an inline policy to a user. This example creates a
// policy that
// grants a list of actions on a specified role.
```

```
// PolicyDocument shows how to work with a policy document as a data structure
and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
actions []string,
roleArn string) error {
policyDoc := PolicyDocument{
Version: "2012-10-17",
Statement: []PolicyStatement{{
Effect: "Allow",
Action: actions,
Resource: aws.String(roleArn),
}},
}
policyBytes, err := json.Marshal(policyDoc)
if err != nil {
log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
err)
return err
}
_, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
&iam.PutUserPolicyInput{
PolicyDocument: aws.String(string(policyBytes)),
PolicyName: aws.String(policyName),
UserName: aws.String(userName),
})
if err != nil {
log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
err)
}
return err
}

// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
var policies []string
result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
&iam.ListUserPoliciesInput{
UserName: aws.String(userName),
})
if err != nil {
```

```
    log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
err)
} else {
    policies = result.PolicyNames
}
return policies, err
}

// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
error {
    _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
&iam.DeleteUserPolicyInput{
    PolicyName: aws.String(policyName),
    UserName:   aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
err)
    }
    return err
}

// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
    _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
    UserName: aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
    }
    return err
}

// CreateAccessKeyPair creates an access key for a user. The returned access key
contains
// the ID and secret credentials needed to use the key.
```

```
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
(*types.AccessKey, error) {
    var key *types.AccessKey
    result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
&iam.CreateAccessKeyInput{
    UserName: aws.String(userName)})
    if err != nil {
        log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
userName, err)
    } else {
        key = result.AccessKey
    }
    return key, err
}

// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
_, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
&iam.DeleteAccessKeyInput{
    AccessKeyId: aws.String(keyId),
    UserName:    aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
    }
    return err
}

// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
([]types.AccessKeyMetadata, error) {
    var keys []types.AccessKeyMetadata
    result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
&iam.ListAccessKeysInput{
    UserName: aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
err)
    } else {
```

```
keys = result.AccessKeyMetadata
}
return keys, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
/*
```

To run this Java V2 code example, set up your development environment, including your credentials.

For information, see this documentation topic:

<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html>

This example performs these operations:

1. Creates a user that has no permissions.
2. Creates a role and policy that grants Amazon S3 permissions.
3. Creates a role.
4. Grants the user permissions.
5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service client object with the temporary credentials.
6. Deletes the resources.

\*/

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"s3:*\" " +
        "      ], " +
        "      \"Resource\": \"*\\" " +
        "    } " +
        "  ] " +
        "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = ""

            Usage:
            <username> <policyName> <roleName> <roleSessionName>
            <bucketName>\s
```

```
        Where:
            username - The name of the IAM user to create.\s
            policyName - The name of the policy to create.\s
            roleName - The name of the role to create.\s
            roleSessionName - The name of the session required for the
assumeRole operation.\s
            bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String bucketName = args[4];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IAM example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(" 1. Create the IAM user.");
    User createUser = createIAMUser(iam, userName);

    System.out.println(DASHES);
    userArn = createUser.arn();

    AccessKey myKey = createIAMAccessKey(iam, userName);
    String accessKey = myKey.accessKeyId();
    String secretKey = myKey.secretAccessKey();
    String assumeRolePolicyDocument = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
```

```
        "\Effect\": \"Allow\", \" +
        \"Principal\": {\" +
        \"AWS\": \"\" + userArn + \"\" +
        \"},\" +
        \"Action\": \"sts:AssumeRole\"\" +
        \"}]\" +
        \"}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
System.out.println("The policy " + polArn + " was successfully
created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Creates a role.");
TimeUnit.SECONDS.sleep(30);
String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
System.out.println(roleArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Grants the user permissions.");
attachIAMRolePolicy(iam, roleName, polArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait for 30 secs so the resource is available");
TimeUnit.SECONDS.sleep(30);
System.out.println("5. Gets temporary credentials by assuming the
role.");
System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6 Getting ready to delete the AWS resources");
deleteKey(iam, userName, accessKey);
deleteRole(iam, roleName, polArn);
deleteIAMUser(iam, userName);
```



```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                .userName(response.user().userName())
                .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String
json) {

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " +
response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
```

```
waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
    return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
    String keySecret) {

    // Use the creds of the new IAM user that was created in this code
    example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(StaticCredentialsProvider.create(credentials))
            .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
        retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
                StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                .region(region)
                .build();

        System.out.println("Created a S3Client using temp credentials.");
    }
}
```

```
System.out.println("Listing objects in " + bucketName);
ListObjectsRequest listObjects = ListObjectsRequest.builder()
    .bucket(bucketName)
    .build();

ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.println("The name of the key is " + myValue.key());
    System.out.println("The owner is " + myValue.owner());
}

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteRole(IamClient iam, String roleName, String polArn)
{

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();
```

```
        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String
accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
```

```
DeleteAccessKeyCommand,
DeleteUserCommand,
DeleteRoleCommand,
DeletePolicyCommand,
DetachRolePolicyCommand,
IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`
      );
    }
    return User;
  }
}
```



```
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
  // (AWS STS).
  // It's not best practice to use access keys. For more information, see
  // https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
```

```
        accessKeyId: AccessKeyId,
        secretAccessKey: SecretAccessKey,
    },
});

// Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
// thrown while the user and access keys are still stabilizing.
await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
        return await listBuckets(s3Client);
    } catch (err) {
        if (err instanceof Error && err.name === "InvalidAccessKeyId") {
            throw err;
        }
    }
});

// Retry the create role operation until it succeeds. A MalformedPolicyDocument
// error
// is thrown while the user and access keys are still stabilizing.
const { Role } = await retry(
    {
        intervalInMs: 2000,
        maxRetries: 60,
    },
    () =>
        iamClient.send(
            new CreateRoleCommand({
                AssumeRolePolicyDocument: JSON.stringify({
                    Version: "2012-10-17",
                    Statement: [
                        {
                            Effect: "Allow",
                            Principal: {
                                // Allow the previously created user to assume this role.
                                AWS: User.Arn,
                            },
                            Action: "sts:AssumeRole",
                        },
                    ],
                }),
                RoleName: roleName,
            }),
        ),
    ),
);
```

```
);

if (!Role) {
  throw new Error("Role not created");
}

// Create a policy that allows the user to list S3 buckets.
const { Policy: listBucketPolicy } = await iamClient.send(
  new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: ["s3:ListAllMyBuckets"],
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
```

```
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
```

```
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      Username: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      Username: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)

- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
suspend fun main(args: Array<String>) {
    val usage = ""
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
    <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
    operation.
```

```
fileLocation - The file location to the JSON required to create the role
(see Readme).
bucketName - The name of the Amazon S3 bucket from which objects are
read.
"""

if (args.size != 6) {
    println(usage)
    exitProcess(1)
}

val userName = args[0]
val policyName = args[1]
val roleName = args[2]
val roleSessionName = args[3]
val fileLocation = args[4]
val bucketName = args[5]

createUser(userName)
println("$userName was successfully created.")

val polArn = createPolicy(policyName)
println("The policy $polArn was successfully created.")

val roleArn = createRole(roleName, fileLocation)
println("$roleArn was successfully created.")
attachRolePolicy(roleName, polArn)

println("*** Wait for 1 MIN so the resource is available.")
delay(60000)
assumeGivenRole(roleArn, roleSessionName, bucketName)

println("*** Getting ready to delete the AWS resources.")
deleteRole(roleName, polArn)
deleteUser(userName)
println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }
}
```

```
IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    val response = iamClient.createUser(request)
    return response.user?.userName
}
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "      \"Effect\": \"Allow\"," +
            "      \"Action\": [" +
            "        \"s3:*\"" +
            "      ]," +
            "      \"Resource\": \"*\"" +
            "    }" +
            "  ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }
}
```



```
    }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
        $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
```

```
) : Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
```

```
        credentialsProvider = staticCredentials
        region = "us-east-1"
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
```

```
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)

- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
namespace IAM\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use IAM\IAMService;

echo("\n");
echo("-----\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("-----\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_{$uuid}");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
```

```
        \"Action\": \"sts:AssumeRole\"
    }]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$(Get-UUID)",
    $assumeRolePolicyDocument);
echo "Created role: $($assumeRoleRole['RoleName'])\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$(Get-UUID)",
    $listAllBucketsPolicyDocument);
echo "Created policy: $($listAllBucketsPolicy['PolicyName'])\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"sts:AssumeRole\",
        \"Resource\": \"${$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$(Get-UUID)",
    $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
```

```
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_{$uuid}",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail!\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)

- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError

def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()

def setup(iam_resource):
```



```
"""
Creates a new user with no permissions.
Creates an access key pair for the user.
Creates a role with a policy that lets the user assume the role.
Creates a policy that allows listing Amazon S3 buckets.
Attaches the policy to the role.
Creates an inline policy for the user that lets the user assume the role.
```

```
    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                        that has permissions to create users, roles, and
policies
                        in the account.
```

```
    :return: The newly created user, user key, and role.
```

```
"""
try:
    user = iam_resource.create_user(UserName=f"demo-user-{{uuid4()}}")
    print(f"Created user {user.name}.")
except ClientError as error:
    print(
        f"Couldn't create a user for the demo. Here's why: "
        f"{{error.response['Error']['Message']}}")
    )
    raise

try:
    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")
except ClientError as error:
    print(
        f"Couldn't create access keys for user {user.name}. Here's why: "
        f"{{error.response['Error']['Message']}}")
    )
    raise

print(f"Wait for user to be ready.", end="")
progress_bar(10)

try:
    role = iam_resource.create_role(
        RoleName=f"demo-role-{{uuid4()}}",
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
```

```
        "Statement": [  
            {  
                "Effect": "Allow",  
                "Principal": {"AWS": user.arn},  
                "Action": "sts:AssumeRole",  
            }  
        ],  
    },  
),  
print(f"Created role {role.name}.")  
except ClientError as error:  
    print(  
        f"Couldn't create a role for the demo. Here's why: "  
        f"{error.response['Error']['Message']}"  
    )  
    raise  
  
try:  
    policy = iam_resource.create_policy(  
        PolicyName=f"demo-policy-{uuid4()}",  
        PolicyDocument=json.dumps(  
            {  
                "Version": "2012-10-17",  
                "Statement": [  
                    {  
                        "Effect": "Allow",  
                        "Action": "s3:ListAllMyBuckets",  
                        "Resource": "arn:aws:s3:::*"  
                    }  
                ],  
            }  
        ),  
    )  
    role.attach_policy(PolicyArn=policy.arn)  
    print(f"Created policy {policy.policy_name} and attached it to the  
role.")  
except ClientError as error:  
    print(  
        f"Couldn't create a policy and attach it to role {role.name}. Here's  
why: "  
        f"{error.response['Error']['Message']}"  
    )  
    raise
```

```
try:
    user.create_policy(
        PolicyName=f"demo-user-policy-{uuid4()}",
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume
"
        f"the role."
    )
except ClientError as error:
    print(
        f"Couldn't create an inline policy for user {user.name}. Here's why:
"
        f"{error.response['Error']['Message']}"
    )
    raise

print("Give AWS time to propagate these new resources and connections.",
end="")
progress_bar(10)

return user, user_key, role

def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
        have permission to list buckets in the account.
    """
```

```
print(f"Try to list buckets without first assuming the role.")
s3_denied_resource = boto3.resource(
    "s3", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
)
try:
    for bucket in s3_denied_resource.buckets.all():
        print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
buckets!")
except ClientError as error:
    if error.response["Error"]["Code"] == "AccessDenied":
        print("Attempt to list buckets with no permissions: AccessDenied.")
    else:
        raise

def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
account.
    Uses the temporary credentials from the role to list the buckets that are
owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
```

```
        f"{error.response['Error']['Message']}"
    )
    raise

# Create an S3 resource that can access the account with the temporary
credentials.
s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)
print(f"Listing buckets for the assumed role's account:")
try:
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
except ClientError as error:
    print(
        f"Couldn't list buckets for the account. Here's why: "
        f"{error.response['Error']['Message']}"
    )
    raise

def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
```

```
        f"{error.response['Error']['Message']}"
    )
    raise

try:
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    user.delete()
    print(f"Deleted {user.name}.")
except ClientError as error:
    print(
        "Couldn't delete user policy or delete user. Here's why: "
        f"{error.response['Error']['Message']}"
    )

def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
"AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")

if __name__ == "__main__":
    usage_demo()
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
```

```
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
end

# Waits for the specified number of seconds.
#
# @param duration [Integer] The number of seconds to wait.
def wait(duration)
  puts("Give AWS time to propagate resources...")
  sleep(duration)
end

# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_client.create_user(user_name: user_name).user
  @logger.info("Created demo user named #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Tried and failed to create demo user.")
  @logger.info("\t#{e.code}: #{e.message}")
  @logger.info("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
  @logger.info("Created accesskey pair for user #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create access keys for user #{user.user_name}.")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end
```



```
# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
  trust_policy = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Principal: {'AWS': user.arn},
      Action: "sts:AssumeRole"
    }]
  }.to_json
  role = @iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: trust_policy
  ).role
  @logger.info("Created role #{role.role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create a role for the demo. Here's why: ")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  role
end

# Creates a policy that grants permission to list S3 buckets in the account,
and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy_document = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Action: "s3:ListAllMyBuckets",
      Resource: "arn:aws:s3:::*"
    }]
  }
```

```
}.to_json
policy = @iam_client.create_policy(
  policy_name: policy_name,
  policy_document: policy_document
).policy
@iam_client.attach_role_policy(
  role_name: role.role_name,
  policy_arn: policy.arn
)
@logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
end

# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
  policy_document = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Action: "sts:AssumeRole",
      Resource: role.arn
    }]
  }.to_json
  @iam_client.put_user_policy(
    user_name: user.user_name,
    policy_name: policy_name,
    policy_document: policy_document
  )
  puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
```

```
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated
  # into a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
  # resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able
  # to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == "AccessDenied"
      puts("Attempt to list buckets with no permissions: AccessDenied.")
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end
end

# Creates an AWS Security Token Service (AWS STS) client with specified
# credentials.
# This is separated into a factory function so that it can be mocked for unit
# testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
# client.
```

```
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
# are used.
# @param sts_client [AWS::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end

# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
    @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
    @iam_client.delete_policy(policy_arn: policy.policy_arn)
    @logger.info("Detached and deleted policy #{policy.policy_name}.")
  end
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Role deleted: #{role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
end
```

```
# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
  user.each do |key|
    @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end

  @iam_client.delete_user(user_name: user_name)
  @logger.info("Deleted user '#{user_name}'.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting user '#{user_name}': #{e.message}")
end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
user_key.secret_access_key))
```

```
puts("Here are your buckets:")
scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
puts("Deleting role '#{role.role_name}' and attached policies.")
scenario.delete_role(role.role_name)
puts("Deleting user '#{user.user_name}', policies, and keys.")
scenario.delete_user(user.user_name)
puts("Thanks for watching!")
puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
$PROGRAM_NAME == __FILE__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
    =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
```

```
let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

let shared_config =
aws_config::from_env().region(region_provider).load().await;
let client = iamClient::new(&shared_config);
let uuid = Uuid::new_v4().to_string();

let list_all_buckets_policy_document = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3::*\"}]
}"
.to_string();
let inline_policy_document = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"sts:AssumeRole\",
        \"Resource\": \"{}\"}]
}"
.to_string();

(
    client,
    uuid,
    list_all_buckets_policy_document,
    inline_policy_document,
)
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}",
"iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;
```



```
let assume_role_policy_document = "{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Principal\": {\"AWS\": \"{}\"},
    \"Action\": \"sts:AssumeRole\"
  }]
}"
.to_string()
.replace("{}", user.arn());

let assume_role_role = iam_service::create_role(
  &client,
  &format!("{}", "iam_demo_role_", uuid),
  &assume_role_policy_document,
)
.await?;
println!("Created the role with the ARN: {}", assume_role_role.arn());

let list_all_buckets_policy = iam_service::create_policy(
  &client,
  &format!("{}", "iam_demo_policy_", uuid),
  &list_all_buckets_policy_document,
)
.await?;
println!(
  "Created policy: {}",
  list_all_buckets_policy.policy_name.as_ref().unwrap()
);

let attach_role_policy_result =
  iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
  .await?;
println!(
  "Attached the policy to the role: {:?}",
  attach_role_policy_result
);

let inline_policy_name = format!("{}", "iam_demo_inline_policy_", uuid);
let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
```

```
        .await?;
println!("Created inline policy.");

//First, fail to list the buckets with the user.
let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
let fail_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
println!("Fail config: {:?}", fail_config);
let fail_client: s3Client = s3Client::new(&fail_config);
match fail_client.list_buckets().send().await {
    Ok(e) => {
        println!("This should not run. {:?}", e);
    }
    Err(e) => {
        println!("Successfully failed with error: {:?}", e)
    }
}

let sts_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
let sts_client: stsClient = stsClient::new(&sts_config);
sleep(Duration::from_secs(10)).await;
let assumed_role = sts_client
    .assume_role()
    .role_arn(assume_role_role.arn())
    .role_session_name(&format!("{}", "iam_demo_assumerole_session_",
uuid))
    .send()
    .await;
println!("Assumed role: {:?}", assumed_role);
sleep(Duration::from_secs(10)).await;

let assumed_credentials = iamCredentials::from_keys(
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
```

```
        .access_key_id(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .secret_access_key(),
    Some(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .session_token
            .clone(),
    ),
);

let succeed_config = aws_config::from_env()
    .credentials_provider(assumed_credentials)
    .load()
    .await;
println!("succeed config: {:?}", succeed_config);
let succeed_client: s3Client = s3Client::new(&succeed_config);
sleep(Duration::from_secs(10)).await;
match succeed_client.list_buckets().send().await {
    Ok(_) => {
        println!("This should now run successfully.")
    }
    Err(e) => {
        println!("This should not run. {:?}", e);
        panic!()
    }
}

//Clean up.
iam_service::detach_role_policy(
    &client,
    assume_role_role.role_name(),
    list_all_buckets_policy.arn().unwrap_or_default(),
)
.await?;
```

```
iam_service::delete_policy(&client, list_all_buckets_policy).await?;
iam_service::delete_role(&client, &assume_role_role).await?;
println!("Deleted role {}", assume_role_role.role_name());
iam_service::delete_access_key(&client, &user, &key).await?;
println!("Deleted key for {}", key.user_name());
iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
println!("Deleted inline user policy: {}", inline_policy_name);
iam_service::delete_user(&client, &user).await?;
println!("Deleted user {}", user.user_name());

Ok(())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する

Amazon EC2 インスタンスで実行されるアプリケーションはその AWS API リクエストに AWS 認証情報を含める必要があります。デベロッパーは Amazon EC2 インスタンス内に直接 AWS 認証情報を保存し、そのインスタンス内のアプリケーションに対してそれらの認証情報の使用を許可できま

す。しかし、デベロッパーは認証情報を管理し、その更新時には各インスタンスに安全に渡して、各 Amazon EC2 インスタンスを更新する必要があります。これは、かなりの追加作業です。

代わりに、IAM ロールを使用すると、Amazon EC2 インスタンスで実行されるアプリケーションに対して一時的な認証情報を管理するだけで済みます。ロールを使用する場合、長期認証情報 (サインイン認証情報、アクセスキーなど) を Amazon EC2 インスタンスに配布する必要はありません。代わりに、ロールは、アプリケーションが他の AWS リソースへの呼び出しを行うときに使用できる一時的なアクセス権限を提供します。Amazon EC2 インスタンスを起動するときに、そのインスタンスに関連付ける IAM ロールを指定します。そのインスタンスで実行されるアプリケーションは、そのロールから提供される一時的な認証情報を使用して API リクエストに署名できます。

ロールを使用して、Amazon EC2 インスタンスで実行されるアプリケーションに許可を付与するには、いくつかの追加設定が必要です。Amazon EC2 インスタンスで実行されるアプリケーションは、仮想化されたオペレーティングシステムによって AWS から抽象化されます。この追加の分離のため、AWS ロールとその関連付けられた許可を Amazon EC2 インスタンスに割り当て、アプリケーションに対してそれらの使用を許可するための追加の手順が必要になります。この別手順は、インスタンスにアタッチされる [インスタンスプロファイル](#) の作成です。インスタンスプロファイルは、ロールを含んでおり、インスタンスで実行されるアプリケーションにロールの一時的な認証情報を提供できます。それらの一時的な認証情報は、アプリケーションの API コールで、リソースへのアクセスを許可するために、または、ロールで指定されたリソースのみにアクセスを制限するために使用できます。

#### Note

同時に Amazon EC2 インスタンスに割り当てることができるのは 1 つのロールだけです。インスタンスのすべてのアプリケーションは、同じロールとアクセス許可を共有します。Amazon ECS を利用して Amazon EC2 インスタンスを管理する場合、Amazon ECS タスクが実行されている Amazon EC2 インスタンスのロールと区別できるロールを Amazon ECS タスクに割り当てることができます。各タスクにロールを割り当てることは、最小権限アクセスの原則に沿っており、アクションとリソースをよりきめ細かく制御できます。詳細については、Amazon Elastic Container Service ベストプラクティスガイドの「[Amazon ECS のタスクによる IAM ロールの使用](#)」を参照してください。

この方法によるロールの使用には、いくつかの利点があります。ロールの認証情報は一時的なもので、自動的に更新されるため、開発者は認証情報を管理する必要がなく、長期のセキュリティリスクを心配する必要もありません。また、複数のインスタンスに 1 つのロールを使用する場合、その 1 つのロールに変更を加えるとその変更がすべてのインスタンスに自動的に反映されます。

**Note**

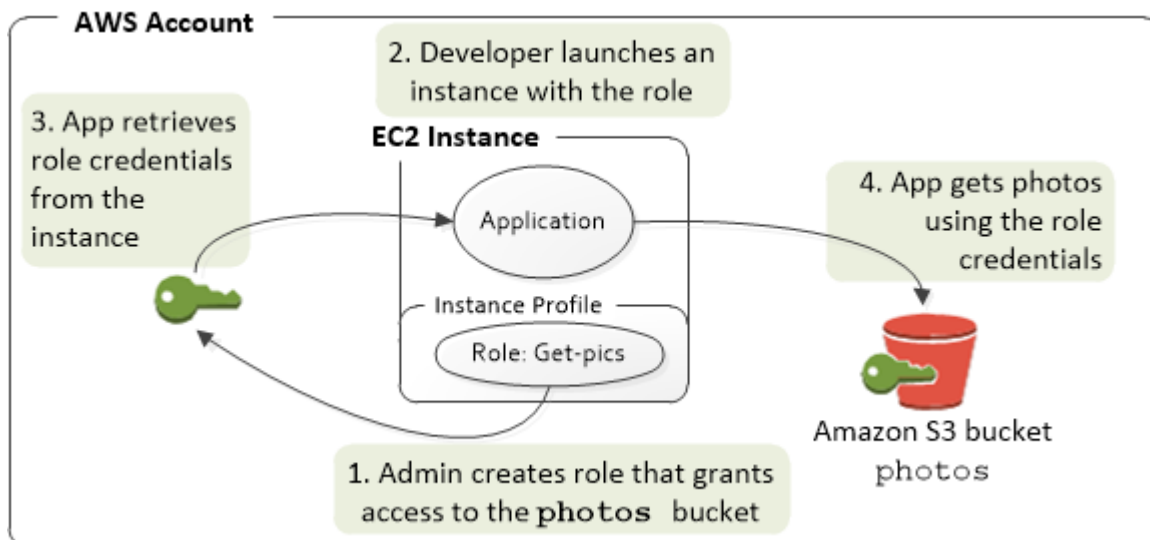
ロールは通常、起動時に Amazon EC2 インスタンスに割り当てられますが、現在実行中の Amazon EC2 インスタンスにロールをアタッチすることもできます。実行中のインスタンスにロールをアタッチする方法については、「[Amazon EC2 の IAM ロール](#)」を参照してください。

## トピック

- [Amazon EC2 インスタンスのロールの仕組み](#)
- [Amazon EC2 でのロールの使用に必要なアクセス許可](#)
- [使用を開始するには](#)
- [関連情報](#)
- [インスタンスプロファイルの使用](#)

## Amazon EC2 インスタンスのロールの仕組み

以下の図では、デベロッパーは Amazon EC2 インスタンスでアプリケーションを実行しており、そのアプリケーションは photos という名前の S3 バケットにアクセスする必要があります。管理者は、Get-pics サービスロールを作成し、Amazon EC2 インスタンスにロールをアタッチします。ロールには、指定された S3 バケットへの読み取り専用アクセスを付与するアクセス許可ポリシーが含まれています。また、Amazon EC2 インスタンスがロールを引き受けて一時的な認証情報を取得することを許可する信頼ポリシーも含まれています。アプリケーションはインスタンスで実行されると、ロールの一時的な認証情報を使用して写真バケットにアクセスできます。管理者はデベロッパーに写真バケットにアクセスする権限を与える必要はなく、デベロッパーが認証情報を共有または管理する必要はありません。



1. 管理者は、**Get-pics** ロールの作成に IAM を使用します。ロールの信頼ポリシーでは、Amazon EC2 インスタンスのみがロールを引き受けることができることを指定します。ロールのアクセス許可ポリシーでは、photos バケットに対する読み取り専用アクセス許可を指定します。
2. デベロッパーは Amazon EC2 インスタンスを起動し、Get-pics ロールをそのインスタンスに割り当てます。

### Note

IAM コンソールを使用する場合、インスタンスプロファイルは自動的に管理され、ほとんど透過的です。ただし、AWS CLI または API を使用してロールと Amazon EC2 インスタンスを作成および管理する場合、インスタンスプロファイルを作成し、別の手順でそのプロファイルにロールを割り当てる必要があります。次に、インスタンスを起動するときに、ロール名ではなくインスタンスのプロファイル名を指定する必要があります。

3. 「[インスタンスメタデータからのセキュリティ認証情報の取得](#)」で説明されているように、アプリケーションは実行時に Amazon EC2 [インスタンスメタデータ](#) からセキュリティ認証情報を取得します。これらは、ロールを表す [一時的セキュリティ認証情報](#) で、制限された期間の間有効です。一部の [AWS SDK](#) では、デベロッパーは、一時的なセキュリティ認証情報を透過的に管理するプロバイダーを使用できます (認証情報を管理するためにその SDK によってサポートされている機能については、各 AWS SDK のドキュメントを参照してください。)

あるいは、アプリケーションは Amazon EC2 インスタンスのインスタンスメタデータから一時的な認証情報を直接取得することもできます。認証情報とその関連する値はメタデータの `iam/security-credentials/role-name` カテゴリ (この場合は `iam/security-credentials/`

Get-pics) から入手できます。アプリケーションがインスタンスメタデータから認証情報を取得する場合、アプリケーションは認証情報をキャッシュすることができます。

4. 取得された一時的な認証情報を使用して、アプリケーションは写真バケットにアクセスします。**Get-pics** ロールにアタッチされたポリシーのために、アプリケーションには読み取り専用のアクセス許可があります。

インスタンスで使用可能な一時的な認証情報は、失効前に自動的に更新されるため、有効な認証情報のセットを常に使用できます。アプリケーションは、現在の認証情報セットが失効する前に、インスタンスメタデータから新しい認証情報セットを取得するだけで済みます。AWS SDK を使用して認証情報を管理できるため、認証情報を更新するための追加のロジックをアプリケーションに含める必要がありません。たとえば、インスタンスプロファイル認証情報プロバイダーを使用してクライアントをインスタンス化します。ただし、アプリケーションがインスタンスメタデータから一時的な認証情報を取得してキャッシュしている場合、1 時間おき、または少なくとも現在のセットが失効する 15 分前までに、更新された認証情報セットを取得する必要があります。失効時刻は、iam/security-credentials/*role-name* カテゴリに返された情報に含まれます。

## Amazon EC2 でのロールの使用に必要なアクセス許可

ロールを割り当てたインスタンスを起動するために、デベロッパーには Amazon EC2 インスタンスを起動する許可と IAM ロールを割り当てる許可が必要です。

以下のポリシーサンプルは、ロールを使用してインスタンスを起動させるために、ユーザーに AWS Management Console を使用することを許可します。ポリシーにアスタリスク (\*) を含むことで、任意のロールの割り当てとリストにある Amazon EC2 アクションの実行をユーザーに許可します。ListInstanceProfiles アクションでは、AWS アカウント で使用できるすべてのロールの表示をユーザーに許可します。

Example 任意のロールが割り当てられたインスタンスを Amazon EC2 コンソールで起動するアクセス許可をユーザーに付与するポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
```



```
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "ec2.amazonaws.com"
      }
    },
    {
      "Sid": "ListEc2AndListInstanceProfiles",
      "Effect": "Allow",
      "Action": [
        "iam:ListInstanceProfiles",
        "ec2:Describe*",
        "ec2:Search*",
        "ec2:Get*"
      ],
      "Resource": "*"
    }
  ]
}
```

## Amazon EC2 インスタンスに割り当て可能なロールを制限する (PassRole を使用)

PassRole 許可を使用すると、インスタンスの起動時にユーザーが Amazon EC2 インスタンスに割り当てることができるロールを制限できます。これにより、ユーザーに付与されているアクセス許可よりも多くのアクセス許可を持つアプリケーションをユーザーが実行することを回避できます。つまり、ユーザーは、昇格されたアクセス許可を取得することができなくなります。例えば、ユーザー Alice には、Amazon EC2 インスタンスを起動し Amazon S3 バケットを操作する許可のみがあるが、このユーザーが Amazon EC2 インスタンスに割り当てられるロールには、IAM と Amazon DynamoDB を操作する許可があるとします。その場合、Alice はインスタンスを起動してログインし、一時的セキュリティ認証情報を取得して、彼女にはアクセス許可がない IAM または DynamoDB アクションを実行できます。

ユーザーが Amazon EC2 インスタンスに割り当てることができるロールを制限するには、PassRole アクションを許可するポリシーを作成します。その後、Amazon EC2 インスタンスを起動するユーザー (またはそのユーザーが属する IAM グループ) にそのポリシーをアタッチします。ポリシーの Resource の要素に、ユーザーが Amazon EC2 インスタンスに渡すことを許可するロールを記載します。ユーザーがインスタンスを起動し、そのインスタンスにロールを割り当てるとき、Amazon EC2 は、ユーザーがそのロールの割り当てを許可されているかどうかを確認します。もちろん、ユーザーが割り当てることができるロールに想定外のアクセス権限が含まれていないことを確認する必要があります。

**Note**

PassRole は、RunInstances や ListInstanceProfiles とは異なり、API アクションではありません。それは API にロールの ARN がパラメータとして渡されるときに、必ず AWS が確認を行う許可です (またはユーザーの代わりにコンソールがこれを行います)。これにより管理者は、渡すことができるロールと、そのロールを渡すことができるユーザーを制御するのに役立ちます。この場合では、ユーザーが特定のロールを Amazon EC2 インスタンスにアタッチできるようにします。

**Example** 特定のロールが割り当てられた Amazon EC2 インスタンスを起動する許可をユーザーに付与するポリシーの例

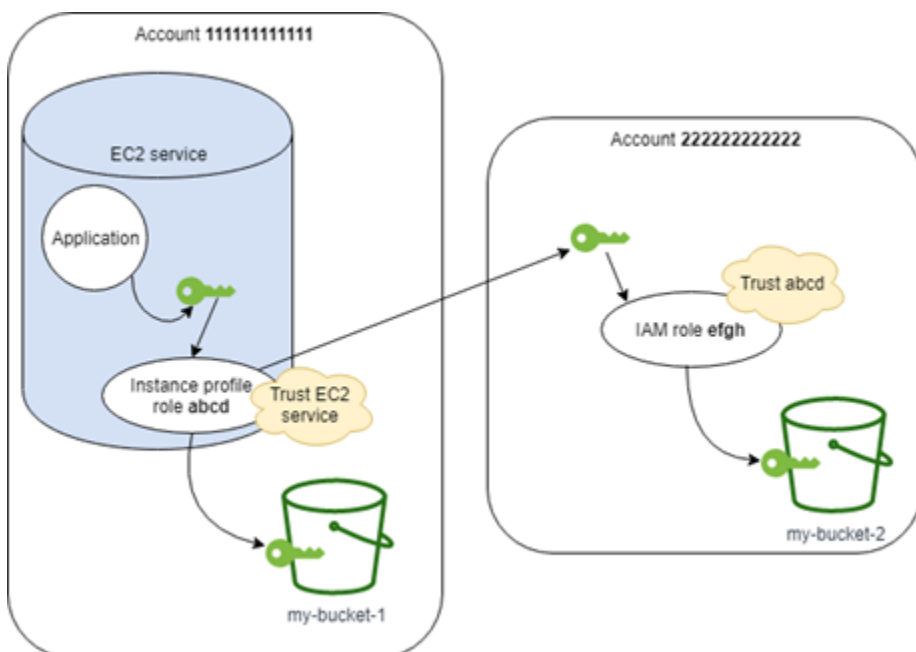
以下のポリシーサンプルは、ロールを使用してインスタンスを起動するために、ユーザーに Amazon EC2 API の使用を許可します。Resource 要素は、ロールの Amazon リソースネーム (ARN) を指定します。ポリシーで ARN を指定することで、Get-pics ロールのみを割り当てるアクセス許可をユーザーに付与します。インスタンスの起動時にユーザーが異なるロールを指定した場合、アクションは失敗します。ユーザーには、ロールを渡したかどうかにかかわらず、すべてのインスタンスを実行する権限があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/Get-pics"
    }
  ]
}
```

## インスタンスプロファイルロールが他のアカウントのロールへの切り替えることを許可する

Amazon EC2 インスタンスで実行されているアプリケーションが別のアカウントでコマンドを実行することを許可できます。これを行うには、最初のアカウントの Amazon EC2 インスタンスロールを 2 番目のアカウントのロールに切り替えることを許可する必要があります。

2 つの AWS アカウント を使用していて Amazon EC2 インスタンスで実行されているアプリケーションが両方のアカウントで [AWS CLI](#) コマンドを実行できるようにするとします。アカウント 111111111111 に Amazon EC2 インスタンスが存在すると仮定します。このインスタンスには、同じ abcd アカウント内の my-bucket-1 バケットで読み取り専用の 111111111111 タスクをアプリケーションが実行する Amazon S3 インスタンスプロファイルのロールが含まれています。ただし、アプリケーションは、アカウント my-bucket-2 の efgh Amazon S3 バケットにアクセスするために 222222222222 クロスアカウントロールを引き受けることも許可されている必要があります。



アプリケーションが my-bucket-1 Amazon S3 バケットにアクセスできるようにするには、abcd Amazon EC2 インスタンスプロファイルロールは次のアクセス許可ポリシーを持っている必要があります。

アカウント 111111111111 **abcd** ロールのアクセス許可ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
```

```

    "Effect": "Allow",
    "Action": [
      "s3:GetBucketLocation",
      "s3:GetAccountPublicAccessBlock",
      "s3:ListAccessPoints",
      "s3:ListAllMyBuckets"
    ],
    "Resource": "arn:aws:s3:::*"
  },
  {
    "Sid": "AllowListAndReadS3ActionOnMyBucket",
    "Effect": "Allow",
    "Action": [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::my-bucket-1/*",
      "arn:aws:s3:::my-bucket-1"
    ]
  },
  {
    "Sid": "AllowIPToAssumeCrossAccountRole",
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::222222222222:role/efgh"
  }
]
}

```

この abcd ロールは、ロールを引き受ける上で Amazon EC2 サービスを信頼する必要があります。これを行うには、abcd ロールは次の信頼ポリシーを持っている必要があります。

アカウント 111111111111 **abcd**ロールの信頼ポリシー

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "abcdTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"Service": "ec2.amazonaws.com"}
    }
  ]
}

```

```

    }
  ]
}

```

efgh クロスアカウントのロールが、同じ my-bucket-2 アカウント内の222222222222 バケットで読み取り専用 Amazon S3 タスクを許可すると仮定します。これを行うには、efgh クロスアカウントのロールは、以下のアクセス許可ポリシーを持っている必要があります。

アカウント 222222222222 **efgh**ロールのアクセス許可ポリシー

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-2/*",
        "arn:aws:s3:::my-bucket-2"
      ]
    }
  ]
}

```

efgh ロールは abcd インスタンスプロファイルのロールがそれを引き受けることを信頼する必要があります。これを行うには、efgh ロールは次の信頼ポリシーを持っている必要があります。

アカウント 222222222222 **efgh** ロールの信頼ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "efghTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
    }
  ]
}
```

## 使用を開始するには

Amazon EC2 インスタンスでロールがどのように機能するかを理解するには、IAM コンソールでロールを作成し、そのロールが割り当てられた Amazon EC2 インスタンスを起動して、実行中のインスタンスを調べる必要があります。[インスタンスのメタデータ](#)を調べて、ロールの一時的な認証情報がインスタンスでどのようにして使用可能になったかを確認できます。また、インスタンスで実行されるアプリケーションがそのロールをどのように使用できるかも確認できます。詳細については、以下のリソースを参照してください。

- 
- SDK ウォークスルー。AWS SDK ドキュメント内のこれらのチュートリアルでは、Amazon EC2 インスタンスで実行されるアプリケーションがロールの一時的な認証情報を使用して Amazon S3 バケットを読み取る手順について説明しています。以下の各ウォークスルーでは、別のプログラミング言語での同様の手順について説明しています。
  - [AWS SDK for Java デベロッパーガイド](#)のJava 用 SDK を使用した Amazon EC2 の IAM ロール の設定
  - 「AWS SDK for .NET デベロッパーガイド」の「[Launch an Amazon EC2 Instance using the SDK for .NET](#)」(SDK for .NET を使用した Amazon EC2 インスタンスの起動)
  - AWS SDK for Ruby デベロッパーガイドの[Ruby 用の SDK for Ruby を使用した Amazon EC2 インスタンスの作成](#)

## 関連情報

ロールの作成または Amazon EC2 インスタンスのロールに関する詳細については、次の情報を参照してください。

- [Amazon EC2 インスタンスでの IAM ロールの使用](#)の詳細については、「Amazon EC2 ユーザーガイド」を参照してください。
- ロールの作成については、「[IAM ロールの作成](#)」を参照してください。
- 一時的なセキュリティ認証情報の使用方法の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。
- IAM API または CLI で作業する場合、IAM インスタンスプロファイルを作成および管理する必要があります。インスタンスプロファイルの詳細については、「[インスタンスプロファイルの使用](#)」を参照してください。
- インスタンスメタデータにあるロールの一時的なセキュリティ認証情報に関する詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスメタデータからのセキュリティ認証情報の取得](#)」を参照してください。

## インスタンスプロファイルの使用

インスタンスプロファイルを使用して、IAM ロールを EC2 インスタンスに渡します。詳細については、[Amazon EC2 ユーザーガイド](#) の「Amazon EC2 の IAM ロール」を参照してください。

## インスタンスプロファイルの管理 (コンソール)

AWS Management Console を使用して Amazon EC2 のロールを作成する場合、コンソールはインスタンスプロファイルを自動的に作成し、そのインスタンスプロファイルにロールと同じ名前を付けます。IAM ロールを使用してインスタンスを起動するのに Amazon EC2 コンソールを使用する場合、インスタンスに関連付けるロールを選択できます。コンソールに表示されるリストは実際にはインスタンスプロファイル名のリストです。コンソールでは、Amazon EC2 に関連付けられていないロールのインスタンスプロファイルは作成されません。

ロールとインスタンスプロファイルの名前が同じである場合は、AWS Management Console を使用して Amazon EC2 の IAM ロールとインスタンスプロファイルを削除できます。インスタンスプロファイルの削除の詳細については、「[ロールまたはインスタンスプロファイルの削除](#)」を参照してください。

## インスタンスプロファイルの管理 (AWS CLI または AWS API)

AWS CLI または AWS API からロールを管理する場合、ロールとインスタンスプロファイルを別個のアクションとして作成します。ロールとインスタンスプロファイルには異なる名前を使用できるため、インスタンスプロファイルの名前とこれらのプロファイルに含まれるロールの名前を知っている必要があります。これにより、EC2 インスタンスを起動するときに適切なインスタンスプロファイルを選択できます。

IAM リソース (インスタンスプロファイルを含む) にタグをアタッチして、それらへのアクセスを特定、整理、制御することができます。インスタンスプロファイルにタグを付けることができるのは、AWS CLI または AWS API を使用する場合があります。

#### Note

インスタンスプロファイルに含めることができる IAM ロールは 1 つのみです。ただし、1 つのロールを複数のインスタンスプロファイルに含めることはできます。このインスタンスプロファイルあたり 1 ロールの制限は、緩和できません。既存のロールを削除してから、別のロールをインスタンスプロファイルに追加することはできます。その後、AWS 全体で変更が反映されるのを待つ必要があります。これは [結果整合性](#) のためです。変更を強制的に実行するには、[インスタンスプロファイル](#) の関連付けを解除してから、[インスタンスプロファイルに関連付ける](#) か、インスタンスを停止してから再起動します。

## インスタンスプロファイルの管理 (AWS CLI)

AWS アカウントでインスタンスプロファイルを使用するには、以下の AWS CLI コマンドを使用できます。

- インスタンスプロファイルを作成する: [aws iam create-instance-profile](#)
- インスタンスプロファイルにタグを付ける: [aws iam tag-instance-profile](#)
- インスタンスプロファイルのタグを一覧表示する: [aws iam list-instance-profile-tags](#)
- インスタンスプロファイルのタグを解除する: [aws iam untag-instance-profile](#)
- ロールをインスタンスプロファイルに追加する: [aws iam add-role-to-instance-profile](#)
- インスタンスプロファイルのリストを取得する: [aws iam list-instance-profiles](#)、[aws iam list-instance-profiles-for-role](#)
- インスタンスプロファイルの情報を取得する: [aws iam get-instance-profile](#)
- ロールをインスタンスプロファイルから削除する: [aws iam remove-role-from-instance-profile](#)
- インスタンスプロファイルを削除する: [aws iam delete-instance-profile](#)

次のコマンドを使用して、既に実行中の EC2 インスタンスにロールをアタッチすることもできます。詳細については、「[Amazon EC2 の IAM ロール](#)」を参照してください。



- ロールが含まれたインスタンスプロファイルを停止中または実行中の EC2 インスタンスにアタッチする: [aws ec2 associate-iam-instance-profile](#)
- EC2 インスタンスにアタッチされたインスタンスプロファイルに関する情報を取得する: [aws ec2 describe-iam-instance-profile-associations](#)
- ロールが含まれたインスタンスプロファイルを停止中または実行中の EC2 インスタンスからデタッチする: [aws ec2 disassociate-iam-instance-profile](#)

## インスタンスプロファイルの管理 (AWS API)

以下の AWS API オペレーションを呼び出して AWS アカウント のインスタンスプロファイルを使用できます。

- インスタンスプロファイルを作成する: [CreateInstanceProfile](#)
- インスタンスプロファイルにタグを付ける: [TagInstanceProfile](#)
- インスタンスプロファイルのタグを一覧表示する: [ListInstanceProfileTags](#)
- インスタンスプロファイルのタグを解除する: [UntagInstanceProfile](#)
- ロールをインスタンスプロファイルに追加する: [AddRoleToInstanceProfile](#)
- インスタンスプロファイルのリストを取得する:  
[ListInstanceProfiles](#)、[ListInstanceProfilesForRole](#)
- インスタンスプロファイルの情報を取得する: [GetInstanceProfile](#)
- ロールをインスタンスプロファイルから削除する: [RemoveRoleFromInstanceProfile](#)
- インスタンスプロファイルを削除する: [DeleteInstanceProfile](#)

以下のオペレーションを呼び出して、既に実行中の EC2 インスタンスにロールをアタッチすることもできます。詳細については、「[Amazon EC2 の IAM ロール](#)」を参照してください。

- ロールが含まれたインスタンスプロファイルを停止中または実行中の EC2 インスタンスにアタッチする: [AssociateIamInstanceProfile](#)
- EC2 インスタンスにアタッチされたインスタンスプロファイルに関する情報を取得する:  
[DescribeIamInstanceProfileAssociations](#)
- ロールが含まれたインスタンスプロファイルを停止中または実行中の EC2 インスタンスからデタッチする: [DisassociateIamInstanceProfile](#)

## IAM ロールの一時的なセキュリティ認証情報の取り消し

### Warning

このページの手順に従うと、ロールを引き受けることによって作成された現在のセッションのすべてのユーザーは、すべての AWS アクションとリソースへのアクセスを拒否されます。その結果、保存されていない作業は失われます。

ユーザーが長いセッションの有効期間 (12 時間など) を使用して AWS Management Console にアクセスできるようにすると、一時的な認証情報がすぐに期限切れになることはありません。ユーザーが意図せずに認証情報を不正なサードパーティーに公開した場合、そのパーティーはセッションの期間アクセスできます。ただし、必要がある場合は、特定の時点より前に発行したロールの認証情報の、すべてのアクセス許可をすぐに取り消しできます。指定された時間より前に発行された、そのロールのすべての一時的な認証情報が無効になります。これにより、すべてのユーザーは新しい認証情報を再認証し、リクエストしなければならなくなります。

### Note

[サービスにリンクされたロール](#)のセッションを取り消すことはできません。

このトピックの手順を使用してロールのアクセス許可を取り消す場合、AWS は、すべてのアクションへのすべてのアクセス許可を拒否するロールに新しいインラインポリシーをアタッチします。このポリシーに含まれる条件によって制限が適用されるのは、アクセス許可が取り消される前にユーザーがロールを引き受けた場合のみです。アクセス許可が取り消された後にユーザーがロールを引き受けた場合、拒否ポリシーはそのユーザーに適用されません。

アクセスの拒否についての詳細は、「[一時的なセキュリティ認証情報のアクセス権限を無効にする](#)」を参照してください。

### Important

この拒否ポリシーは、期間が長いコンソールセッションを使用するユーザーにだけでなく、指定されたロールのすべてのユーザーに適用されます。

## ロールからセッションアクセス許可を取り消すための最小限のアクセス許可

ロールから正常にセッションアクセス許可を取り消すには、ロールの PutRolePolicy アクセス許可が必要です。これにより、AWSRevokeOlderSessions インラインポリシーをロールにアタッチできます。

### セッションアクセス許可のキャンセル

ロールからセッションのアクセス許可を取り消すと、ロールを引き受けたすべてのユーザーに対するすべてのアクセス許可を拒否できます。

#### Note

IAM アイデンティティセンターのアクセス許可セットから作成された IAM のロールを編集することはできません。IAM アイデンティティセンターでユーザーのアクティブなアクセス許可セットセッションを取り消す必要があります。詳細については、「IAM Identity Center ユーザーガイド」の「[Revoke active IAM role sessions created by permission sets](#)」を参照してください。

ロール認証情報のすべての現在のユーザーの、すべてのアクセス許可をすぐに拒否するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [ダッシュボード] のナビゲーションペインで、[ロール] に続いて、アクセス許可を取り消すロールの名前 (チェックボックスではありません) を選択します。
3. 選択したロールの [Summary (概要)] ページで、[セッションの無効化] タブを選択します。
4. [セッションの無効化] タブで、[アクティブなセッションの無効化] を選択します。
5. AWS によってアクションを確認するよう求められます。[I acknowledge that I am revoking all active sessions for this role] (このロールのアクティブなセッションをすべて無効にすることに同意します) のチェックボックスをオンにして、ダイアログボックスの [Revoke active sessions] (アクティブなセッションの無効化) を選択します。

その後、IAM がロールに AWSRevokeOlderSessions という名前のポリシーをアタッチします。[アクティブなセッションの無効化] を選択すると、ポリシーは、過去にロールを引き受けたユーザーに加えて、約 30 秒後にロールを引き受けるユーザーに対するすべてのアクセスを拒否します。この将来の時間の選択では、更新されたポリシーが特定のリージョンで有効になる前に取得または更新された新しいセッションに対処するために、ポリシーの伝播遅延が考慮され

ます。[アクティブなセッションの無効化] を選択した後、約 30 秒が経過した後にロールを引き受けるユーザーは影響を受けません。変更がすぐに表示されとは限らない理由については、「[行った変更がすぐに表示されないことがある](#)」をご参照ください。

### Note

後で再度 [アクティブなセッションの無効化] を選択すると、ポリシーのタイムスタンプが更新され、新しく指定された時間の前にロールを引き受けたユーザーのすべてのアクセス許可が再度拒否されます。

この方法でセッションが取り消された有効なユーザーは、作業を続行するには新しいセッション用の一時的な認証情報を取得する必要があります。この AWS CLI は、期限切れになるまで認証情報をキャッシュします。有効でなくなった、キャッシュされている認証情報の削除と更新を CLI に強制するには、次のいずれかのコマンドを実行します。

Linux、macOS、または Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\aws\cli\cache
```

指定した時間よりも前にセッションのアクセス許可を取り消す

また、AWS CLI または SDK を使用して、任意のタイミングでセッション許可を無効化することで、ポリシーの条件の要素で [aws:TokenIssueTime](#) キーの値を指定することもできます。

このポリシーは、aws:TokenIssueTime の値が指定した日時より前である場合、すべてのアクセス許可を拒否します。aws:TokenIssueTime の値は、一時的なセキュリティ認証情報が作成された正確な時間に相当します。aws:TokenIssueTime の値は、一時的なセキュリティ認証情報を使用して署名された AWS リクエストのコンテキストでのみ存在します。そのため、ポリシーの Deny ステートメントは、IAM ユーザーの長期の認証情報を使用して署名されたリクエストには影響しません。

このポリシーは、ロールにアタッチすることもできます。その場合、このポリシーは、指定した日時より前に、そのロールによって作成された一時的なセキュリティ認証情報のみに影響します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}
    }
  }
}
```

この方法でセッションが取り消された有効なユーザーは、作業を続行するには新しいセッション用の一時的な認証情報を取得する必要があります。この AWS CLI は、期限切れになるまで認証情報をキャッシュします。有効でなくなった、キャッシュされている認証情報の削除と更新を CLI に強制するには、次のいずれかのコマンドを実行します。

Linux、macOS、または Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\aws\cli\cache
```

## IAM ロールの管理

作成したロールを変更、または削除しなければならない場合があります。ロールを変更するには、以下のいずれかを実行します。

- そのロールに関連するすべてのポリシーを変更する
- ロールにアクセスできるユーザーを変更する
- そのロールがユーザーに付与するアクセス許可を編集する
- AWS Management Console、AWS CLI、または API を使用して引き受けたロールの最大セッション期間設定を変更する

また、不要になったロールを削除することもできます。ロールは AWS Management Console、AWS CLI、および API から管理できます。

## トピック

- [ロールの修正](#)
- [ロールまたはインスタンスプロファイルの削除](#)

## ロールの修正

AWS Management Console、AWS CLI、または IAM API を使用して、ロールを変更できます。

## トピック

- [ロールのアクセスの表示](#)
- [アクセス情報に基づくポリシーの生成](#)
- [ロールの修正 \(コンソール\)](#)
- [ロールの修正 \(AWS CLI\)](#)
- [ロールの変更 \(AWS API\)](#)

## ロールのアクセスの表示

ロールのアクセス許可を変更する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## アクセス情報に基づくポリシーの生成

IAM エンティティ (ユーザーまたはロール) に必要な権限を超えるアクセス許可を付与することがあります。付与するアクセス権限を調整するために、エンティティのアクセスアクティビティに基づく IAM ポリシーを生成できます。IAM Access Analyzer は AWS CloudTrail ログを確認し、指定した日付範囲内のロールが使用したアクセス許可を含むポリシーテンプレートを生成します。テンプレートを使用して、きめ細かなアクセス権限で管理ポリシーを作成し、それを IAM エンティティにアタッチできます。これにより、特定のユースケースでロールが AWS リソースとインタラクションするために必要なアクセス権限のみを付与します。詳細については、「[アクセスアクティビティに基づいてポリシーを生成する](#)」を参照してください。

## ロールの修正 (コンソール)

AWS Management Console を使用してロールを変更できます。ロールのタグのセットを変更するには、「[IAM ロールのタグの管理 \(コンソール\)](#)」を参照してください。

## トピック

- [ロールの信頼ポリシーの変更 \(コンソール\)](#)
- [ロールのアクセス許可ポリシーの変更 \(コンソール\)](#)
- [ロールの説明の変更 \(コンソール\)](#)
- [ロールの最大セッション時間の変更 \(コンソール\)](#)
- [ロールのアクセス許可の境界の変更 \(コンソール\)](#)

### ロールの信頼ポリシーの変更 (コンソール)

ロールを引き受けるユーザーを変更するには、ロールの信頼ポリシーを変更する必要があります。[サービスにリンクされたロール](#)の信頼ポリシーは変更できません。

#### メモ

- ユーザーがプリンシパルとしてロールの信頼ポリシーに表示されているが、そのロールを引き受けることができない場合は、ユーザーの[アクセス許可の境界](#)を確認します。アクセス許可の境界がユーザーに対して設定されている場合は、sts:AssumeRole アクションを許可する必要があります。
- ロールセッション内でユーザーが現在のロールを再び引き受けることができるようにするには、ロール信頼ポリシーでロール ARN または AWS アカウント ARN をプリンシパルとして指定します。Amazon EC2、Amazon ECS、Amazon EKS、Lambda などのコンピューティングリソースを提供する AWS のサービスは、一時的な認証情報を提供し、これらの認証情報を自動的に更新します。これにより、常に有効な認証情報セットを確保できます。これらのサービスでは、一時的な認証情報を取得するために現在のロールを再度引き受ける必要はありません。ただし、[セッションタグ](#)または[セッションポリシー](#)を渡す場合は、現在のロールを再度引き受ける必要があります。

### ロールの信頼ポリシーを変更するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。
3. アカウントのロールの一覧で、変更するロールの名前を選択します。

4. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。
5. 必要に応じて信頼ポリシーを編集します。ロールを引き受ける他のプリンシパルを追加するには、Principal 要素で指定します。以下のポリシースニペットの例では、Principal 要素の 2 つの AWS アカウント を参照する方法を示しています。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:root",
    "arn:aws:iam::444455556666:root"
  ]
},
```

別のアカウントでプリンシパルを指定した場合、ロールの信頼ポリシーにアカウントを追加しても、クロスアカウントの信頼関係は半分しか確立されません。デフォルトでは、信頼されたアカウントのユーザーはロールを引き受けることができません。新しく信頼されたアカウントの管理者は、ロールを引き受けるアクセス許可をユーザーに付与する必要があります。これを行うには、ユーザーにアタッチするポリシーを作成または編集し、sts:AssumeRole アクションへのアクセスをユーザーに許可する必要があります。詳細については、次の手順または「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

以下のポリシースニペットでは、AWS 要素で Principal の 2 つのサービスを参照する方法を示します。

```
"Principal": {
  "Service": [
    "opsworks.amazonaws.com",
    "ec2.amazonaws.com"
  ]
},
```

6. 信頼ポリシーの編集を完了したら、[Update policy] (ポリシーの更新) を選択して変更を保存します。

ポリシーの構造や構文の詳細については、「[IAM でのポリシーとアクセス許可](#)」および「[IAM JSON ポリシー要素のリファレンス](#)」を参照してください。



## 信頼された外部アカウントのユーザーにロールの使用を許可するには (コンソール)

この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

1. 信頼された外部 AWS アカウント にサインインします。
2. ユーザーとグループのどちらにアクセス許可をアタッチするかを決定します。IAM コンソールのナビゲーションペインで [Users] (ユーザー) または [User groups] (ユーザーグループ) を適切に選択します。
3. アクセスを許可する対象となるユーザーまたはグループの名前を選択し、[Permissions (アクセス許可)] タブを選択します。
4. 次のいずれかを行います。

- 既存のカスタマー管理ポリシーを編集するには、ポリシーの名前を選択してから [ポリシーの編集] を選択し、[JSON] タブを選択します。AWS の管理ポリシーを編集することはできません。AWS 管理ポリシーには AWS アイコン



が表示されます。AWS 管理ポリシーとカスタマー管理ポリシーの違いの詳細については、「[管理ポリシーとインラインポリシー](#)」を参照してください。

- インラインポリシーを編集するには、ポリシーの名前の横にある矢印を選択してから、[ポリシーの編集] を選択します。
5. ポリシーエディターで、新しい Statement 要素を追加して、次のように指定します。

```
{
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Resource": "arn:aws:iam::ACCOUNT-ID:role/ROLE-NAME"
}
```

ステートメント内の ARN を、ユーザーが引き受けるロールの ARN に置き換えます。

6. 画面のプロンプトに従って、ポリシーの編集を終了します。


## ロールのアクセス許可ポリシーの変更 (コンソール)

ロールで許可されているアクセス許可を変更するには、ロールのアクセス許可のポリシーを修正します。IAM の[サービスにリンクされたロール](#)のアクセス許可ポリシーは変更できません。ロールに依存するサービス内では、アクセス許可ポリシーを変更できる場合があります。サービスでこ

の機能がサポートされているかどうかを確認するには、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-linked roles] (サービスにリンクされたロール) 列が [Yes] (はい) となっているサービスを探します。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

ロールで許可されているアクセス権限を変更するには (コンソール)

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。
3. 変更するロールの名前を選択し、[Permissions (アクセス許可)] タブを選択します。
4. 次のいずれかを行います。
  - 既存のカスタマー管理ポリシーを編集するには、ポリシーの名前を選択してから [ポリシーの編集] を選択します。

 Note

AWS の管理ポリシーを編集することはできません。AWS 管理ポリシーには AWS アイコン



が表示されます。AWS 管理ポリシーとカスタマー管理ポリシーの違いの詳細については、「[管理ポリシーとインラインポリシー](#)」を参照してください。

- 既存の管理ポリシーをロールにアタッチするには、[Add permissions] (アクセス許可を追加) を選択して、[Attach policies] (ポリシーのアタッチ) を選択します。
- 既存のインラインポリシーを編集するには、ポリシーを展開して、[Edit] (編集) を選択します。
- 新しいインラインポリシーを埋め込むには、[Add permissions] (アクセス許可を追加) を選択して、[Create inline policy] (インラインポリシーの作成) を選択します。
- ロールから既存のポリシーを削除するには、ポリシー名の横にあるチェックボックスを選択し、[削除] を選択します。

ロールの説明の変更 (コンソール)

ロールの説明を変更するには、説明テキストを変更します。

## ロールの説明を変更するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。
3. 変更するロールの名前を選択します。
4. [Summary] (概要) セクションで [Edit] (編集) を選択します。
5. ボックスに新しい説明を入力し、[Save changes] (変更の保存) を選択します。

## ロールの最大セッション時間の変更 (コンソール)

コンソール、AWS CLI または AWS API を使用して、引き受けるロールの最大セッション期間設定を指定するには、最大セッション期間設定の値を変更します。この設定の値は 1 時間 ~ 12 時間です。値を指定しない場合、デフォルトの最大 1 時間が適用されます。この設定では、AWS サービスが引き受けるセッションは制限されません。

コンソール、AWS CLI または AWS API を使用して引き受けたロールの最大セッション期間設定を変更するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。
3. 変更するロールの名前を選択します。
4. [Summary] (概要) セクションで [Edit] (編集) を選択します。
5. [Maximum session duration] (最大セッション期間) には、値を選択します。または、[Custom duration] (カスタム期間) を選択して、値 (秒単位) を入力します。
6. [Save changes] (変更の保存) をクリックします。

変更は、次にこのロールが引き受けられるまで有効になりません。このロールの既存のセッションを取り消す方法については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

AWS Management Console の場合、IAM ユーザーセッションはデフォルトで 12 時間です。コンソールでロールを切り替える IAM ユーザーには、ロールに設定された最大セッション期間、またはユーザーのセッションの残り時間のいずれか短い方が付与されます。

AWS CLI または AWS API からロールを引き受けるユーザーは、この最大数まで長いセッションを要求できます。MaxSessionDuration 設定は、リクエストできるロールセッションの最大セッション期間を決定します。

- AWS CLI を使用してセッション期間を指定するには、duration-seconds パラメータを使用します。詳細については、「[IAM ロール \(AWS CLI\) の切り替え](#)」を参照してください。
- AWS API を使用してセッション期間を指定するには、DurationSeconds パラメータを使用します。詳細については、「[IAM ロール \(AWS API\) の切り替え](#)」を参照してください。

## ロールのアクセス許可の境界の変更 (コンソール)

ロールに許可されるアクセス許可の上限を変更するには、ロールの[アクセス許可の境界](#)を変更します。

ロールのアクセス許可の境界を設定するために使用するポリシーを変更するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. 変更する [permissions boundary](#) (許可の境界) を持つロールの名前を選択します。
4. [アクセス許可] タブを選択します。必要に応じて、[Permissions boundary (アクセス許可の境界)] セクションを開き、[Change boundary (境界の変更)] を選択します。
5. アクセス許可の境界として使用するポリシーを選択します。
6. [Change boundary (境界の変更)] を選択します。

変更は、次にこのロールが引き受けられるまで有効になりません。

## ロールの修正 (AWS CLI)

AWS Command Line Interface を使用してロールを変更できます。ロールのタグのセットを変更するには、「[IAM ロールのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

## トピック

- [ロールの信頼ポリシーの変更 \(AWS CLI\)](#)
- [ロールのアクセス許可ポリシーの変更 \(AWS CLI\)](#)
- [ロールの説明の変更 \(AWS CLI\)](#)

- [ロールの最大セッション時間の変更 \(AWS CLI\)](#)
- [ロールのアクセス許可の境界の変更 \(AWS CLI\)](#)

## ロールの信頼ポリシーの変更 (AWS CLI)

ロールを引き受けるユーザーを変更するには、ロールの信頼ポリシーを変更する必要があります。[サービスにリンクされたロール](#)の信頼ポリシーは変更できません。

### メモ

- ユーザーがプリンシパルとしてロールの信頼ポリシーに表示されているが、そのロールを引き受けることができない場合は、ユーザーの[アクセス許可の境界](#)を確認します。アクセス許可の境界がユーザーに対して設定されている場合は、sts:AssumeRole アクションを許可する必要があります。
- ロールセッション内でユーザーが現在のロールを再び引き受けることができるようにするには、ロール信頼ポリシーでロール ARN または AWS アカウント ARN をプリンシパルとして指定します。Amazon EC2、Amazon ECS、Amazon EKS、Lambda などのコンピューティングリソースを提供する AWS のサービスは、一時的な認証情報を提供し、これらの認証情報を自動的に更新します。これにより、常に有効な認証情報セットを確保できます。これらのサービスでは、一時的な認証情報を取得するために現在のロールを再度引き受ける必要はありません。ただし、[セッションタグ](#)または[セッションポリシー](#)を渡す場合は、現在のロールを再度引き受ける必要があります。ロールの信頼ポリシーを変更してプリンシパルロールの ARN または AWS アカウント ARN を追加する方法については、[ロールの信頼ポリシーの変更 \(コンソール\)](#) を参照してください。

## ロールの信頼ポリシーを変更するには (AWS CLI)

1. (オプション) 変更するロールの名前が不明である場合は、次のコマンドを実行してアカウントのロールを一覧表示します。
  - [aws iam list-roles](#)
2. (オプション) ロールの現在の信頼ポリシーを表示するには、次のコマンドを実行します。
  - [aws iam get-role](#)

3. ロールにアクセス可能な信頼されたプリンシパルを変更するには、更新された信頼ポリシーを使用してテキストファイルを作成します。ポリシーの作成には任意のテキストエディタを使用できます。

以下の信頼ポリシーの例では、Principal 要素で 2 つの AWS アカウントを参照する方法を示します。これにより、2 つの別個の AWS アカウント 内のユーザーが、このロールを引き受けることができます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::111122223333:root",
      "arn:aws:iam::444455556666:root"
    ]},
    "Action": "sts:AssumeRole"
  }
}
```

別のアカウントでプリンシパルを指定した場合、ロールの信頼ポリシーにアカウントを追加しても、クロスアカウントの信頼関係は半分しか確立されません。デフォルトでは、信頼されたアカウントのユーザーはロールを引き受けることができません。新しく信頼されたアカウントの管理者は、ロールを引き受けるアクセス許可をユーザーに付与する必要があります。これを行うには、ユーザーにアタッチするポリシーを作成または編集し、sts:AssumeRole アクションへのアクセスをユーザーに許可する必要があります。詳細については、次の手順または「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

4. 先ほど作成したファイルを使用して信頼ポリシーを更新するには、次のコマンドを実行します。
  - [aws iam update-assume-role-policy](#)

信頼された外部アカウントのユーザーにロールの使用を許可するには (AWS CLI)

この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

1. JSON ファイルを作成し、ロールを引き受けるためのアクセス許可を付与するアクセス許可ポリシーを含めます。たとえば、次のポリシーには必要最小限のアクセス権限が含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
  }
}
```

ステートメント内の ARN を、ユーザーが引き受けるロールの ARN に置き換えます。

2. 次のコマンドを実行し、信頼ポリシーが含まれている JSON ファイルを IAM にアップロードします。

- [aws iam create-policy](#)

このコマンドの出力には、ポリシーの ARN が含まれています。この ARN を書き留めます。後のステップで必要になります。

3. ポリシーをアタッチするユーザーまたはグループを決定します。目的のユーザーやグループの名前が不明である場合は、以下のいずれかのコマンドを使用して、アカウントのユーザーやグループを一覧表示します。

- [aws iam list-users](#)
- [aws iam list-groups](#)

4. 以下のいずれかのコマンドを使用して、前のステップで作成したポリシーをユーザーまたはグループにアタッチします。

- [aws iam attach-user-policy](#)
- [aws iam attach-group-policy](#)

## ロールのアクセス許可ポリシーの変更 (AWS CLI)

ロールで許可されているアクセス許可を変更するには、ロールのアクセス許可のポリシーを修正します。IAM の[サービスにリンクされたロール](#)のアクセス許可ポリシーは変更できません。ロールに依存するサービス内では、アクセス許可ポリシーを変更できる場合があります。サービスでこの機能がサポートされているかどうかを確認するには、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-linked roles] (サービスにリンクされたロール) 列が [Yes] (はい) となっているサー

ビスを探します。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

ロールで許可されたアクセス許可を変更するには (AWS CLI)

1. (オプション) ロールに現在関連付けられているアクセス許可を表示するには、以下のコマンドを実行します。
  1. [aws iam list-role-policies](#) (インラインポリシーの一覧表示)
  2. [aws iam list-attached-role-policies](#) (管理ポリシーの一覧表示)
2. ロールのアクセス権限を更新するコマンドは、管理ポリシーとインラインポリシーのいずれを更新するかによって異なります。

管理ポリシーを更新するには、次のコマンドを実行して新しいバージョンの管理ポリシーを作成します。

- [aws iam create-policy-version](#)

インラインポリシーを更新するには、次のコマンドを実行します。

- [aws iam put-role-policy](#)

ロールの説明の変更 (AWS CLI)

ロールの説明を変更するには、説明テキストを変更します。

ロールの説明を変更するには (AWS CLI)

1. (オプション) ロールの現在の説明を表示するには、次のコマンドを実行します。
  - [aws iam get-role](#)
2. ロールの説明を更新するには、説明パラメータを指定して、次のコマンドを実行します。
  - [aws iam update-role](#)

ロールの最大セッション時間の変更 (AWS CLI)

AWS CLI または API を使用して、引き受けるロールの最大セッション期間設定を指定するには、最大セッション期間設定の値を変更します。この設定の値は 1 時間 ~ 12 時間です。値を指定しない場



合、デフォルトの最大 1 時間が適用されます。この設定では、AWS サービスが引き受けるセッションは制限されません。

#### Note

AWS CLI または API からロールを引き受けると、`duration-seconds` CLI パラメータまたは `DurationSeconds` API パラメータを使用して、より長いロールセッションをリクエストできます。MaxSessionDuration 設定は、DurationSeconds パラメータを使用してリクエストできるロールセッションの最大セッション期間を決定します。DurationSeconds パラメータの値を指定しない場合、セキュリティ認証情報は 1 時間有効です。

引き受けたロールの最大セッション継続時間設定を AWS CLI で変更するには (AWS CLI)

1. (オプション) ロールの現在の最大セッション継続時間設定を表示するには、次のコマンドを実行します。
  - [aws iam get-role](#)
2. ロールの最大セッション継続時間設定を更新するには、`max-session-duration` CLI パラメータまたは `MaxSessionDuration` API パラメータを指定して、次のコマンドを実行します。
  - [aws iam update-role](#)

変更は、次にこのロールが引き受けられるまで有効になりません。このロールの既存のセッションを取り消す方法については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

ロールのアクセス許可の境界の変更 (AWS CLI)

ロールに許可されるアクセス許可の上限を変更するには、ロールの[アクセス許可の境界](#)を変更します。

ロールのアクセス許可の境界を設定するために使用する管理ポリシーを変更するには (AWS CLI)

1. (オプション) ロールの現在の[アクセス許可の境界](#)を表示するには、以下のコマンドを実行します。
  - [aws iam get-role](#)

2. 別の管理ポリシーを使用してロールのアクセス許可の境界を更新するには、次のコマンドを実行します。

- [aws iam put-role-permissions-boundary](#)

ロールにアクセス許可の境界として設定できる管理ポリシーは 1 つのみです。アクセス許可の境界を変更する場合は、ロールに許可されるアクセス許可の上限を変更します。

## ロールの変更 (AWS API)

AWS API を使用してロールを変更できます。ロールのタグのセットを変更するには、「[IAM ロールのタグの管理 \(AWS CLI または AWS API\)](#)」を参照してください。

## トピック

- [ロールの信頼ポリシーの変更 \(AWS API\)](#)
- [ロールのアクセス許可ポリシーの変更 \(AWS API\)](#)
- [ロール説明の変更 \(AWS API\)](#)
- [ロールの最大セッション時間の変更 \(AWS API\)](#)
- [ロールのアクセス許可の境界の変更 \(AWS API\)](#)

## ロールの信頼ポリシーの変更 (AWS API)

ロールを引き受けるユーザーを変更するには、ロールの信頼ポリシーを変更する必要があります。[サービスにリンクされたロール](#)の信頼ポリシーは変更できません。

### メモ

- ユーザーがプリンシパルとしてロールの信頼ポリシーに表示されているが、そのロールを引き受けることができない場合は、ユーザーの[アクセス許可の境界](#)を確認します。アクセス許可の境界がユーザーに対して設定されている場合は、sts:AssumeRole アクションを許可する必要があります。
- ロールセッション内でユーザーが現在のロールを再び引き受けることができるようにするには、ロール信頼ポリシーでロール ARN または AWS アカウント ARN をプリンシパルとして指定します。Amazon EC2、Amazon ECS、Amazon EKS、Lambda などのコンピューティングリソースを提供する AWS のサービスは、一時的な認証情報を提供し、これらの認証情報を自動的に更新します。これにより、常に有効な認証情報セットを確保

できます。これらのサービスでは、一時的な認証情報を取得するために現在のロールを再度引き受ける必要はありません。ただし、[セッションタグ](#)または[セッションポリシー](#)を渡す場合は、現在のロールを再度引き受ける必要があります。ロールの信頼ポリシーを変更してプリンシパルロールの ARN または AWS アカウント ARN を追加する方法については、[ロールの信頼ポリシーの変更 \(コンソール\)](#) を参照してください。

ロールの信頼ポリシーを変更するには (AWS API)

1. (オプション) 変更するロールの名前が不明な場合は、次のオペレーションを呼び出してアカウントのロールを一覧表示します。
  - [ListRoles](#)
2. (オプション) ロールの現在の信頼ポリシーを表示するには、次のオペレーションを呼び出します。
  - [GetRole](#)
3. ロールにアクセス可能な信頼されたプリンシパルを変更するには、更新された信頼ポリシーを使用してテキストファイルを作成します。ポリシーの作成には任意のテキストエディタを使用できます。

以下の信頼ポリシーの例では、Principal 要素で 2 つの AWS アカウント を参照する方法を示します。これにより、2 つの別個の AWS アカウント 内のユーザーが、このロールを引き受けることができます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::111122223333:root",
      "arn:aws:iam::444455556666:root"
    ]},
    "Action": "sts:AssumeRole"
  }
}
```

別のアカウントでプリンシパルを指定した場合、ロールの信頼ポリシーにアカウントを追加しても、クロスアカウントの信頼関係は半分しか確立されません。デフォルトでは、信頼されたア

カウントのユーザーはロールを引き受けることができません。新しく信頼されたアカウントの管理者は、ロールを引き受けるアクセス許可をユーザーに付与する必要があります。これを行うには、ユーザーにアタッチするポリシーを作成または編集し、`sts:AssumeRole` アクションへのアクセスをユーザーに許可する必要があります。詳細については、次の手順または「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

4. 先ほど作成したファイルを使用して信頼ポリシーを更新するには、次のオペレーションを呼び出します。

- [UpdateAssumeRolePolicy](#)

信頼された外部アカウントのユーザーにロールの使用を許可するには (AWS API)

この手順の詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

1. JSON ファイルを作成し、ロールを引き受けるためのアクセス許可を付与するアクセス許可ポリシーを含めます。たとえば、次のポリシーには必要最小限のアクセス権限が含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
  }
}
```

ステートメント内の ARN を、ユーザーが引き受けるロールの ARN に置き換えます。

2. 次のオペレーションを呼び出し、信頼ポリシーが含まれている JSON ファイルを IAM にアップロードします。

- [CreatePolicy](#)

このオペレーションの出力には、ポリシーの ARN が含まれています。この ARN を書き留めます。後のステップで必要になります。

3. ポリシーをアタッチするユーザーまたはグループを決定します。目的のユーザーやグループの名前が不明である場合は、以下のいずれかのオペレーションを呼び出して、アカウントのユーザーやグループを一覧表示します。

- [ListUsers](#)
  - [ListGroups](#)
4. 以下のいずれかのオペレーションを呼び出して、前のステップで作成したポリシーをユーザーまたはグループにアタッチします。
- API: [AttachUserPolicy](#)
  - [AttachGroupPolicy](#)

## ロールのアクセス許可ポリシーの変更 (AWS API)

ロールで許可されているアクセス許可を変更するには、ロールのアクセス許可のポリシーを修正します。IAM の [サービスにリンクされたロール](#) のアクセス許可ポリシーは変更できません。ロールに依存するサービス内では、アクセス許可ポリシーを変更できる場合があります。サービスでこの機能がサポートされているかどうかを確認するには、「[IAM と連携する AWS のサービス](#)」を参照し、[Service-linked roles] (サービスにリンクされたロール) 列が [Yes] (はい) となっているサービスを探します。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

## ロールで許可されたアクセス許可を変更するには (AWS API)

1. (オプション) ロールに現在関連付けられているアクセス許可を表示するには、以下のオペレーションを呼び出します。
  1. [ListRolePolicies](#) (インラインポリシーの一覧表示)
  2. [ListAttachedRolePolicies](#) (管理ポリシーの一覧表示)
2. ロールのアクセス許可を更新するオペレーションは、管理ポリシーとインラインポリシーのいずれを更新するかによって異なります。

管理ポリシーを更新するには、次のオペレーションを呼び出して新しいバージョンの管理ポリシーを作成します。

- [CreatePolicyVersion](#)

インラインポリシーを更新するには、次のオペレーションを呼び出します。

- [PutRolePolicy](#)

## ロール説明の変更 (AWS API)

ロールの説明を変更するには、説明テキストを変更します。

ロールの説明を変更するには (AWS API)

1. (オプション) ロールの現在の説明を表示するには、次のオペレーションを呼び出します。
  - [GetRole](#)
2. ロールの説明を更新するには、説明パラメータを指定して、次のオペレーションを呼び出します。
  - [UpdateRole](#)

## ロールの最大セッション時間の変更 (AWS API)

AWS CLI または API を使用して、引き受けるロールの最大セッション期間設定を指定するには、最大セッション期間設定の値を変更します。この設定の値は 1 時間 ~ 12 時間です。値を指定しない場合、デフォルトの最大 1 時間が適用されます。この設定では、AWS サービスが引き受けるセッションは制限されません。

### Note

AWS CLI または API からロールを引き受けると、`duration-seconds` CLI パラメータまたは `DurationSeconds` API パラメータを使用して、より長いロールセッションをリクエストできます。MaxSessionDuration 設定は、DurationSeconds パラメータを使用してリクエストできるロールセッションの最大セッション期間を決定します。DurationSeconds パラメータの値を指定しない場合、セキュリティ認証情報は 1 時間有効です。

引き受けたロールの最大セッション継続時間設定を API で変更するには (AWS API)

1. (オプション) ロールの現在の最大セッション継続時間設定を表示するには、次のオペレーションを呼び出します。
  - [GetRole](#)
2. ロールの最大セッション継続時間設定を更新するには、`max-sessionduration` CLI パラメータまたは `MaxSessionDuration` API パラメータを指定して、次のオペレーションを呼び出します。

- [UpdateRole](#)

変更は、次にこのロールが引き受けられるまで有効になりません。このロールの既存のセッションを取り消す方法については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

## ロールのアクセス許可の境界の変更 (AWS API)

ロールに許可されるアクセス許可の上限を変更するには、ロールの[アクセス許可の境界](#)を変更します。

ロールのアクセス許可の境界を設定するために使用する管理ポリシーを変更するには (AWS API)

1. (オプション) ロールの現在の[アクセス許可の境界](#)を表示するには、以下のオペレーションを呼び出します。

- [GetRole](#)

2. 別の管理ポリシーを使用してロールのアクセス許可の境界を更新するには、次のオペレーションを呼び出します。

- [PutRolePermissionsBoundary](#)

ロールにアクセス許可の境界として設定できる管理ポリシーは 1 つのみです。アクセス許可の境界を変更する場合は、ロールに許可されるアクセス許可の上限を変更します。

## ロールまたはインスタンスプロファイルの削除

ロールが不要になった場合は、ロールとその関連する権限を削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。

ロールが EC2 インスタンスに関連付けられている場合、インスタンスプロファイルからロールを削除した後、インスタンスプロファイルを削除することもできます。

**⚠ Warning**

削除しようとしているロールまたはインスタンスプロファイルで実行されている Amazon EC2 インスタンスがないことを確認してください。実行中のインスタンスに関連付けられているロールまたはインスタンスプロファイルを削除すると、そのインスタンスで実行されているすべてのアプリケーションが中断されます。

ロールを完全に削除しない場合は、ロールを無効にできます。そのためには、ロールのポリシーを変更してから、現在のすべてのセッションを取り消します。例えば、ロールに、すべての AWS へのアクセスを拒否したポリシーを追加できます。ロールを引き受けようとするすべてのユーザーへのアクセスを拒否するように、信頼ポリシーを編集することもできます。セッションの取り消しの詳細については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

**トピック**

- [ロールのアクセスの表示](#)
- [サービスにリンクされたロールの削除](#)
- [IAM ユーザーの削除 \(コンソール\)](#)
- [IAM ロール \(AWS CLI\) の削除](#)
- [IAM ロールの削除 \(AWS API\)](#)
- [関連情報](#)

**ロールのアクセスの表示**

ロールを削除する前に、ロールが最後に使用された日時を確認することをお勧めします。これを行うには、AWS Management Console、AWS CLI、または AWS API を使用します。ロールを使用しているユーザーからアクセスを削除したくないため、この情報を表示する必要があります。

ロールの最後のアクティビティの日付が、[Access Advisor (アクセスアドバイザー)] タブで報告された最後の日付と一致しない場合があります。[\[Access Advisor \(アクセスアドバイザー\)\]](#) タブでは、ロールのアクセス許可ポリシーで許可されているサービスのアクティビティのみがレポートされます。ロールの最後のアクティビティの日付には、AWS でサービスにアクセスしようとした最後の試みが含まれます。



**Note**

ロールの直近のアクティビティと Access Advisor データの追跡期間は、400 日間です。ユーザーのリージョンが昨年内にこれらの機能をサポートし始めた場合、この期間は短くなる可能性があります。このロールは 400 日以上前に使用された可能性があります。追跡期間の詳細については、「[AWS が最終アクセス情報を追跡する場所](#)」を参照してください。

ロールが最後に使用された日時を表示するには ( コンソール )

1. AWS Management Console にサインインして、<https://console.aws.amazon.com/iam/> で IAM コンソールを開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. アクティビティを表示するロールの行を探します。検索フィールドを使用して結果を絞り込むことができます。[Last activity (最後のアクティビティ)] 列を表示して、ロールが最後に使用されてからの日数を確認します。ロールが追跡期間内に使用されていない場合、テーブルには [None (なし)] と表示されます。
4. 詳細情報を表示するには、ロールの名前を選択します。ロールの [Summary (概要)] ページには、ロールが最後に使用された日付を表示する [Last activity (最後のアクティビティ)] も含まれます。ロールが過去 400 日以内に使用されていない場合、[Last activity (最後のアクティビティ)] には [Not accessed in the tracking period (追跡期間中にアクセスされていません)] と表示されます。

ロールが最後に使用された日時を表示するには (AWS CLI)

[aws iam get-role](#) - RoleLastUsed オブジェクトを含むロールに関する情報を返すには、このコマンドを実行します。このオブジェクトには、ロールが最後に使用された LastUsedDate と Region が含まれます。RoleLastUsed が存在しても値が含まれていない場合、ロールは追跡期間内に使用されていません。

ロールが最後に使用された日時を表示するには (AWS API)

[GetRole](#) - このオペレーションを呼び出して、RoleLastUsed オブジェクトを含むロールに関する情報を返します。このオブジェクトには、ロールが最後に使用された LastUsedDate と Region が含まれます。RoleLastUsed が存在しても値が含まれていない場合、ロールは追跡期間内に使用されていません。

## サービスにリンクされたロールの削除

ロールが「[サービスにリンクされたロール](#)」の場合は、リンクされたサービスに関するドキュメントで、ロールを削除する方法を確認します。アカウントのサービスにリンクされたロールを表示するには、コンソールの [IAM ロール] ページに移動します。サービスにリンクされたロールが、テーブルの [Trusted entities] (信頼されたエンティティ) 列の [(Service-linked role)] ((サービスにリンクされたロール)) に表示されます。ロールの [Summary (概要)] ページのバナーにも、そのロールがサービスにリンクされたロールであることが示されています。

サービスにリンクされたロールを削除するためのドキュメントがサービスに含まれていない場合は、IAM コンソール、AWS CLI、または API を使用してロールを削除できます。詳細については、「[サービスにリンクされたロールの削除](#)」を参照してください。

### IAM ユーザーの削除 (コンソール)

AWS Management Console を使用してロールを削除すると、IAM ではロールに関連付けられた管理ポリシーも自動的にデタッチされます。また、ロールに関連したインラインポリシーとロールを含む Amazon EC2 インスタンスプロファイルも自動的に削除されます。

#### Important

場合によっては、ロールは、Amazon EC2 インスタンスプロファイルに関連付いていることがあります。また、ロールとインスタンスプロファイルの名前が同じ場合があります。このような場合は、AWS Management Console を使用して、ロールやインスタンスプロファイルを削除することはできません。この関連付けは、コンソールで作成したロールとインスタンスプロファイルに対して自動的に行われます。AWS CLI、Tools for Windows PowerShell、または AWS API からロールを作成した場合は、ロールとインスタンスプロファイルで名前が異なることがあります。その場合、コンソールを使用してそれらのロールを削除することはできません。代わりに、まず、AWS CLI、Tools for Windows PowerShell、または AWS API を使用して、インスタンスプロファイルからロールを削除する必要があります。その後、ロールを削除する別の手順を実行する必要があります。

### ロールを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ロール] を選択し、削除するロール名の隣にあるチェックボックスをオンにします。

3. ページの上部で、[削除] を選択します。
4. 確認ダイアログボックスで、最終アクセス情報を確認します。これは、選択したそれぞれのロールの AWS サービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。続行する場合は、テキスト入力フィールドにロール名を入力し、削除を選択します。確実に削除する場合は、最終アクセス情報をまだロード中であっても、削除を実行できます。

#### Note

ロールと同じ名前でない限り、コンソールを使用してインスタンスプロファイルを削除することはできません。ロールを削除する過程でインスタンスプロファイルを削除する必要があります。ロールを同時に削除することなくインスタンスプロファイルを削除するには、AWS CLI または AWS API を使用する必要があります。詳細については、次のセクションを参照してください。

## IAM ロール ( AWS CLI) の削除

AWS CLI を使用してロールを削除する場合、最初にそのロールに関連したインラインポリシーを削除しなければなりません。また、ロールに関連付けられた管理ポリシーもデタッチする必要があります。ロールを含む関連付けられたインスタンスプロファイルを削除する場合は、別途、削除する必要があります。

### ロールを削除するには (AWS CLI)

1. 削除するロールの名前が分からない場合、以下のコマンドを入力してお客様のアカウントにあるロールを表示します。

```
aws iam list-roles
```

リストには、各ロールの Amazon リソースネーム (ARN) が含まれます。CLI コマンドでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。

2. ロールが関連付けられたすべてのインスタンスプロファイルからロールを削除します。

- a. そのロールが関連付けられているすべてのインスタンスプロファイルを表示するには、以下のコマンドを入力します。

```
aws iam list-instance-profiles-for-role --role-name role-name
```

- b. 各インスタンスプロファイルで以下のコマンドを入力して、インスタンスプロファイルからロールを削除します。

```
aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
```

3. そのロールに関連するすべてのポリシーを削除します。

- a. ロールに存在するすべてのインラインポリシーを一覧表示するには、以下のコマンドを入力します。

```
aws iam list-role-policies --role-name role-name
```

- b. ロールから各インラインポリシーを削除するには、各ポリシーで以下のコマンドを入力します。

```
aws iam delete-role-policy --role-name role-name --policy-name policy-name
```

- c. ロールにアタッチされたすべてのマネージドポリシーを一覧表示するには、以下のコマンドを入力します。

```
aws iam list-attached-role-policies --role-name role-name
```

- d. ロールから各マネージドポリシーをデタッチするには、各ポリシーで以下のコマンドを入力します。

```
aws iam detach-role-policy --role-name role-name --policy-arn policy-arn
```

4. 次のコマンドを入力してロールを削除します。

```
aws iam delete-role --role-name role-name
```

5. そのロールに関連付けられたインスタンスプロファイルを再利用する予定がない場合、以下のコマンドを入力して削除できます。

```
aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

## IAM ロールの削除 (AWS API)

IAM API を使用してロールを削除する場合、最初にそのロールに関連したインラインポリシーを削除しなければなりません。また、ロールに関連付けられた管理ポリシーもデタッチする必要があります。ロールを含む関連付けられたインスタンスプロファイルを削除する場合は、別途、削除する必要があります。

### ロールを削除するには (AWS API)

1. ロールが関連付けられたすべてのインスタンスプロファイルを一覧表示するには、[ListInstanceProfilesForRole](#) を呼び出します。

インスタンスプロファイルからロールを削除するには、[RemoveRoleFromInstanceProfile](#) を呼び出します。ロール名およびインスタンスプロファイル名を引き渡さなければいけません。

そのロールに関連付けられていたインスタンスプロファイルを再利用しない場合、[DeleteInstanceProfile](#) を呼び出して削除します。

2. ロールのすべてのインラインポリシーを一覧表示するには、[ListRolePolicies](#) を呼び出します。

ロールに関連付けられたインラインポリシーを削除するには、[DeleteRolePolicy](#) を呼び出します。ロール名およびインラインポリシー名を渡す必要があります。

3. ロールにアタッチされたすべてのマネージドポリシーを一覧表示するには、[ListAttachedRolePolicies](#) を呼び出します。

ロールにアタッチされたマネージドポリシーをデタッチするには、[DetachRolePolicy](#) を呼び出します。ロール名およびマネージドポリシー ARN を渡す必要があります。

4. ロールを削除するには、[DeleteRole](#) を呼び出します。

## 関連情報

インスタンスプロファイルの一般的な情報については、[インスタンスプロファイルの使用](#)を参照してください。

サービスにリンクされたロールの一般情報については、「[サービスリンクロールの使用](#)」を参照してください。

## ID プロバイダーとフェデレーション

既にユーザー ID を AWS の外で管理している場合、AWS アカウントに IAM ユーザーを作成する代わりに、ID プロバイダーを利用できます。ID プロバイダー (IdP) を使用すると、AWS の外部のユーザー ID を管理して、これらの外部ユーザー ID にアカウント内の AWS リソースに対するアクセス許可を与えることができます。これは、会社に既に企業ユーザーディレクトリなどの独自の ID システムがある場合に便利です。また、AWS リソースへのアクセスが必要なモバイルアプリやウェブアプリケーションを作成する場合にも便利です。

外部 IdP は、[OpenID Connect \(OIDC\)](#) または [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#) のいずれかを使用して ID 情報を AWS に提供します。OIDC は、GitHub Actions など、AWS 上で実行しないアプリケーションを AWS リソースに接続します。よく知られている SAML ID プロバイダーの例には、Shibboleth や Active Directory Federation Services があります。

### Note

セキュリティ上のベストプラクティスとして、IAM で SAML フェデレーションを使用する代わりに、外部 SAML ID プロバイダーを使用して [IAM Identity Center](#) で人間ユーザーを管理することをお勧めします。IAM ユーザーが必要な特定の状況についての情報は、「[IAM ユーザー \(ロールの代わりに\) を作成する場合](#)」を参照してください。

ID プロバイダーを使用すると、カスタムサインインコードを作成したり独自のユーザー ID を管理したりする必要がありません。IdP では、これらを行うための環境が用意されています。外部ユーザーは IdP を使用してサインインします。これらの外部 ID に、アカウントの AWS リソースを使用するアクセス許可を与えることができます。ID プロバイダーは、アクセスキーなどの長期的セキュリティ認証情報をアプリケーションに配布したり組み込んだりする必要がないので、AWS アカウントの安全性の維持に役立ちます。

このガイドでは IAM フェデレーションについて説明します。ユースケースによっては、IAM Identity Center や Amazon Cognito の方がサポートしやすいかもしれません。以下の概要と表は、ユーザーが AWS リソースへのフェデレーションアクセスを取得するために使用できる方法の概略を示しています。

	アカウントの種類	アクセス管理の対象	サポートされている ID ソース
IAM Identity Center とのフェデレーション	AWS Organizations によって管理される複数のアカウント	ワークフォースの人間ユーザー	<ul style="list-style-type: none"> <li>• SAML 2.0</li> <li>• マネージド Active Directory</li> <li>• Identity Center ディレクトリ</li> </ul>
IAM とのフェデレーション	単一のスタンドアロンアカウント	<ul style="list-style-type: none"> <li>• 短期間の小規模デプロイにおける人間ユーザー</li> <li>• マシンユーザー</li> </ul>	<ul style="list-style-type: none"> <li>• SAML 2.0</li> <li>• OIDC</li> </ul>
Amazon Cognito アイデンティティプールとのフェデレーション	すべて	リソースへのアクセスに IAM 認証を必要とするアプリのユーザー	<ul style="list-style-type: none"> <li>• SAML 2.0</li> <li>• OIDC</li> <li>• OAuth 2.0 ソーシャル ID プロバイダーの選択</li> </ul>

## IAM Identity Center とのフェデレーション

人間ユーザーの一元的なアクセス管理を行うには、[IAM Identity Center](#) を使用して、ご自分のアカウントへのアクセスと、それらのアカウント内でのアクセス許可を管理することをお勧めします。IAM Identity Center のユーザーには、AWS リソースへの短期的な認証情報が付与されます。Active Directory、外部 ID プロバイダー (IdP)、または IAM Identity Center のディレクトリを、AWS リソースへのアクセスを割り当てるユーザーおよびグループの ID ソースとして使用できます。

IAM Identity Center は、SAML (Security Assertion Markup Language) 2.0 との ID フェデレーションをサポートしています。これにより、AWS アクセスポータル内のアプリケーションの使用が許可されているユーザーに、フェデレーションシングルサインオンアクセスを提供します。それにより、ユーザーは、AWS Management Consoleやサードパーティー製アプリケーション (Microsoft 365、SAP Concur、Salesforce など) を含め、SAML をサポートするサービスへのシングルサインオンが可能になります。



## IAM とのフェデレーション

IAM Identity Center で人間ユーザーを管理することを強くお勧めしますが、短期間の小規模デプロイでは、人間ユーザーのために IAM によるフェデレーションユーザーアクセスを有効にできます。IAM では、SAML 2.0 と Open ID Connect (OIDC) IdP を個別に使用し、アクセス制御にフェデレーションユーザー属性を使用することができます。IAM を使用すると、コストセンター、役職、ロケールなどのユーザー属性を IdP から AWS に渡し、これらの属性に基づいてきめ細かいアクセス権限を実装できます。

ワークロードとは、アプリケーションやバックエンドプロセスなど、ビジネス価値を提供するリソースやコードの集合体のことです。ワークロードでは、AWS サービス、アプリケーション、運用ツール、およびコンポーネントにリクエストを送信するために IAM ID が必要になる場合があります。これらの ID には、Amazon EC2 インスタンスや AWS Lambda 関数などの AWS 環境で実行中のマシンが含まれます。

また、アクセスを必要とする外部の関係者のマシン ID を管理することもできます。マシン ID へのアクセスを許可するには、IAM ロールを使用できます。IAM ロールは特定のアクセス許可を持ち、ロールセッションで一時的なセキュリティ認証情報に依存することで AWS にアクセスする方法を提供します。さらに、AWS 環境にアクセスする必要がある AWS の外部のマシンが含まれる場合があります。AWS の外部で実行されるマシンには、[IAM Roles Anywhere](#) を使用できます。ロールの詳細については、「[IAM ロール](#)」をご参照ください。ロールを使用して AWS アカウント へのアクセス許可を委任する方法については「[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)」を参照してください。

IdP を直接 IAM にリンクするには、ID プロバイダーエンティティを作成して、AWS アカウントと IdP の間に信頼関係を確立します。IAM は、[OpenID Connect \(OIDC\)](#) または [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#) と互換性のある IdP をサポートします。これら IdP の AWS での使用についての詳細は、以下のセクションを参照してください。

- [OIDC フェデレーション](#)
- [SAML 2.0 フェデレーション](#)

## Amazon Cognito アイデンティティプールとのフェデレーション

Amazon Cognito は、モバイルアプリやウェブアプリでユーザーを認証および承認したい開発者向けに設計されています。Amazon Cognito ユーザープールは、アプリにサインインおよびサインアップ機能を追加します。アイデンティティプールは、AWS で管理する保護されたリソー



スへのアクセス権をユーザーに付与する IAM 認証情報を提供します。アイデンティティプールは、[AssumeRoleWithWebIdentity](#) API オペレーションを通じて一時セッションの認証情報を取得します。

Amazon Cognito は、SAML と OpenID Connect をサポートする外部 ID プロバイダー、Facebook、Google、Amazon などのソーシャル ID プロバイダーと連携しています。アプリは、ユーザープールまたは外部 IdP を使用してユーザーをサインインさせ、IAM ロールのカスタマイズされた一時セッションを使用して、ユーザーに代わってリソースを取得できます。

## 一般的なシナリオ

### Note

人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用を必須とすることをお勧めします。AWS IAM Identity Center の使用を検討したことがある場合 IAM Identity Center を使用すると、複数の AWS アカウント へのアクセスを一元的に管理できます。ユーザーには、割り当てられたすべてのアカウントに対する MFA で保護された Single Sign-On によるアクセスを、1 つの場所から提供することができます。IAM Identity Center では、その内部でユーザー ID の作成および管理を行います。あるいは、既存の SAML 2.0 互換 ID プロバイダーにも簡単に接続することができます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM Identity Center とは?) を参照してください。

外部 ID プロバイダー (IdP) を使用して、AWS と外部 IdP 外のユーザー ID を管理できます。外部 IdP は、OpenID Connect (OIDC) または Security Assertion Markup Language (SAML) のいずれかを使用して ID 情報を AWS に提供できます。OIDC は、AWS 上で実行しないアプリケーションが AWS リソースにアクセスする必要がある場合に広く使用されます。

外部 IdP サービスとの連携を設定する場合、IAM の ID プロバイダーを作成し、外部 IdP とその設定について AWS に通知します。これにより、AWS アカウント と外部 IdP の間の信頼が確立されます。以下のトピックでは、IAM ID プロバイダーを使用する一般的なシナリオについて説明します。

### トピック

- [モバイルアプリのための Amazon Cognito の使用](#)
- [モバイルアプリへの OIDC フェデレーション API オペレーションの使用](#)

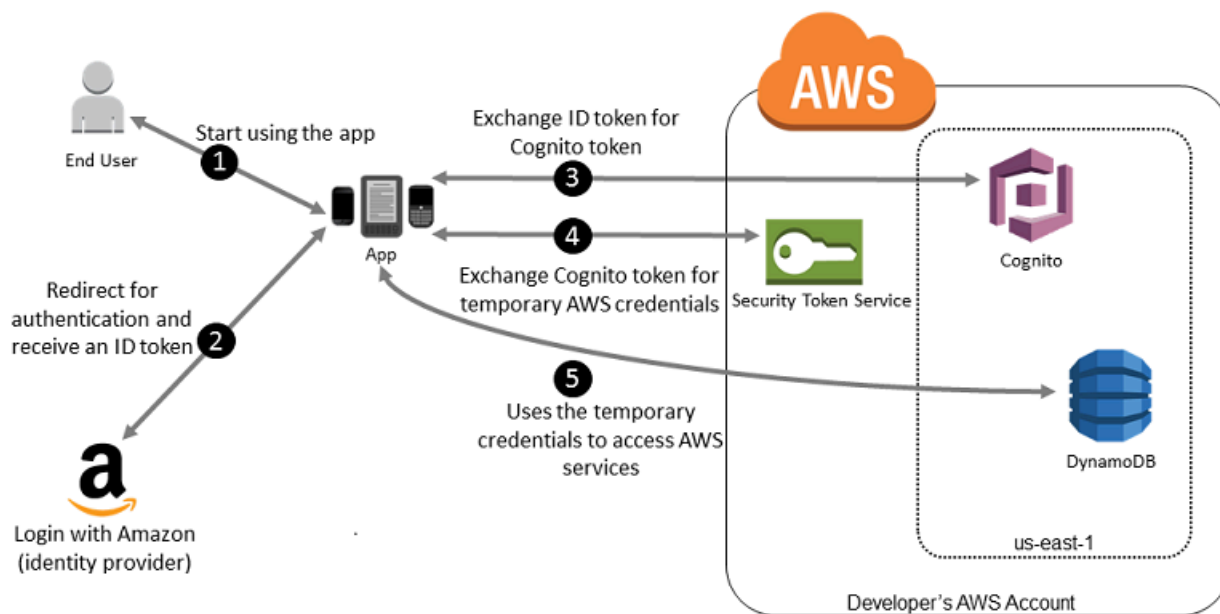
## モバイルアプリのための Amazon Cognito の使用

OIDC フェデレーションの推奨される使い方は、[Amazon Cognito](#) を使用することです。たとえば、開発者の Adele は、モバイルデバイス用のゲームを開発しています。スコアやプロフィールのようなユーザーデータは Amazon S3 と Amazon DynamoDB に保存します。Adele はこのデータを 1 台のデバイスにローカルで保存して、Amazon Cognito を使って他のデバイス間で同期を取ることでもあります。セキュリティおよびメンテナンス上の理由により、長期的な AWS セキュリティ認証情報をこのゲームで配布することはできません。また、このゲームが多数のユーザーを集めることもわかっています。以上のような理由から、IAM でプレーヤーごとに新しいユーザー ID を作成することはしません。代わりに、ユーザーがよく知られた外部 ID プロバイダー (IdP) (Login with Amazon、Facebook、Google など) や任意の OpenID Connect (OIDC) 互換の IdP ですでに確立した ID を使用してサインインできるようにゲームを開発します。ゲームでは、このようなプロバイダーの認証メカニズムを活用して、ユーザーの ID を検証できます。

モバイルアプリから自分の AWS リソースにアクセスできるようにするために、Adele はまず自分の選択した IdP に開発者 ID を登録します。また、これらのプロバイダーそれぞれについてアプリケーションを設定します。ゲーム用の Amazon S3 バケットおよび DynamoDB テーブルを含む AWS アカウントで、Amazon Cognito を使用して、ゲームに必要なアクセス許可を厳密に定義する IAM ロールを作成します。OIDC IdP を使用している場合は、IAM OIDC ID プロバイダーのエントリも作成して、AWS アカウント内の [Amazon Cognito ID プール](#) と IdP の間に信頼を確立します。

アプリのコードでは、前の手順で設定した IdP のサインインインターフェイスを呼び出します。IdP がユーザーのサインイン操作をすべて処理し、アプリは OAuth アクセストークンまたは OIDC ID トークンをプロバイダーから取得します。アプリは、この認証情報を、AWS アクセスキー ID、シークレットアクセスキー、およびセッショントークンで構成される一時的セキュリティ認証情報のセットと交換できます。その後、アプリはこれらの認証情報を使用して、AWS によって提供されるウェブサービスにアクセスできます。アプリは、引き受けるロールで定義されているアクセス権限に制限されます。

以下の図は、この処理の流れを単純化して示しており、IdP として Login with Amazon を使用しています。ステップ 2 については、アプリで、Facebook、Google、その他の OIDC 互換 IdP を利用することもできますが、この図には示していません。



1. 顧客はモバイルデバイスでアプリを起動します。アプリはユーザーにサインインするように求めます。
2. アプリは Login with Amazon を使用して、ユーザーの認証情報を承認します。
3. このアプリは Amazon Cognito API オペレーションの `GetId` と `GetCredentialsForIdentity` を使用して、Login with Amazon ID トークンを Amazon Cognito トークンに置き換えます。Login with Amazon プロジェクトを信頼するように設定されている Amazon Cognito は、一時的なセッション認証情報として AWS STS と交換するトークンを生成します。
4. このアプリで Amazon Cognito から一時的なセキュリティ認証情報を受信します。また、アプリは Amazon Cognito のベーシック (クラシック) ワークフローを使用して、`AssumeRoleWithWebIdentity` を使用する AWS STS からトークンを取得します。詳細については、「Amazon Cognito 開発者ガイド」の「[ID プール \(フェデレーション ID\) 認証フロー](#)」を参照してください。
5. アプリは一時的なセキュリティ証明書を使用して、アプリの動作に必要ないずれの AWS リソースにもアクセスできます。一時的なセキュリティ証明書に関連付けられたロールとその割り当てられたポリシーによって、アクセス可能なリソースが決まります。

以下のプロセスに従って、Amazon Cognito を使用してユーザーを認証してアプリに AWS リソースに対するアクセス許可を付与するように、アプリを設定します。このシナリオを実行するための具体的な手順については、Amazon Cognito のドキュメントを参照してください。

1. (オプション) Login with Amazon、Facebook、Google、または他の任意の OpenID Connect (OIDC) 互換 IdP で開発者としてサインアップし、プロバイダーで 1 つ以上のアプリを設定します。Amazon Cognito ではユーザーの認証されていない (ゲスト) アクセスがサポートされているので、この手順はオプションです。
2. AWS Management Console の [Amazon Cognito](#) に移動します。Amazon Cognito ウィザードを使用して ID プールを作成します。これは、アプリのために Amazon Cognito がエンドユーザー ID を整理しておくために使用するテナントです。ID プールは、アプリ間で共有できます。ID プールをセットアップすると、Amazon Cognito ユーザーのアクセス許可を定義する 1 つまたは 2 つの IAM ロールが Amazon Cognito によって作成されます (1 つは認証された ID 用、2 つ目は認証されていない「ゲスト」ID 用)。
3. [AWS Amplify](#) をアプリと統合して、Amazon Cognito の使用に必要なファイルをインポートします。
4. ID プールの ID、AWS アカウント 番号、および ID プールに関連付けたロールの Amazon リソースネーム (ARN) を渡して、Amazon Cognito 認証情報プロバイダーのインスタンスを作成します。AWS Management Console の Amazon Cognito ウィザードによって、作業の開始に役立つサンプルコードが提供されます。
5. アプリが AWS リソースにアクセスするときに、認証情報プロバイダーインスタンスをクライアントオブジェクトに渡します。このオブジェクトが一時的なセキュリティ認証情報をクライアントに渡します。認証情報のアクセス権限は、前に定義したロールに基づいています。

詳細については、次を参照してください:

- [AWS Amplify Framework Documentation に \(Android\) でサインインします。](#)
- [AWS Amplify Framework Documentation に \(iOS\) でサインインします。](#)

## モバイルアプリへの OIDC フェデレーション API オペレーションの使用

最良の結果を得るには、ほぼすべての OIDC フェデレーションシナリオで Amazon Cognito を認証ブローカーとして使用してください。Amazon Cognito は使いやすく、匿名 (認証されていない) アクセスのような追加機能が用意されていて、複数のデバイスおよびプロバイダーにわたってユーザーデータが同期されます。ただし、手動で AssumeRoleWithWebIdentity API を呼び出すことによって OIDC フェデレーションを使用するアプリを既に作成してある場合は、それを使用し続けることができます。アプリは引き続き正常に動作します。

Amazon Cognito を使用せずに OIDC フェデレーションを使用するプロセスは、次の一般的な概要に従います。

1. 外部 ID プロバイダー (IdP) に開発者としてサインアップし、IdP に対してアプリを設定して、アプリの一意の ID を受け取ります。(異なる IdP は、このプロセスに異なる用語を使用します。この概要では、IdP を使用してアプリを識別するプロセスに configure (設定) という用語を使用します。) 各 IdP からはそれぞれ固有のアプリ ID を受け取るため、複数の IdP に対して同じアプリを設定した場合、アプリには複数のアプリ ID が与えられます。プロバイダーごとに複数のアプリを設定できます。

以下の外部リンクで、一般的ないくつかの ID プロバイダー (IdP) の利用方法に関する情報を入手できます。

- [Login with Amazon 開発者センター](#)
- Facebook 開発者サイトの「[アプリまたはウェブサイトに Facebook ログインを追加する](#)」
- Google 開発者サイトの「[ログインでの OAuth 2.0 の使用 \(OpenID Connect\)](#)」

#### Important

Google、Facebook、または Amazon Cognito の OIDC IDプロバイダーを使用する場合は、AWS Management Console に個別の IAM ID プロバイダーを作成しないでください。AWS にはこれらの OIDC IDプロバイダーが組み込まれており、使用できます。次の手順をスキップして、ID プロバイダーを使用して新しいロールの作成に直接移動します。

2. OIDC と互換性のある Google、Facebook、または Amazon Cognito 以外の IdP を使用する場合は、IAM ID プロバイダーのエンティティを作成します。
3. IAM で、[1つ以上のロールを作成します](#)。ロールごとに、だれがそのロールを引き受けることができるか (信頼ポリシー)、アプリのユーザーがどのアクセス権限を持つか (アクセス権限ポリシー) を定義します。通常は、アプリがサポートする IdP ごとに 1 つのロールを作成します。たとえば、ユーザーが Login with Amazon を使用してサインインするアプリで想定されるロール、ユーザーが Facebook を使用してサインインする同じアプリの 2 つ目のロール、およびユーザーが Google を使用してサインインするアプリの 3 つ目のロールを作成します。信頼関係の確立用に、IdP (Amazon.com など) を Principal (信頼されるエンティティ) として指定し、IdP 提供のアプリ ID と一致する Condition を含めます。異なるプロバイダーのロールの例は、[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)で説明されています。
4. アプリケーションで、IdP に対してユーザーを認証します。この具体的な方法は、どの IdP を使用しているか (Login with Amazon、Facebook、または Google)、どのプラットフォームでアプリを実行しているかの両方によって変わります。たとえば、Android アプリと iOS アプリや JavaScript ベースのウェブアプリとでは、認証方法が異なることがあります。



一般的に、ユーザーがまだサインインしていない場合、IdP はサインインページを表示するように対処します。IdP はユーザーを認証した後、ユーザーに関する情報と共に認証トークンをアプリに返します。含まれる情報は、IdP が公開する情報と、ユーザーが共有する情報によって異なります。この情報はアプリで利用できます。

5. アプリで、`AssumeRoleWithWebIdentity` アクションを署名なしで呼び出して、一時的なセキュリティクレデンシャルをリクエストします。リクエストでは、IdP の認証トークンを渡し、その IdP 用に作成した IAM ロールの Amazon リソースネーム (ARN) を指定します。AWS は、トークンが信頼されていて有効であることを確認します。確認できた場合、リクエストで指定したロールのアクセス許可が割り当てられた一時的セキュリティ認証情報をアプリに返します。応答には、IdP がユーザーに関連付けた一意のユーザー ID など、ユーザーに関する IdP からのメタデータも含まれます。
6. `AssumeRoleWithWebIdentity` レスポンスからの一時的セキュリティ認証情報を使用して、アプリから AWS API オペレーションへの署名付きリクエストを行います。IdP からのユーザー ID 情報は、アプリ内のユーザーを区別できます。例えば、プレフィックスまたはサフィックスとしてユーザー ID を含む Amazon S3 フォルダにオブジェクトを配置できます。これにより、そのフォルダをロックするアクセス制御ポリシーを作成して、その ID を持つユーザーに対してのみフォルダへのアクセスを許可できます。詳細については、「[AWS STS フェデレーティッドユーザーセッションプリンシパル](#)」を参照してください。
7. アプリで一時的なセキュリティ認証情報がキャッシュに格納されるようにすることで、アプリから AWS へのリクエストがあるたびに、新しい証明書を取得する必要がなくなります。デフォルトで、認証情報は 1 時間有効です。認証情報が失効すると (または失効前に)、`AssumeRoleWithWebIdentity` を呼び出して新しい 1 組の一時的なセキュリティ認証情報を取得します。IdP とそのトークンの管理方法によっては、`AssumeRoleWithWebIdentity` を新しく呼び出す前に、IdP のトークンを更新する必要があります。IdP のトークンも通常は一定期間後に失効するためです。AWS SDK for iOS または AWS SDK for Android を使用する場合は、[AmazonSTSCredentialsProvider](#) アクションを使用できます。このアクションは IAM の一時的な認証情報を管理し、必要に応じて更新します。

## OIDC フェデレーション

ワークフローを使用して Amazon S3 や DynamoDB にアクセスする GitHub Actions などの AWS リソースにアクセスするアプリケーションを作成するとします。

このようなワークフローを作成する場合、AWS アクセスキーで署名される必要のある AWS サービスに対してリクエストを実行します。ただし、AWS 外部のアプリケーションには AWS 認証情報を

長期間保存しないことを強くお勧めします。代わりに、「OIDC フェデレーション」を使用して、必要に応じて一時的な AWS セキュリティ認証情報を動的に要求するようにアプリケーションを設定します。提供される一時的な認証情報は、アプリケーションで必要とされるタスクの実行に必要なアクセス許可のみを持つ AWS ロールにマッピングされます。

OIDC フェデレーションを使用すると、カスタムサインインコードを作成したり独自のユーザー ID を管理したりする必要はありません。代わりに、GitHub Actions などのアプリケーションやその他の [OpenID Connect \(OIDC\) 互換 IdP](#) で OIDC を使用して AWS での認証を行うことができます。JSON Web トークン (JWT) として知られる認証トークンが受け取られたら、AWS アカウントの特定のリソースを使用するアクセス許可を持つ IAM ロールにマップされる AWS の一時的なセキュリティ認証情報に変換されます。IdP を使用すると、アプリケーションで長期的なセキュリティ認証情報を埋め込んで配布する必要がないため、AWS アカウントの安全性の維持に役立ちます。

ほとんどのシナリオでは、[Amazon Cognito](#) を使用することをお勧めします。Amazon Cognito は ID ブローカーとして機能し、ユーザーの代わりに多くのフェデレーション作業を行うからです。詳細については、以下のセクション「[モバイルアプリのための Amazon Cognito の使用](#)」を参照してください。

#### Note

OpenID Connect (OIDC) ID プロバイダーによって発行された JSON ウェブトークン (JWT) では、トークンの有効期限を指定する有効期限が exp クレームに含まれています。IAM は、[OpenID Connect \(OIDC\) Core 1.0 標準](#)で許可されているように、クロックスキューを考慮し、JWT で指定された有効期限を 5 分間延長します。つまり、有効期限後 5 分以内に IAM が受信した OIDC JWT は受け入れられ、評価と処理が行われます。

#### トピック

- [IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)
- [OpenID Connect ID プロバイダーのサムプリントを取得する](#)
- [OIDC フェデレーションに関するその他のリソース](#)

## IAM で OpenID Connect (OIDC) ID プロバイダーを作成する

IAM OIDC ID プロバイダーは [OpenID Connect \(OIDC\) 標準](#) (例: Google または Salesforce) をサポートする ID プロバイダー (IdP) サービスを表す IAM のエンティティです。OIDC 互換 IdP と AWS アカウントの間で信頼性を確立するときに IAM OIDC ID プロバイダーを使用します。このプロバイ

ダーは、AWS リソースへのアクセスを必要とするモバイルアプリやウェブアプリケーションを作成するときに、カスタムサインインコードの作成や独自のユーザー ID の管理をしない場合に役立ちます。このシナリオの詳細については、「[the section called “OIDC フェデレーション”](#)」を参照してください。

AWS Management Console、AWS Command Line Interface、Tools for Windows PowerShell、または IAM API を使用して、IAM OIDC ID プロバイダーを作成および管理できます。

IAM OIDC ID プロバイダーを作成したら、1 つ以上の IAM ロールを作成する必要があります。ロールは AWS のアイデンティティであり、それ自体には (ユーザーのような) 認証情報がありません。しかし、この例で、ロールが動的に割り当てられるフェデレティッドユーザーは、組織の IdP から認証されます。このロールで、組織の IdP が AWS にアクセスするための一時的なセキュリティ認証情報をリクエストできるようにします。ロールに割り当てられているポリシーは、フェデレティッドユーザーが AWS で実行できることを決定します。サードパーティー ID プロバイダーのロールを作成するには、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」をご参照ください。

#### Important

oidc-provider リソースをサポートするアクションの ID ベースのポリシーを設定する場合は、IAM は指定されたパスを含む OIDC ID プロバイダーの完全な URL を評価します。OIDC ID プロバイダーの URL にパスがある場合は、そのパスを oidc-provider ARN に Resource 要素の値として含める必要があります。URL ドメインにフォワードスラッシュおよびワイルドカード (/\*) を追加することもできます。または、URL パスの任意の時点でワイルドカード文字 (\* および ?) は任意の時点で使用します。リクエスト内の OIDC ID プロバイダー URL がポリシーの Resource 要素で設定されている値と一致しない場合、リクエストは失敗します。

IAM OIDC フェデレーションの一般的な問題をトラブルシューティングするには、AWS re:Post の [OIDC に関連するエラーの解決](#)に関する記事を参照してください。

#### トピック

- [前提条件: ID プロバイダーの設定を検証する](#)
- [OIDC プロバイダーの作成と管理 \(コンソール\)](#)
- [IAM OIDC ID プロバイダーの作成と管理 \(AWS CLI\)](#)
- [OIDC ID プロバイダーの作成と管理 \(AWS API\)](#)



## 前提条件: ID プロバイダーの設定を検証する

IAM OIDC ID プロバイダーを作成する前に、IdP から次の情報を取得しておく必要があります。OIDC プロバイダー設定情報を取得する方法の詳細については、IdP のドキュメントを参照してください。

1. OIDC ID プロバイダーの公開 URL を決定します。URL は「https://」で始まる必要があります。OIDC 標準によって、パスコンポーネントは許可されますが、クエリパラメータは許可されません。通常、URL は `https://server.example.org` や `https://example.com` などのホスト名のみで構成されます。URL にポート番号を含めることはできません。
2. OIDC ID プロバイダーの URL の末尾に `/.well-known/openid-configuration` を追加して、プロバイダーの公開されている設定ドキュメントとメタデータを確認します。[OpenID Connect プロバイダー検出エンドポイント URL](#) から取得できるプロバイダーの設定ドキュメントとメタデータを含む JSON 形式の検出ドキュメントが必要です。
3. プロバイダーの設定情報に次の値が含まれていることを確認します。openid-configuration にこれらのフィールドのいずれかがない場合は、検出ドキュメントを更新する必要があります。このプロセスは ID プロバイダーによって異なる可能性があるため、IdP のドキュメントに従ってこのタスクを完了してください。
  - issuer: ご使用のドメインの URL。
  - jwks\_uri: IAM がパブリックキーを取得する JSON Web Key Set (JWKS) エンドポイント。ご使用の ID プロバイダーで、openid-configuration に JSON Web Key Set (JWKS) エンドポイントを含んでいる必要があります。この URI は、ID プロバイダーから署名付きトークンを検証するために使用されるパブリックキーを取得する場所を定義します。
  - claims\_supported: IdP からの OIDC 認証レスポンスに、フェデレーションユーザーのアクセス許可を確認するために AWS が IAM ポリシーで使用する必須属性が含まれていることを確認するのに役立つユーザーに関する情報。クレームに使用できる IAM 条件キーのリストについては、「[OIDC AWS フェデレーションで使用できるキー](#)」を参照してください。
  - aud: SON Web Tokens (JWTs) で IdP の問題が発生する audience クレーム値を決定する必要があります。audience (aud) クレームはアプリケーション固有であり、トークンの受取人を識別します。OpenID Connect プロバイダーにモバイルアプリまたはウェブアプリを登録すると、アプリケーションを識別するクライアント ID が確立されます。クライアント ID は、認証のために aud クレームで渡されるアプリケーションの一意の識別子です。aud クレームは、IAM OIDC ID プロバイダーを作成するときに Audience 値と一致する必要があります。

- `iat`: クレームには、ID トークンが発行された時刻を表す `iat` の値を含める必要があります。
- `iss`: ID プロバイダーの URL。URL は `https://` で始まり、IAM に提供されるプロバイダー URL と呼応している必要があります。OIDC 標準によって、パスコンポーネントは許可されますが、クエリパラメータは許可されません。通常、URL は `https://server.example.org` や `https://example.com` などのホスト名のみで構成されます。URL にポート番号を含めることはできません。
- `response_types_supported`: `id_token`
- `subject_types_supported`: `public`
- `id_token_signing_alg_values_supported`: `RS256`

#### Note

以下の例では、`custom` のような追加クレームを含めることができますが、AWS STS はクレームを無視します。

```
{
  "issuer": "https://example-domain.com",
  "jwks_uri": "https://example-domain.com/jwks/keys",
  "claims_supported": [
    "aud",
    "iat",
    "iss",
    "name",
    "sub",
    "custom"
  ],
  "response_types_supported": [
    "id_token"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "subject_types_supported": [
    "public"
  ]
}
```

## OIDC プロバイダーの作成と管理 (コンソール)

AWS Management Console で IAM OIDC ID プロバイダーを作成および管理するには、次の手順に従います。

### Important

Google、Facebook、または Amazon Cognito の OIDC ID プロバイダーを使用している場合は、この手順を使用して別の IAM ID プロバイダーを作成しないでください。これらの OIDC ID プロバイダーは、AWS にすでに組み込まれており、使用できます。代わりに、「[OpenID Connect フェデレーション用のロールを作成する \(コンソール\)](#)」の手順に従って ID プロバイダーの新しいロールを作成します。

## IAM OIDC ID プロバイダーを作成するには (コンソール)

1. IAM OIDC ID プロバイダーを作成する前に、アプリケーションを IdP に登録してクライアント ID を受け取る必要があります。クライアント ID (対象者とも呼ばれます) は、アプリを IdP に登録したときに発行されるアプリの一意の識別子です。クライアント ID を取得する方法の詳細については、IdP のドキュメントを参照してください。

### Note

AWS は、IdP サーバー証明書を検証する証明書のサムプリントを使用する代わりに、信頼されたルート証明機関 (CA) のライブラリを介して一部の OIDC ID プロバイダ (IdP) との通信を保護します。このような場合、従来のサムプリントは設定に残りますが、検証には使用されなくなります。これらの OIDC IdP には、Auth0、GitHub、GitLab、Google に加えて、Amazon S3 バケットを使用して JSON ウェブキーセット (JWKS) エンドポイントをホストするものが含まれます。

2. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
3. ナビゲーションペインで、[Identity providers] (ID プロバイダ) を選択し、[Add provider] (プロバイダを追加) を選択します。
4. [Configure provider] (プロバイダーの設定) で、[OpenID Connect] を選択します。
5. [プロバイダーの URL] には IdP の URL を入力します。URL は次の制限に準拠する必要があります。
  - URL では大文字と小文字は区別されません。

- URL は **https://** で始まる必要があります。
  - URL にポート番号を含めることはできません。
  - AWS アカウント 内で、各 IAM OIDC ID プロバイダーは一意的 URL を使用する必要があります。AWS アカウント で OpenID Connect プロバイダーで既に使用されている URL を送信しようとする、エラーが発生します。
6. [対象者] には、IdP に登録して [Step 1](#) で受け取ったアプリケーションのクライアント ID を入力します。このアプリケーションは AWS に対するリクエストを実行します。この IdP のクライアント ID (対象者) が他にも存在する場合は、後でプロバイダーの詳細ページで追加できます。

**Note**

IdP JWT トークンに azp クレームが含まれている場合は、Audience 値としてこの値を入力します。

7. (オプション) [Add tags (タグの追加)] では、キーバリューのペアを追加して IdP の特定と整理を行うことができます。タグを使用して、AWS リソースへのアクセスを制御することもできます。IAM OIDC ID プロバイダーのタグ付けの詳細については、「[OpenID Connect \(OIDC\) ID プロバイダーのタグ付け](#)」を参照してください。タグを追加を選択します。タグキーバリューのペアごとに値を入力します。
8. 入力した情報を確認します。完了したら、[Add provider] (プロバイダーを追加) を選択します。IAM は、OIDC IdP サーバー証明書の上位中間 CA サンプルを取得して使用し、IAM OIDC ID プロバイダーを作成しようとします。

**Note**

OIDC ID プロバイダーの証明書チェーンは、ドメインまたは発行者の URL で始まり、中間証明書が続いた後、ルート証明書で終わる必要があります。証明書チェーンの順序が異なる場合、または証明書の重複や追加が含まれる場合は、署名の不一致エラーが表示され、STS は JSON ウェブトークン (JWT) の検証に失敗します。エラーを解決するには、サーバーから返されたチェーン内の証明書の順序を修正します。証明書チェーン標準の詳細については、RFC Series ウェブサイトの [RFC 5246 の certificate\\_list](#) を参照してください。

9. ID プロバイダーに IAM ロールを割り当てて、ID プロバイダーによって管理される外部ユーザー ID にアカウント内の AWS リソースへのアクセス許可を与えます。ID フェデレーション用の

ロールの作成の詳細については、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」をご参照ください。

**Note**

ロール信頼ポリシーで使用される OIDC IdP は、そのロール信頼ポリシーを信頼するロールと同じアカウントにある必要があります。

IAM OIDC ID プロバイダーのサムプリントを追加または削除するには (コンソール)

**Note**

AWS は、IdP サーバー証明書を検証する証明書のサムプリントを使用する代わりに、信頼されたルート証明機関 (CA) のライブラリを介して一部の OIDC ID プロバイダ (IdP) との通信を保護します。このような場合、従来のサムプリントは設定に残りますが、検証には使用されなくなります。これらの OIDC IdP には、Auth0、GitHub、GitLab、Google に加えて、Amazon S3 バケットを使用して JSON ウェブキーセット (JWKS) エンドポイントをホストするものが含まれます。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[Identity providers (ID プロバイダ)] を選択します。その後、更新する IAM ID プロバイダーの名前を選択します。
3. [Thumbprints] (サムプリント) セクションで、[Manage] (管理) を選択します。新しいサムプリント値を入力するには、[Add thumbprint] (サムプリントを追加) を選択します。サムプリントを削除するには、削除するサムプリントの横にある [Remove] (削除) を選択します。

**Note**

IAM OIDC ID プロバイダーには少なくとも 1 つのサムプリントが必要であり、最大 5 つのサムプリントを指定できます。

完了したら、[Save changes] (変更を保存) を選択します。

## IAM OIDC ID プロバイダーの対象者を追加するには (コンソール)

1. ナビゲーションペインで、[Identity providers] (ID プロバイダー) を選択し、更新する IAM ID プロバイダーの名前を選択します。
2. [Audiences] (対象者) セクションで [Actions] (アクション) を選択し、[Add audience] (対象者を追加) を選択します。
3. IdP に登録して [Step 1](#) で受け取ったアプリケーションのクライアント ID を入力します。このアプリケーションは AWS に対するリクエストを実行します。その後、[Add audiences] (対象者を追加) を選択します。

### Note

IAM OIDC ID プロバイダーには少なくとも 1 名の閲覧者が必要であり、最大 100 名まで指定できます。

## IAM OIDC ID プロバイダーの対象者を削除するには (コンソール)

1. ナビゲーションペインで、[Identity providers] (ID プロバイダー) を選択し、更新する IAM ID プロバイダーの名前を選択します。
2. [Audiences] (対象者) セクションで、削除する対象者の横にあるラジオボタンを選択し、[Actions] (アクション) を選択します。
3. [Remove audience] (対象者を削除) を選択します。新しいウィンドウが開きます。
4. 対象者を削除すると、対象者とフェデレートする ID は対象者に関連付けられたロールを引き受けることができません。ウィンドウで、警告を読み、フィールドに単語 `remove` を入力して対象者を削除することを確認します。
5. 対象者を削除するには、[Remove] (削除) を選択します。

## IAM OIDC ID プロバイダーを削除するには (コンソール)

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[Identity providers (ID プロバイダ)] を選択します。
3. 削除する IAM ID プロバイダーの横にあるチェックボックスをオンにします。新しいウィンドウが開きます。
4. フィールドに単語 `delete` を入力して、プロバイダーを削除することを確認します。その後、[Delete] (削除) をクリックします。

## IAM OIDC ID プロバイダーの作成と管理 (AWS CLI)

以下の AWS CLI コマンドを使用して IAM OIDC ID プロバイダーを作成および管理できます。

### IAM OIDC ID プロバイダーを作成するには (AWS CLI)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーを一覧表示するには、次のコマンドを実行します。

- [aws iam list-open-id-connect-providers](#)

2. 新しい IAM OIDC ID プロバイダーを作成するには、次のコマンドを実行します。

- [aws iam create-open-id-connect-provider](#)

既存の IAM OIDC ID プロバイダー (AWS CLI) のサーバー証明書のサムプリントのリストを更新するには

• IAM OIDC ID プロバイダーのサーバー証明書のサムプリントのリストを更新するには、次のコマンドを実行します。

- [aws iam update-open-id-connect-provider-thumbprint](#)

既存の IAM OIDC ID プロバイダー (AWS CLI) にタグを付けるには

• 既存の IAM OIDC ID プロバイダーにタグを付けるには、次のコマンドを実行します。

- [aws iam tag-open-id-connect-provider](#)

既存の IAM OIDC ID プロバイダー (AWS CLI) のタグを一覧表示するには

• 既存の IAM OIDC ID プロバイダーのタグを一覧表示するには、次のコマンドを実行します。

- [aws iam list-open-id-connect-provider-tags](#)

IAM OIDC ID プロバイダー (AWS CLI) のタグを削除するには

• 既存の IAM OIDC ID プロバイダーのタグを削除、次のコマンドを実行します。

- [aws iam untag-open-id-connect-provider](#)



既存のIAM OIDC ID プロバイダーのクライアント ID を追加または削除するには (AWS CLI)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーのリストを取得するには、次のコマンドを実行します。
  - [aws iam list-open-id-connect-providers](#)
2. (オプション) IAM OIDC ID プロバイダーの詳細情報を取得するには、次のコマンドを実行します。
  - [aws iam get-open-id-connect-provider](#)
3. 既存の IAM OIDC ID プロバイダーに新しいクライアント ID を追加するには、次のコマンドを実行します。
  - [aws iam add-client-id-to-open-id-connect-provider](#)
4. 既存の IAM OIDC ID プロバイダーからクライアント ID を削除するには、次のコマンドを実行します。
  - [aws iam remove-client-id-from-open-id-connect-provider](#)

IAM OIDC ID プロバイダーを削除するには (AWS CLI)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーのリストを取得するには、次のコマンドを実行します。
  - [aws iam list-open-id-connect-providers](#)
2. (オプション) IAM OIDC ID プロバイダーの詳細情報を取得するには、次のコマンドを実行します。
  - [aws iam get-open-id-connect-provider](#)
3. IAM OIDC ID プロバイダーを削除するには、次のコマンドを実行します。
  - [aws iam delete-open-id-connect-provider](#)

OIDC ID プロバイダーの作成と管理 (AWS API)

以下の IAM API コマンドを使用して OIDC プロバイダーを作成および管理できます。



## IAM OIDC ID プロバイダーを作成するには (AWS API)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーのリストを取得するには、次のオペレーションを呼び出します。

- [ListOpenIDConnectProviders](#)

2. 新しい IAM OIDC ID プロバイダーを作成するには、次のオペレーションを呼び出します。

- [CreateOpenIDConnectProvider](#)

## 既存の IAM OIDC ID プロバイダーのサーバー証明書のサムプリントのリストを更新するには (AWS API)

- IAM OIDC ID プロバイダーのサーバー証明書のサムプリントのリストを更新するには、次のオペレーションを呼び出します。

- [UpdateOpenIDConnectProviderThumbprint](#)

## 既存の IAM OIDC ID プロバイダーにタグを付けるには (AWS API)

- 既存の IAM OIDC ID プロバイダーにタグを付けるには、次のオペレーションを呼び出します。

- [TagOpenIDConnectProvider](#)

## 既存の IAM OIDC ID プロバイダーのタグを一覧表示するには (AWS API)

- 既存の IAM OIDC ID プロバイダーのタグを一覧表示するには、次のオペレーションを呼び出します。

- [ListOpenIDConnectProviderTags](#)

## 既存の IAM OIDC ID プロバイダーのタグを削除するには (AWS API)

- 既存の IAM OIDC ID プロバイダーのタグを削除するには、次のオペレーションを呼び出します。

- [UntagOpenIDConnectProvider](#)

既存の IAM OIDC ID プロバイダーのクライアント ID を追加または削除するには (AWS API)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーのリストを取得するには、次のオペレーションを呼び出します。
  - [ListOpenIDConnectProviders](#)
2. (オプション) IAM OIDC ID プロバイダーの詳細情報を取得するには、次のオペレーションを呼び出します。
  - [GetOpenIDConnectProvider](#)
3. 既存の IAM OIDC ID プロバイダーに新しいクライアント ID を追加するには、次のオペレーションを呼び出します。
  - [AddClientIDToOpenIDConnectProvider](#)
4. 既存の IAM OIDC ID プロバイダーからクライアント ID を削除するには、次のオペレーションを呼び出します。
  - [RemoveClientIDFromOpenIDConnectProvider](#)

IAM OIDC ID プロバイダーを削除するには (AWS API)

1. (オプション) AWS アカウントのすべての IAM OIDC ID プロバイダーのリストを取得するには、次のオペレーションを呼び出します。
  - [ListOpenIDConnectProviders](#)
2. (オプション) IAM OIDC ID プロバイダーの詳細情報を取得するには、次のオペレーションを呼び出します。
  - [GetOpenIDConnectProvider](#)
3. IAM OIDC ID プロバイダーを削除するには、次のオペレーションを呼び出します。
  - [DeleteOpenIDConnectProvider](#)

OpenID Connect ID プロバイダーのサムプリントを取得する

IAM で [OpenID Connect \(OIDC\) ID プロバイダーを作成する](#) 場合は、IAM では、外部 ID プロバイダー (IdP) が使用する証明書に署名した最上位の中間認証局 (CA) のサムプリントが必要です。拇印は、OIDC 互換 IdP の証明書を発行するために使用された CA の証明書の署名です。IAM の OIDC ID

プロバイダーを作成する場合、その IdP からの ID を信頼し、AWS アカウント へのアクセス権限を与えることとなります。CA の証明書のサムプリントを使用することにより、CA によって発行された証明書を、登録されているものと同じ DNS 名で信頼します。これにより、IdP の署名証明書を更新したときに各アカウントの信頼を更新する必要がなくなります。

### Important

ほとんどの場合、フェデレーションサーバーは 2 つの異なる証明書を使用します。

- 1 つ目は、AWS と IdP との間に HTTPS 接続を確立します。この証明書は、AWS Certificate Manager などのよく知られたパブリックルート CA で発行されています。これにより、クライアントは証明書の信頼性とステータスを確認できます。
- 2 つ目はトークンの暗号化に使用され、プライベートまたはパブリックのルート CA によって署名されているはずです。

IAM OIDC ID プロバイダーは、[AWS Command Line Interface](#)、[Tools for Windows PowerShell](#)、または [IAM API](#) を使用して作成できます。これらの方法を使用する場合、サムプリントを手動で指定できます。サムプリントを含めない場合、IAM は OIDC IdP サーバー証明書の最上位中間 CA サムプリントを取得します。サムプリントを含める場合は、サムプリントを手動で取得して AWS に提供する必要があります。

[IAM コンソール](#) を使用して OIDC ID プロバイダーを作成すると、IAM は、OIDC IdP サーバー証明書の最上位中間 CA サムプリントの取得を試みます。

あわせて、OIDC IdP のサムプリントを手動で取得し、IAM で正しいサムプリントが取得されていることを確認することをお勧めします。証明書のサムプリントの取得の詳細については、以下のセクションを参照してください。

### Note

AWS は、IdP サーバー証明書を検証する証明書のサムプリントを使用する代わりに、信頼されたルート証明機関 (CA) のライブラリを介して一部の OIDC ID プロバイダ (IdP) との通信を保護します。このような場合、従来のサムプリントは設定に残りますが、検証には使用されなくなります。これらの OIDC IdP には、Auth0、GitHub、GitLab、Google に加えて、Amazon S3 バケットを使用して JSON ウェブキーセット (JWKS) エンドポイントをホストするものが含まれます。

IAM OIDC フェデレーションの一般的な問題をトラブルシューティングするには、AWS re:Post の [OIDC に関連するエラーの解決](#) に関する記事を参照してください。

## 証明書サムプリントを取得する

OIDC プロバイダーの証明書サムプリントを取得するには、ウェブブラウザと OpenSSL コマンドラインツールを使用します。ただし、IAM OIDC ID プロバイダーを作成するために証明書のサムプリントを手動で取得する必要はありません。次の手順を使用して、OIDC プロバイダーの証明書サムプリントを取得できます。

OIDC IdP のサムプリントを取得するには

1. OIDC IdP のサムプリントを取得する前に、OpenSSL コマンドラインツールを取得する必要があります。このツールを使用して、OIDC IdP の証明書チェーンをダウンロードし、証明書チェーンで最終的な証明書のサムプリントを作成します。OpenSSL のインストールと設定が必要な場合は、「[OpenSSL のインストール](#)」および「[OpenSSL の設定](#)」の手順に従います。
2. 次のように、OIDC IdP の URL (<https://server.example.com> など) で開始し、`/.well-known/openid-configuration` を追加して IdP の設定ドキュメントの URL (以下参照) を作成します。

**`https://server.example.com/.well-known/openid-configuration`**

この URL をウェブブラウザで開き、`server.example.com` を IdP のサーバー名に置き換えます。

3. 表示されたドキュメントで、ウェブブラウザを使用します。検索機能を使用して、テキスト "jwks\_uri" を検索します。テキスト "jwks\_uri" の直後のコロン (:) の後に URL があります。URL の完全修飾ドメイン名をコピーします。`https://` または最上位ドメインより後のパスは含めないでください。

```
{
  "issuer": "https://accounts.example.com",
  "authorization_endpoint": "https://accounts.example.com/o/oauth2/v2/auth",
  "device_authorization_endpoint": "https://oauth2.exampleapis.com/device/code",
  "token_endpoint": "https://oauth2.exampleapis.com/token",
  "userinfo_endpoint": "https://openidconnect.exampleapis.com/v1/userinfo",
  "revocation_endpoint": "https://oauth2.exampleapis.com/revoke",
  "jwks_uri": "https://www.exampleapis.com/oauth2/v3/certs",
  ...
}
```

4. OpenSSL コマンドラインツールを使用して、次のコマンドを実行します。 *keys.example.com* を、[Step 3](#) で取得したドメイン名に置き換えます。

```
openssl s_client -servername keys.example.com -showcerts -  
connect keys.example.com:443
```

5. コマンドウィンドウで、次の例のような証明書が表示されるまでスクロールアップします。複数の証明書が表示される場合は、表示される最後 (コマンド出力の最後) の証明書を見つけます。これには認証機関チェーンの最上位中間 CA の証明書が含まれています。

```
-----BEGIN CERTIFICATE-----  
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC  
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6  
b24xFDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAd  
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN  
MTIwNDI1MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD  
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstC01BTSBDb25z  
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAdBgkqhkiG9w0BCQEWEG5vb251QGft  
YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ  
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T  
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE  
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4  
nUHVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb  
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvjx79LjSTb  
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=  
-----END CERTIFICATE-----
```

証明書 (-----BEGIN CERTIFICATE----- および -----END CERTIFICATE----- 行を含む) をコピーして、テキストファイルに貼り付けます。次に、**certificate.crt** という名前でファイルを保存します。

#### Note

OIDC ID プロバイダーの証明書チェーンは、ドメインまたは発行者の URL で始まり、中間証明書が続いた後、ルート証明書で終わる必要があります。証明書チェーンの順序が異なる場合、または証明書の重複や追加が含まれる場合は、署名の不一致エラーが表示され、STS は JSON ウェブトークン (JWT) の検証に失敗します。エラーを解決するには、サーバーから返されたチェーン内の証明書の順序を修正します。証明書チェーン

標準の詳細については、RFC Series ウェブサイトの [RFC 5246 の certificate\\_list](#) を参照してください。

6. OpenSSL コマンドラインツールを使用して、次のコマンドを実行します。

```
openssl x509 -in certificate.crt -fingerprint -sha1 -noout
```

コマンドウィンドウには、次の例のような証明書サムプリントが表示されます。

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

この文字列からコロンの (:) を削除して、次のような最終的なサムプリントを作成します。

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

7. AWS CLI を使用して IAM OIDC ID プロバイダーを作成する場合は、Windows PowerShell 用ツール、または IAM API では、サムプリントの提供は任意です。作成時にサムプリントを含めない場合、IAM は OIDC IdP サーバー証明書の最上位中間 CA サムプリントを取得します。IAM OIDC ID プロバイダーを作成したら、このサムプリントを IAM によって取得されたサムプリントと比較できます。

IAM コンソールで IAM OIDC ID プロバイダーを作成する場合、コンソールは OIDC IdP サーバー証明書の上位中間 CA サムプリントの取得を試みます。このサムプリントを IAM によって取得されたサムプリントと比較できます。IAM OIDC ID プロバイダーが作成されたら、OIDC プロバイダーの [概要] コンソールページの [エンドポイント検証] タブで IAM OIDC ID プロバイダーのサムプリントを表示できます。

#### Important

取得したサムプリントが、IAM OIDC ID プロバイダーに表示されるサムプリントに一致しない場合は、その OIDC プロバイダーを作成しないでください。代わりに、作成した OIDC プロバイダーを削除してから、しばらく待ってから OIDC プロバイダーの作成を再試行してください。プロバイダーを使用する前に、サムプリントが一致していることを確認します。再試行してもサムプリントが一致しない場合は、[IAM フォーラム](#)を使用して AWS にお問い合わせください。

## OpenSSL のインストール

まだ OpenSSL がインストールされていない場合は、このセクションの手順に従います。

Linux または Unix で OpenSSL をインストールするには

1. [OpenSSL: ソース、Tarballs](https://openssl.org/source/)(https://openssl.org/source/) にアクセスします。
2. 最新のソースをダウンロードし、パッケージを構築します。

Windows へ OpenSSL をインストールするには

1. [OpenSSL: バイナリディストリビューション](https://wiki.openssl.org/index.php/Binaries)(https://wiki.openssl.org/index.php/Binaries) にアクセスして、Windows バージョンをインストールできるサイトの一覧を参照してください。
2. 選択したサイトの指示に従って、インストールを開始します。
3. Microsoft Visual C++ 2008 再頒布可能パッケージをインストールするように求められ、それがまだシステムにインストールされていない場合は、ご使用の環境に適したダウンロードリンクを選択してください。「Microsoft Visual C++ 2008 再配布可能セットアップウィザード」の指示に従ってください。

### Note

Microsoft Visual C++ 2008 再頒布可能ファイルがシステムに既にインストールされているかどうか分からない場合は、最初に OpenSSL をインストールしてみてください。Microsoft Visual C++ 2008 再頒布可能ファイルがまだインストールされていない場合、OpenSSL インストーラーに警告が表示されます。インストールする OpenSSL のバージョンと一致するアーキテクチャ (32 ビットまたは 64 ビット) をインストールすることを確認します。

4. Microsoft Visual C++ 2008 再頒布可能ファイルをインストールしたら、ご使用の環境に適したバージョンの OpenSSL バイナリを選択し、ファイルをローカルに保存します。OpenSSL のセットアップウィザードを起動します。
5. 「OpenSSL のセットアップウィザード」の指示に従ってください。

## OpenSSL の設定

OpenSSL コマンドを使用する前に、OpenSSL がインストールされている場所に関する情報が含まれるように、オペレーティングシステムを構成する必要があります。



## Linux または Unix で OpenSSL を設定するには

1. コマンドラインで、OpenSSL\_HOME 変数を OpenSSL のインストール場所に設定します。

```
$ export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. OpenSSL インストールを含めるパスを設定します。

```
$ export PATH=$PATH:$OpenSSL_HOME/bin
```

### Note

export コマンドを使用して環境変数に加えた変更は、現在のセッションでのみ有効です。環境変数をシェル設定ファイルで設定することで、環境変数に永続的な変更を加えることができます。詳細については、「インスタンスのオペレーティングシステムに関するドキュメント」を参照してください。

## Windows での OpenSSL を設定するには

1. [コマンドプロンプト] ウィンドウを開きます。
2. OpenSSL\_HOME 変数を OpenSSL のインストール場所に設定します。

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. OpenSSL\_CONF 変数を OpenSSL インストールの設定ファイルの場所に設定します。

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. OpenSSL インストールを含めるパスを設定します。

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

### Note

コマンドプロンプトウィンドウで Windows 環境変数に加えた変更は、現在のコマンドラインセッションでのみ有効です。環境変数をシステムプロパティとして設定することで、環境変数を永続的に変更することができます。正確な手順は、使用している



Windows のバージョンによって異なります。(たとえば、Windows 7では、[コントロールパネル]、[システムとセキュリティ]、[システム]の順に開きます。次に、[システムの詳細設定]、[詳細設定]タブ、[環境変数]を選択します。) 詳細については、「Windows dドキュメント」を参照してください。

## OIDC フェデレーションに関するその他のリソース

OIDC フェデレーションの詳細を理解するのに、次のリソースが役に立ちます。

- [Amazon Web Services で OpenID Connect を設定](#) することによって GitHub ワークフロー内で OpenID Connect を使用します。
- 「Android ガイド」の「Amplify ライブラリ」にある [Amazon Cognito アイデンティティ](#) および 「Swift ガイド」の「Amplify ライブラリ」にある [Amazon Cognito アイデンティティ](#)
- AWS パートナーネットワーク (APN) ブログの「[Microsoft Entra ID を使用した OpenID Connect ベースの AWS IAM ウェブ ID ロールの自動化](#)」では、マシンツーマシン OIDC 認証を使用して、AWS の外部で実行する自動バックグラウンドプロセスまたはアプリケーションを認証する方法について説明しています。
- 記事「[モバイルアプリケーションを使用したウェブ ID フェデレーション](#)」では、OIDC フェデレーションについて説明し、OIDC フェデレーションを使用して Amazon S3 のコンテンツにアクセスする例を挙げています。

## SAML 2.0 フェデレーション

AWS は [SAML 2.0 \(Security Assertion Markup Language\)](#) を使用した ID フェデレーションをサポートします。これは、多くの ID プロバイダー (IdP) により使用されているオープンスタンダードです。この機能はフェデレーションシングルサインオン (SSO) を有効にします。したがって、組織内の全員について IAM ユーザーを作成しなくても、ユーザーは AWS Management Console にログインしたり、AWS API オペレーションを呼び出したりできるようになります。SAML を使用すると、[カスタム ID プロキシコードの書き込み](#)の代わりに IdP のサービスを使用できるため、AWS を使用してフェデレーションを構成するプロセスを簡素化できます。

IAM フェデレーションは次のユースケースをサポートします。

- [組織のユーザーまたはアプリケーションが AWS API オペレーションを呼び出すことを許可するフェデレーティッドアクセス](#)。このユースケースについては、次のセクションで説明します。組織で生成した SAML アサーションを (認証レスポンスの一部として) 使用して、一時的セキュリティ

認証情報を取得します。このシナリオは、「[一時的なセキュリティ認証情報のリクエスト](#)」および「[OIDC フェデレーション](#)」で説明されているような、IAM でサポートされている他のフェデレーションのシナリオに似ています。ただし、認証と認可のチェックの実行時は、組織の SAML 2.0 ベースの IdP によってその詳細の多くが処理されます。

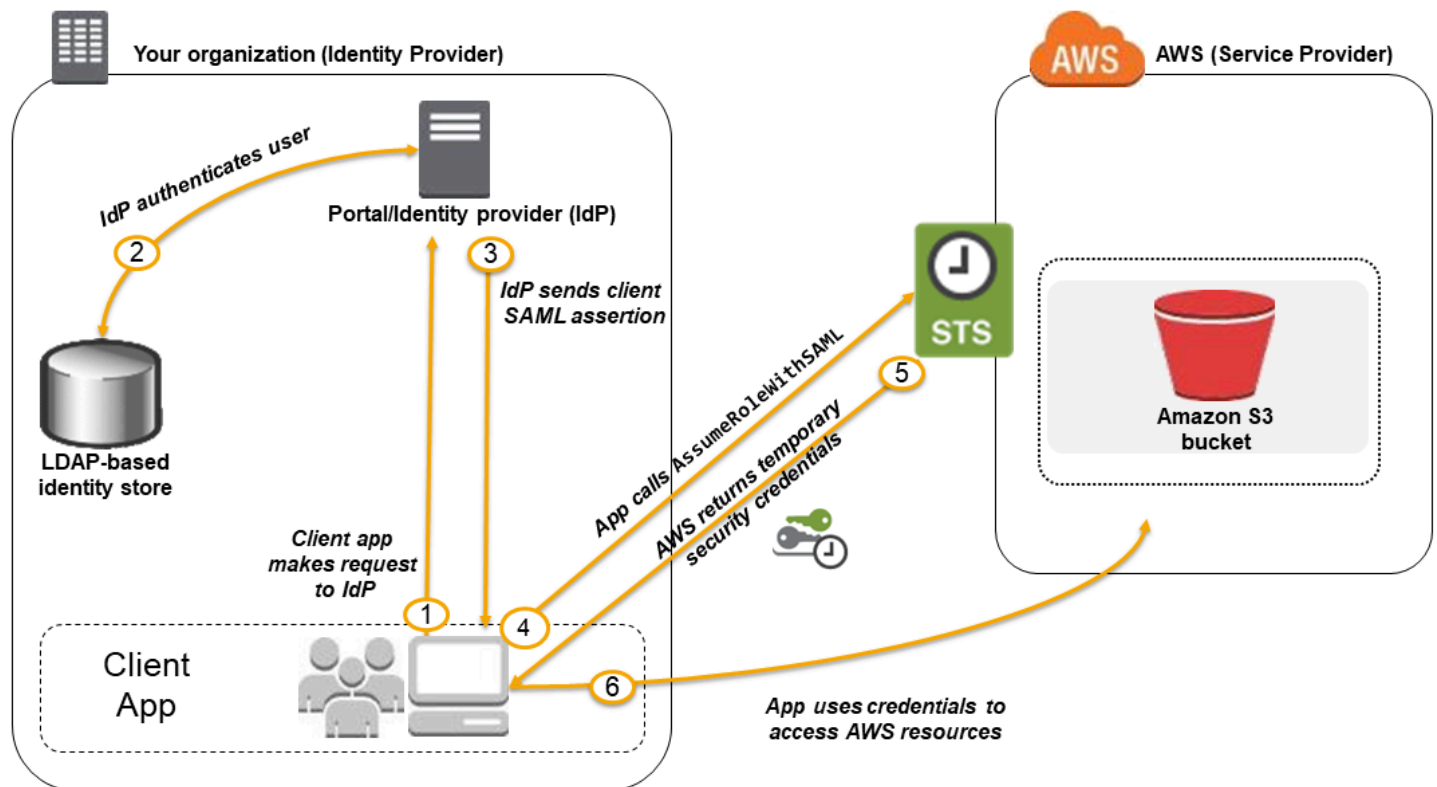
- [組織から AWS Management Console へのウェブベースのシングルサインオン \(SSO\)](#)。ユーザーが SAML 2.0 互換 IdP でホストされる組織のポータルにサインインして、AWS に移動するオプションを選択すると、追加でサインインの情報を入力しなくてもコンソールにリダイレクトされます。サードパーティーの SAML IdP を使用してコンソールへの SSO アクセスを確立するか、外部ユーザーのコンソールアクセスを有効にするカスタム IdP を作成することができます。カスタム IdP の構築の詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

## トピック

- [SAML ベースのフェデレーションを使用した AWS への API アクセス](#)
- [SAML 2.0 ベースのフェデレーション設定の概要](#)
- [AWS リソースへの SAML フェデレーションアクセスを許可するロールの概要](#)
- [SAML ベースのフェデレーションでユーザーを一意に識別する](#)
- [IAM で SAML ID プロバイダーを作成する](#)
- [証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する](#)
- [サードパーティーの SAML ソリューションプロバイダーを AWS に統合する](#)
- [認証レスポンス用の SAML アサーションを設定する](#)
- [SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする](#)

## SAML ベースのフェデレーションを使用した AWS への API アクセス

自分のコンピューターからバックアップフォルダにデータをコピーする方法を従業員に提供すると仮定します。ユーザーが、自分のコンピューターで実行できるアプリケーションを構築します。バックエンドで、アプリケーションは Amazon S3 バケットのオブジェクトを読み書きします。ユーザーは AWS に直接アクセスできません。代わりに、次のプロセスを使用します。



1. 組織のユーザーが、クライアントアプリを使用して、組織の IdP に認証を要求します。
2. IdP がユーザーを組織の ID ストアに対して認証します。
3. IdP はユーザーに関する情報を使用して SAML アサーションを構築し、クライアントアプリにアサーションを送信します。
4. クライアントアプリが、AWS STS [AssumeRoleWithSAML](#) API を呼び出して、SAML プロバイダーの ARN、引き受けるロールの ARN、および IdP からの SAML アサーションを渡します。
5. API は一時的なセキュリティ認証情報を含むレスポンスをクライアントアプリに返します。
6. クライアントアプリでは、一時的なセキュリティ証明書を使用して Amazon S3 API オペレーションを呼び出します。

## SAML 2.0 ベースのフェデレーション設定の概要

前述のシナリオと図に示すように SAML 2.0 ベースのフェデレーションを使用するには、事前に組織の IdP と AWS アカウント を設定して相互に信頼する必要があります。この信頼を設定するための一般的なプロセスを次の手順で説明します。組織内に、Microsoft Active Directory フェデレーションサービス (AD FS、Windows Server の一部)、Shibboleth など、[SAML 2.0 をサポートする IdP](#)、または互換性のある他の SAML 2.0 プロバイダーを用意する必要があります。

**Note**

フェデレーションの耐障害性を高めるには、IdP と AWS フェデレーションを、複数の SAML サインインエンドポイントをサポートするように設定することをお勧めします。詳細については、AWS セキュリティブログの記事「[フェイルオーバーにリージョン SAML エンドポイントを使用する方法](#)」を参照してください。

組織の IdP と AWS が互いを信頼するように設定する

1. 組織の IdP を使用し、サービスプロバイダー (SP) として AWS を登録します。 <https://region-code.signin.aws.amazon.com/static/saml-metadata.xml> から SAML メタデータドキュメントを使用する

実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。

任意で、<https://signin.aws.amazon.com/static/saml-metadata.xml> から SAML メタデータドキュメントが使用できます。

2. 組織の IdP を使用して、AWS の IAM ID プロバイダーとして IdP を記述できる同等のメタデータ XML ファイルを生成します。これには、発行元名、作成日、失効日、AWS が組織からの認証応答 (アサーション) を検証するために使用するキーを含める必要があります。
3. IAM コンソールで、SAML ID プロバイダーを作成します。このプロセスの一環として、組織の IdP によって生成された SAML メタデータドキュメントとを [Step 2](#) でアップロードします。詳細については、「[IAM で SAML ID プロバイダーを作成する](#)」を参照してください。
4. IAM で、1 つ以上の ロールを作成します。ロールの信頼ポリシーで SAML プロバイダーを、組織と AWS 間の信頼関係を確立するプリンシパルとして設定します。ロールのアクセス許可ポリシーは、AWS で組織のユーザーが実行できることを設定します。詳細については、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

**Note**

ロール信頼ポリシーで使用される SAML IDP は、そのロールと同じアカウントにある必要があります。

5. 組織の IdP で、組織のユーザーまたはグループを IAM ロールにマッピングするアサーションを定義します。組織内の異なるユーザーとグループは、異なる IAM ロールにマップされている場



```
"Action": "sts:AssumeRoleWithSAML",
"Condition": {
  "StringEquals": {
    "saml:aud": "https://signin.aws.amazon.com/saml",
    "saml:iss": "https://openidp.feide.no"
  },
  "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
}
}]
}
```

### Note

ロール信頼ポリシーで使用される SAML IDP は、そのロールと同じアカウントにある必要があります。

ポリシーでチェック可能な SAML キーの詳細については、「[SAML ベースの AWS STS フェデレーションに利用可能なキー](#)」を参照してください。

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` で `saml:aud` 属性のリージョナルエンドポイントを含めることができます。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。

ロールのアクセス許可ポリシーでは、ロールに付与するアクセス許可を自由に指定します。たとえば、組織のユーザーが Amazon Elastic Compute Cloud インスタンスを管理できる場合、AmazonEC2FullAccess 管理ポリシーのように、アクセス許可ポリシーで明示的に Amazon EC2 アクションを許可します。

## SAML ベースのフェデレーションでユーザーを一意に識別する

IAM でアクセスポリシーを作成するときに、ユーザーの ID に基づいてアクセス許可を指定できると、便利です。たとえば、SAML を使用してフェデレーションされたユーザーの情報は、アプリケーションで以下のような構造体を使用して Amazon S3 に保存することをお勧めします。

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
```



バケット (myBucket) およびフォルダ (app1) は静的な値であるため、Amazon S3 コンソールまたは AWS CLI で作成できます。ただし、ユーザー固有のフォルダ (*user1*、*user2*、*user3* など) は、ユーザーが最初にフェデレーションプロセスを通してサインインするまでユーザーを特定する値がわからないため、実行時にコードを使用して作成する必要があります。

リソース名の一部としてユーザー固有の詳細を参照するポリシーを記述するには、ユーザー ID がポリシー条件で使用できる SAML キーで利用可能である必要があります。IAM ポリシーで使用する SAML 2.0 ベースのフェデレーションでは、次のキーを使用できます。以下のキーによって返される値を使用して、Amazon S3 フォルダなどのリソースの一意のユーザー ID を作成できます。

- `saml:namequalifier.Issuer` のレスポンス値 (`saml:iss`)、AWS アカウント ID および IAM の SAML プロバイダーのわかりやすい名前 (ARN の最後の部分) の文字列を連結したハッシュ値。アカウント ID および SAML プロバイダーのわかりやすい名前の連結はキー `saml:doc` として、IAM のポリシーで使用できます。アカウント ID とプロバイダー名は、「123456789012/provider\_name」のように「/」で区切る必要があります。詳細については、「[saml:doc](#)」の [SAML ベースの AWS STS フェデレーションに利用可能なキー](#) キーを参照してください。

NameQualifier と Subject の組み合わせを使用して、フェデレーテッドユーザーを一意に識別できます。次の擬似コードは、この値の計算方法を示します。この擬似コードで + は連結を表し、SHA1 は SHA-1 を使用してメッセージダイジェストを生成する関数を、Base64 は Base-64 でエンコードされたバージョンのハッシュ出力を生成する関数を示します。

```
Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/"  
MySAMLIdP" ) )
```

SAML ベースのフェデレーションで利用可能なポリシーキーの詳細については、「[SAML ベースの AWS STS フェデレーションに利用可能なキー](#)」を参照してください。

- `saml:sub` (文字列)。\* これはクレームの件名です。組織内の個々のユーザーを一意に識別する値が含まれます (例: `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`)。
- `saml:sub_type` (文字列)。このキーは、persistent、transient、または SAML アサーションで使用されている Format および Subject 要素の完全な NameID URI とすることができます。persistent の値は、すべてのセッションでユーザーの `saml:sub` 値が同じことを意味します。値が transient の場合、ユーザーの `saml:sub` 値はセッションごとに異なります。NameID 要素の Format 属性の詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。

以下の例は前述のキーを使用して Amazon S3 のユーザー固有のフォルダにアクセス許可を付与するためのアクセスポリシーを示します。ポリシーは、Amazon S3 オブジェクトが `saml:namequalifier` と `saml:sub` の両方を含むプレフィックスを使用して特定されていることを前提としています。Condition 要素には、`saml:sub_type` が `persistent` に設定されていることを確認するテストが含まれることに注意してください。`transient` に設定されている場合、ユーザーの `saml:sub` 値はセッションごとに異なる可能性があるため、値の組み合わせを使用してユーザー固有のフォルダを識別することはできません。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}",
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
}
```

IdP からポリシーキーにアサーションをマッピングする方法については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。

## IAM で SAML ID プロバイダーを作成する

IAM SAML 2.0 ID プロバイダーは、[SAML 2.0 \(Security Assertion Markup Language 2.0\)](#) 基準をサポートする外部 ID プロバイダー (IdP) を記述する IAM のエンティティです。SAML 互換 IdP 間 (Shibboleth か Active Directory フェデレーションサービスと AWS など) の信頼を確立し、組織内のユーザーが AWS リソースにアクセスできるようにする場合は、IAM ID プロバイダーを使用します。IAM の SAML プロバイダーは IAM 信頼ポリシーでプリンシパルとして使用されます。

このシナリオの詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。

AWS Management Console または AWS CLI、Tools for Windows PowerShell、または AWS API 呼び出しを使用して IAM ID プロバイダーを作成および管理できます。



SAML プロバイダーを作成した後、IAM ロールを作成する必要があります。ロールは AWS のアイデンティティであり、それ自体には (ユーザーのような) 認証情報がありません。しかし、この例で、ロールが動的に割り当てられるフェデレーティッドユーザーは、組織の IdP から認証されます。このロールで、組織の IdP が AWS にアクセスするための一時的なセキュリティ認証情報をリクエストできるようにします。ロールに割り当てられているポリシーは、フェデレーティッドユーザーが AWS で実行できることを決定します。SAML フェデレーション用のロールを作成するには、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」を参照してください。

ロールを作成した後、最後に AWS およびフェデレーティッドユーザーが使用するロールに関する情報で IdP を設定し、SAML の信頼を完了します。これを、IdP と AWS 間の証明書利用者の設定といいます。証明書利用者の信頼を設定するには、「[証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する](#)」を参照してください。

## トピック

- [前提条件](#)
- [IAM SAML ID プロバイダーを作成および管理する \(コンソール\)](#)
- [IAM SAML ID プロバイダーを作成および管理する \(AWS CLI\)](#)
- [IAM SAML ID プロバイダーを作成および管理する \(AWS API\)](#)

## 前提条件

SAML ID プロバイダーを作成する前に、IdP から次の情報を取得しておく必要があります。

- IdP から SAML メタデータドキュメントを取得します。このドキュメントには発行者の名前、失効情報、およびキーが含まれており、これらを使用して、IdP から受け取った SAML 認証レスポンス (アサーション) を検証できます。メタデータドキュメントを生成するには、外部 IdP として提供されている ID 管理ソフトウェアを使用します。

### Important

このメタデータファイルには発行者の名前、失効情報、およびキーが含まれており、これらを使用して、IdP から受け取った SAML 認証レスポンス (アサーション) を検証できます。メタデータファイルはバイトオーダーマーク (BOM) なしで UTF-8 形式でエンコードする必要があります。BOM を削除するには、Notepad++ などのテキスト編集ツールを使用して UTF-8 としてファイルをエンコードできます。

SAML メタデータドキュメントの一部として含まれている x.509 証明書では、少なくとも 1024 ビットのキーサイズを使用する必要があります。また、x.509 証明書には、拡張領域

が繰り返されていないことが必要です。拡張領域は使用できますが、証明書に 1 回しか出現できません。x.509 証明書がいずれかの条件を満たしていない場合、IdP の作成は失敗し、「メタデータを解析できない」エラーが返されます。

[SAML V2.0 メタデータ相互運用性プロファイルバージョン 1.0](#) で定義されているように、IAM はメタデータドキュメントの X.509 証明書の有効期限を評価したりアクションを実行したりすることはできません。

必要な SAML メタデータドキュメントの生成方法を含め、使用可能な多くの IdP を AWS と連携するように設定するステップについては、「[サードパーティーの SAML ソリューションプロバイダーを AWS に統合する](#)」を参照してください。

SAML フェデレーションの詳細については、「[Troubleshooting SAML 2.0 federation with AWS](#)」を参照してください。

### IAM SAML ID プロバイダーを作成および管理する (コンソール)


AWS Management Console を使用して、IAM SAML ID プロバイダーの作成、更新、および削除を実行できます。SAML フェデレーションの詳細については、「[Troubleshooting SAML 2.0 federation with AWS](#)」を参照してください。

### IAM SAML ID プロバイダーを作成するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[Identity providers] (ID プロバイダー) を選択し、[Add provider] (プロバイダーを追加) を選択します。
3. [Configure provider] (プロバイダーの設定) で、[SAML] を選択します。
4. ID プロバイダーの名前を入力します。
5. [Metadata document] (メタデータドキュメント) で、[Choose file] (ファイルを選択) を選択し、[the section called “前提条件”](#) でダウンロードした SAML メタデータドキュメントを指定します。
6. (オプション) [Add tags (タグの追加)] では、キーバリューのペアを追加して IdP の特定と整理を行うことができます。タグを使用して、AWS リソースへのアクセスを制御することもできます。SAML ID プロバイダーのタグ付けの詳細については、「[IAM SAML ID プロバイダーのタグ付け](#)」を参照してください。

タグを追加 を選択します。タグキーバリューのペアごとに値を入力します。

7. 入力した情報を確認します。完了したら、[Add provider] (プロバイダーを追加) を選択します。
8. ID プロバイダーに IAM ロールを割り当てて、ID プロバイダーによって管理される外部ユーザー ID にアカウント内の AWS リソースへのアクセス許可を与えます。ID フェデレーション用のロールの作成の詳細については、「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールの作成](#)」をご参照ください。

 Note

ロール信頼ポリシーで使用される SAML IDP は、そのロールと同じアカウントにある必要があります。

### SAML プロバイダーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[Identity providers (ID プロバイダ)] を選択します。
3. 削除する ID プロバイダーの横にあるラジオボタンをオンにします。
4. [削除] を選択します。新しいウィンドウが開きます。
5. フィールドに単語 delete を入力して、プロバイダーを削除することを確認します。その後、[Delete] (削除) をクリックします。

### IAM SAML ID プロバイダーを作成および管理する (AWS CLI)

AWS CLI を使用して、SAML プロバイダーの作成、更新、および削除を実行できます。SAML フェデレーションの詳細については、「[Troubleshooting SAML 2.0 federation with AWS](#)」を参照してください。

### IAM ID プロバイダーを作成してメタデータドキュメントをアップロードするには (AWS CLI)

- 次のコマンドを実行します。 [aws iam create-saml-provider](#)

### IAM SAML ID プロバイダーを更新するには (AWS CLI)

- 次のコマンドを実行します。 [aws iam update-saml-provider](#)

既存の IAM ID プロバイダー (AWS CLI) にタグを付けるには

- 次のコマンドを実行します。 [aws iam tag-saml-provider](#)

既存の IAM ID プロバイダー (AWS CLI) のタグを一覧表示するには

- 次のコマンドを実行します。 [aws iam list-saml-provider-tags](#)

既存の IAM ID プロバイダー (AWS CLI) のタグを削除するには

- 次のコマンドを実行します。 [aws iam untag-saml-provider](#)

IAM SAML ID プロバイダーを削除するには (AWS CLI)

1. (オプション) すべてのプロバイダーに関する ARN、作成日、失効などの情報を表示するには、次のコマンドを実行します。

- [aws iam list-saml-providers](#)

2. (オプション) ARN、作成日、有効期限、暗号化の設定、プライベートキーの情報など、特定のプロバイダーに関する情報を取得するには、次のコマンドを実行します。

- [aws iam get-saml-provider](#)

3. IAM ID プロバイダーを削除するには、次のコマンドを実行します。

- [aws iam delete-saml-provider](#)

IAM SAML ID プロバイダーを作成および管理する (AWS API)

AWS API を使用して、SAML プロバイダーの作成、更新、および削除を実行できます。SAML フェデレーションの詳細については、「[Troubleshooting SAML 2.0 federation with AWS](#)」を参照してください。

IAM ID プロバイダーを作成してメタデータドキュメントをアップロードするには (AWS API)

- 呼び出すオペレーション: [CreateSAMLProvider](#)

IAM SAML ID プロバイダーを更新するには (AWS API)

- 呼び出すオペレーション: [UpdateSAMLProvider](#)

既存の IAM ID プロバイダーにタグを付けるには (AWS API)

- 呼び出すオペレーション: [TagSAMLProvider](#)

既存の IAM ID プロバイダーのタグを一覧表示するには (AWS API)

- 呼び出すオペレーション: [ListSAMLProviderTags](#)

既存の IAM ID プロバイダーのタグを削除するには (AWS API)

- 呼び出すオペレーション: [UntagSAMLProvider](#)

IAM ID プロバイダーを削除するには (AWS API)

1. (オプション) すべての IdP に関する ARN、作成日、失効などの情報を表示するには、次のオペレーションを呼び出します。

- [ListSAMLProviders](#)

2. (オプション) ARN、作成日、有効期限、暗号化の設定、プライベートキーの情報など、特定のプロバイダーに関する情報を取得するには、次のオペレーションを呼び出します。

- [GetSAMLProvider](#)

3. IdP を削除するには、次のオペレーションを呼び出します。

- [DeleteSAMLProvider](#)

## 証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する

IAM ID プロバイダー、および SAML アクセス用のロールを作成すると、外部 ID プロバイダー (IdP) の詳細とそのユーザーが許可されているアクションが AWS に通知されます。次のステップでは、IdP に対し、サービスプロバイダーとしての AWS について通知します。これは、IdP と AWS の間に証明書利用者の信頼を追加することと呼ばれます。証明書利用者の信頼を追加するための正

確なプロセスは、使用する IdP によって異なります。詳細については、使用している ID 管理ソフトウェアのドキュメントを参照してください。

多くの IdP が URL の指定を許可しています。IdP はこの URL から、証明書利用者の情報と証明書が含まれる XML ドキュメントを読み取ることができます。AWS については、<https://region-code.signin.aws.amazon.com/static/saml-metadata.xml> または <https://signin.aws.amazon.com/static/saml-metadata.xml> を使用します。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。

URL を直接指定できない場合は、XML ドキュメントを前述の URL からダウンロードし、ダウンロードした XML ドキュメントを IdP ソフトウェアにインポートします。

また、IdP で、AWS を証明書利用者として指定する適切なクレームルールを作成する必要があります。IdP が AWS エンドポイントに SAML 応答を送信すると、1 つ以上のクレームを含む SAML アサーションが含まれます。クレームとは、ユーザーとそのグループに関する情報です。クレームルールはその情報を SAML 属性にマッピングします。これにより、AWS が IAM ポリシー内でフェデレーテッドユーザーのアクセス許可を確認するのに必要な属性が、IdP からの SAML 認証レスポンスに確実に含まれます。詳細については、次のトピックを参照してください。

- [AWS リソースへの SAML フェデレーションアクセスを許可するロールの概要](#)。このトピックでは、IAM ポリシー内の SAML 固有のキーの使用について説明するほか、そのキーを使用して SAML フェデレーテッドユーザーの権限を制限する方法について説明します。
- [認証レスポンス用の SAML アサーションを設定する](#)。このトピックでは、ユーザーに関する情報を含む SAML クレームを設定する方法について説明します。クレームは SAML アサーションにバンドルされ、AWS に送信される SAML レスポンスに含まれます。AWS ポリシーによって必要とされる情報が、AWS が認識して使用できる形式で SAML アサーションに含まれていることを確認する必要があります。
- [サードパーティーの SAML ソリューションプロバイダーを AWS に統合する](#)。このトピックには、サードパーティーの組織から提供されている、ID ソリューションを AWS と統合する方法に関するドキュメントへのリンクが記載されています。

#### Note

フェデレーションの耐障害性を高めるには、IdP と AWS フェデレーションを、複数の SAML サインインエンドポイントをサポートするように設定することをお勧めします。詳細

については、AWS セキュリティブログの記事「[フェイルオーバーにリージョン SAML エンドポイントを使用する方法](#)」を参照してください。

## サードパーティーの SAML ソリューションプロバイダーを AWS に統合する

### Note

人間のユーザーが AWS にアクセスする際は、一時的な認証情報の使用を必須とすることをお勧めします。AWS IAM Identity Center の使用を検討したことのある場合 IAM Identity Center を使用すると、複数の AWS アカウント へのアクセスを一元的に管理できます。ユーザーには、割り当てられたすべてのアカウントに対する MFA で保護された Single Sign-On によるアクセスを、1つの場所から提供することができます。IAM Identity Center では、その内部でユーザー ID の作成および管理を行います。あるいは、既存の SAML 2.0 互換 ID プロバイダーにも簡単に接続することができます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM Identity Center とは?) を参照してください。

以下のリンクは、AWS フェデレーションで動作するようにサードパーティーの SAML 2.0 ID プロバイダー (IdP) ソリューションを設定するために役立ちます。

### Tip

AWS サポートエンジニアは、ビジネスおよびエンタープライズサポートプランを利用し、サードパーティー製のソフトウェアを一部の統合タスクで実行しているお客様をサポートできます。サポートされているプラットフォームおよびアプリケーションの最新のリストについては、「AWS サポート FAQ」の「[サポート済みのサードパーティーソフトウェア](#)」を参照してください。

ソリューション	詳細情報
Auth0	<a href="#">Amazon Web Services との統合</a> – Auth0 ドキュメント ウェブサイトにあるこのページには、AWS Management Console でシングルサインオン (SSO) を設定する方法を説明するリソースへのリンクがあり、JavaScript の例が含ま



ソリューション	詳細情報
	<p>れています。<a href="#">セッションタグ</a>を渡すように Auth0 を設定できます。詳細については、「<a href="#">Auth0 Announces Partnership with AWS for IAM Session Tags</a>」を参照してください。</p>
Microsoft Entra	<p><a href="#">チュートリアル: Microsoft Entra SSO と AWS シングルアカウントアクセスの統合</a> – Microsoft のウェブサイトにあるこのチュートリアルには、SAML フェデレーションを使用して Microsoft Entra (旧称: Azure AD) を ID プロバイダー (IdP) として設定する方法が記載されています。</p>
Centrify	<p><a href="#">AWS への SSO に SAML を使用するように Centrify を設定する</a> – Centrify のウェブサイトのこのページでは、AWS への SSO に SAML を使用するように Centrify を設定方法について説明しています。</p>
CyberArk	<p>CyberArk User Portal から SAML のシングルサインオン (SSO) 経由でログインしているユーザーが、Amazon Web Services (AWS) へアクセスできるように、<a href="#">CyberArk</a> を設定します。</p>
ForgeRock	<p><a href="#">ForgeRock アイデンティティプラットフォーム</a>は、AWS と統合されています。<a href="#">セッションタグ</a>を渡すように ForgeRock を設定できます。詳細については、「<a href="#">Attribute Based Access Control for Amazon Web Services</a>」を参照してください。</p>
Google Workspace	<p><a href="#">Amazon Web Services クラウドアプリケーション</a> – Google Workspace 管理者ヘルプサイトのこの記事は、AWS をサービスプロバイダーとし、Google Workspace を SAML 2.0 IdP として構成する方法について説明しています。</p>
IBM	<p><a href="#">セッションタグ</a>を渡すように IBM を設定できます。詳細については、「<a href="#">IBM Cloud Identity IDaaS one of first to support AWS session tags</a>」を参照してください。</p>



ソリューション	詳細情報
JumpCloud	<a href="#">Amazon AWS で Single Sign On (SSO) の IAM ロールを介してアクセス権を付与する</a> – JumpCloud ウェブサイトのこの記事では、AWS のために IAM ロールに基づいて SSO を設定して有効にする方法について説明します。
Matrix42	<a href="#">MyWorkspace 入門ガイド</a> – このガイドでは、AWS Identity サービスを Matrix42 MyWorkspace と統合する方法について説明します。
Microsoft Active Directory フェデレーションサービス (AD FS)	<a href="#">フィールドノート: AWS IAM Identity Center と Active Directory Federation Service の統合</a> – この AWS アーキテクチャブログの記事では、AD FS と AWS IAM Identity Center (IAM Identity Center) 間の認証フローについて説明しています。IAM Identity Center では、SAML 2.0 による ID フェデレーションがサポートされており、AD FS ソリューションとの統合が可能です。ユーザーは企業の認証情報を持つ IAM Identity Center ポータルにサインインできるので、IAM Identity Center で個別に認証情報を管理するオーバーヘッドを削減できます。また <a href="#">セッションタグ</a> を渡すように AD FS を設定できます。詳細については、「 <a href="#">Use attribute-based access control with AD FS to simplify IAM permissions management</a> 」を参照してください。
miniOrange	<a href="#">AWS の SSO</a> – miniOrange ウェブサイトのこのページでは、エンタープライズ向けの AWS への安全なアクセスと、AWS アプリケーションへのアクセスの完全コントロールを確立する方法が説明されています。
Okta	<a href="#">Okta を使用した Amazon Web Services コマンドラインインターフェイスの統合</a> – Okta サポートサイトのこのページから、AWS で使用するように Okta を設定する方法について参照できます。 <a href="#">セッションタグ</a> を渡すように Okta を設定できます。詳細については、「 <a href="#">Okta and AWS Partner to Simplify Access Via Session Tags</a> 」を参照してください。

ソリューション	詳細情報
Okta	<a href="#">AWS アカウントフェデレーション</a> – Okta ウェブサイトにあるこのセクションは、AWS の IAM Identity Center をセットアップして有効にする方法を説明します。
OneLogin	<a href="#">OneLogin Knowledgebase</a> で「 <b>SAML AWS</b> 」を検索し、単一ロールや複数ロールのシナリオで、OneLogin と AWS 間で IAM Identity Center の機能をセットアップする方法を説明する記事の一覧を表示します。 <a href="#">セッションタグ</a> を渡すように OneLogin を設定できます。詳細については、「 <a href="#">OneLogin and Session Tags: Attribute-Based Access Control for AWS Resources</a> 」を参照してください。
Ping Identity	<a href="#">PingFederate AWS Connector</a> – PingFederate AWS Connector の詳細を表示します。これは、シングルサインオン (SSO) と接続のプロビジョニングを簡単にセットアップするためのクイック接続テンプレートです。AWS との統合に関するドキュメントを読み、最新の PingFederate AWS Connector をダウンロードします。 <a href="#">セッションタグ</a> を渡すように Ping ID を設定できます。詳細については、「 <a href="#">Announcing Ping Identity Support for Attribute-Based Access Control in AWS</a> 」を参照してください。
RadiantLogic	<a href="#">Radiant Logic Technology Partners</a> – Radiant Logic の RadiantOne Federated Identity Service を AWS に統合すると、SAML ベースの SSO のための ID ハブを提供できます。
RSA	「 <a href="#">Amazon Web Services - RSA Ready 実装ガイド</a> 」には、AWS と RSA を統合するためのガイダンスが記載されています。SAML 設定の詳細については、「 <a href="#">Amazon Web Services - SAML My Page SSO Configuration - RSA Ready Implementation Guide</a> 」を参照してください。

ソリューション	詳細情報
Salesforce.com	<a href="#">Salesforce から AWS への SSO を設定する方法</a> – Salesforce.com 開発者サイトのこの操作手順記事では、Salesforce で ID プロバイダー (IdP) をセットアップし、AWS をサービスプロバイダーとして設定する方法が説明されています。
SecureAuth	<a href="#">AWS - SecureAuth SAML SSO</a> – SecureAuth ウェブサイトのこの記事では、SecureAuth アプライアンス用に SAML と AWS の統合をセットアップする方法について説明されています。
Shibboleth	<a href="#">AWS Management Console への SSO に Shibboleth を使用する方法</a> – AWS セキュリティブログのこのエントリには、Shibboleth をセットアップして AWS の ID プロバイダーとして設定する方法をステップバイステップで示したチュートリアルがあります。 <a href="#">セッションタグ</a> を渡すように Shibboleth を設定できます。

詳細については、AWS ウェブサイトの「[IAM パートナー](#)」ページを参照してください。

## 認証レスポンス用の SAML アサーションを設定する

組織内でユーザーの ID が確認されると、外部 ID プロバイダー (IdP) によって `https://region-code.signin.aws.amazon.com/saml` の AWS SAML エンドポイントに認証レスポンスが送信されます。潜在的な *region-code* の差し替えリストについては、「AWS Sign-In endpoints」(サインインエンドポイント) の [\[Region\]](#) (リージョン) 列を参照してください。このレスポンスは、[HTTP POST Binding for SAML 2.0](#) 標準に従った SAML トークンを含み、さらに以下の要素またはクレームを含む POST リクエストであることが必要です。これらのクレームを SAML 互換 IdP に設定します。これらのクレームの入力方法については、IdP のドキュメントを参照してください。

IdP が AWS へのクレームを含むレスポンスを送信すると、多くの受信クレームは AWS コンテキストキーにマッピングされます。これらのコンテキストキーは Condition 要素を使用して IAM ポリシーにチェックインすることができます。使用可能なマッピングの一覧は、「[SAML 属性の AWS 信頼ポリシーコンテキストキーへのマッピング](#)」に掲載されています。

## Subject および NameID

例を以下に示します。マークされた値を独自の値に置き換えます。SubjectConfirmation 属性と SubjectConfirmationData 属性の両方が含まれる NotOnOrAfter 要素を持つ Recipient 要素が 1 つだけ必要です。これらの属性には、AWS エンドポイント `https://region-code.signin.aws.amazon.com/saml` と一致する値が含まれます。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。AWS の値については、次の例に示すように `https://signin.aws.amazon.com/static/saml` も使用できます。

NameID 要素は、永続的な値、一時的な値で構成することも、IdP ソリューションから提供される完全な URI で構成することもできます。永続的な値は、NameID の値がセッション間のユーザーでも同じになることを意味します。値が一時的な場合、ユーザーの NameID 値はセッションごとに異なります。シングルサインオン操作では、次の識別子タイプをサポートします。

- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">_cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z"
Recipient="https://signin.aws.amazon.com/saml"/>
  </SubjectConfirmation>
</Subject>
```

**⚠ Important**

saml:aud コンテキストキーは SAML 受取人属性から取得します。この属性は、SAML バージョンの OIDC オーディエンスフィールドと言えるものだからです (例えば、accounts.google.com:aud)。

**PrincipalTag SAML 属性**

( オプション ) Name 属性が `https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}` に設定された Attribute 要素を使用できます。この要素を使用すると、SAML アサーションでセッションタグとして属性を渡すことができます。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

属性をセッションタグとして渡すには、タグの値を指定する AttributeValue 要素を含めます。たとえば、タグのキーバリューのペア Project = Marketing と CostCenter = 12345 を渡すには、次の属性を使用します。タグごとに個別の Attribute 要素を含めます。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

上記のタグを推移的として設定するには、Attribute 属性を Name に設定した別の `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys` 要素を含めます。これは、セッションタグを推移的として設定するオプションの多値属性です。推移タグは、SAML セッションを使用して AWS で別のロールを引き受けるときに保持されます。これは、[ロールの連鎖](#)と呼ばれます。たとえば、Principal タグ と CostCenter タグの両方を推移的として設定するには、次の属性を使用してキーを指定します。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>CostCenter</AttributeValue>
</Attribute>
```

## Role SAML 属性

Name 属性が `https://aws.amazon.com/SAML/Attributes/Role` に設定された Attribute 要素を使用できます。この要素には、IdP によってユーザーがマッピングされている IAM SAML ID プロバイダーおよびロールおよびを一覧表示する AttributeValue 要素が 1 つ以上含まれます。IAM ロールと IAM ID プロバイダーは、[AssumeRoleWithSAML](#) に渡される RoleArn パラメーターと PrincipalArn パラメーターと同じ形式でコンマ区切りの ARN のペアとして指定されます。この要素には、少なくとも 1 つのロールとプロバイダーのペア (AttributeValue 要素) を含める必要があり、複数のペアを含めることができます。要素に複数のペアを含める場合、ユーザーが WebSSO を使用して AWS Management Console にサインインすると、引き受けるロールを選択する画面が表示されます。

### Important

Name タグの Attribute 属性の値は大文字と小文字が区別されます。これは厳密に `https://aws.amazon.com/SAML/Attributes/Role` に設定する必要があります。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

## RoleSessionName SAML 属性

Name 属性が `https://aws.amazon.com/SAML/Attributes/RoleSessionName` に設定された Attribute 要素を使用できます。この要素には、ロールが引き継がれたときに発行される一時的な認証情報の識別子を提供する AttributeValue 要素が 1 つ含まれています。これを使用して、アプリケーションを使用しているユーザーに一時的な認証情報を関連付けることができます。この要素は、AWS Management Console でユーザー情報を表示するときに使用されます。AttributeValue 要素の値は 2~64 文字で、英数字、アンダースコア、および `. + = @ -` (ハイフン) のみを含めることができます。スペースを含めることはできません。通常、この値はユーザー ID (johndoe) またはメールアドレス (johndoe@example.com) です。ユーザーの表示名 (John Doe) のように、スペースを含む値とすることはできません。

**⚠ Important**

Name タグの Attribute 属性の値は大文字と小文字が区別されます。これは厳密に `https://aws.amazon.com/SAML/Attributes/RoleSessionName` に設定する必要があります。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

**SessionDuration SAML 属性**

( オプション ) Name 属性が `https://aws.amazon.com/SAML/Attributes/SessionDuration` に設定された Attribute 要素を使用できます。この要素には、ユーザーが新しい一時的な認証情報をリクエストする前に、ユーザーが AWS Management Console にアクセスできる時間を指定する 1 つの AttributeValue 要素が含まれます。値は、セッションの秒数を表す整数です。値の範囲は 900 秒 (15 分) から 43200 秒 (12 時間) です。この属性が存在しない場合は、認証情報は 1 時間有効です (DurationSeconds API の AssumeRoleWithSAML パラメータのデフォルト値)。

この属性を使用するには、AWS Management Console でコンソールのサインインウェブエンドポイントを通じて `https://region-code.signin.aws.amazon.com/saml` へのシングルサインオンアクセスを提供する SAML プロバイダーを設定する必要があります。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。任意で次の URL `https://signin.aws.amazon.com/static/saml` が使用できます。この属性が AWS Management Console にのみセッションを拡張することに注意してください。他の認証情報の存続期間を延長することはできません。ただし、AssumeRoleWithSAML API コール中に存在する場合は、セッション期間を短縮するために使用できます。呼び出しによって返される認証情報のデフォルトの有効期間は 60 分です。

また、SessionNotOnOrAfter 属性も定義されている場合は、2 つの属性の小さい方の値、SessionDuration または SessionNotOnOrAfter がコンソールセッションの最大期間を確立することにも注意してください。

コンソールセッションを拡張された期間有効にする場合、認証情報が侵害されるリスクが高まります。このリスクを軽減するには、IAM コンソールの [Role Summary] ページで、[Revoke Sessions]



を選択して、どのロールのアクティブなコンソールセッションもすぐに無効にできます。詳細については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

### Important

Name タグの Attribute 属性の値は大文字と小文字が区別されます。これは厳密に `https://aws.amazon.com/SAML/Attributes/SessionDuration` に設定する必要があります。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>1800</AttributeValue>
</Attribute>
```

## SourceIdentity SAML 属性

( オプション ) Name 属性が `https://aws.amazon.com/SAML/Attributes/SourceIdentity` に設定された Attribute 要素を使用できます。この要素には、1 つの IAM ロールを使用しているユーザーまたはアプリケーションの識別子を提供する AttributeValue 要素が含まれています。SAML セッションを使用して、[ルールチェーン](#)と呼ばれる AWS の別のロールを引き受ける場合、ソースIDの値は保持されます。ソース ID の値は、ロールセッション中に実行されるすべてのアクションのリクエストに存在します。設定される値は、ロールセッション中に変更できません。その後、管理者は AWS CloudTrail ログを使用して、ソース ID 情報をモニタリングおよび監査し、共有ロールでアクションを実行したユーザーを特定します。

AttributeValue 要素の値は 2~64 文字で、英数字、アンダースコア、および `., += @ -` (ハイフン)のみを含めることができます。スペースを含めることはできません。値は通常、ユーザー ID ( johndoe ) やメールアドレス ( johndoe@example.com ) など、ユーザーに関連付けられている属性です。ユーザーの表示名 (John Doe) のように、スペースを含む値とすることはできません。ソースアイデンティティの使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

### Important

SAML アサーションが [SourceIdentity](#) 属性を使用するように設定されている場合、信頼ポリシーにも `sts:SetSourceIdentity` アクションを含める必要があります。ソースアイデンティティの使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。



ソース ID 属性を渡すには、ソース ID の値を指定する AttributeValue 要素を含めます。たとえば、ソース ID DiegoRamirez を渡すには次の属性を使用します。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
  <AttributeValue>DiegoRamirez</AttributeValue>
</Attribute>
```

## SAML 属性の AWS 信頼ポリシーコンテキストキーへのマッピング

このセクションの表では、よく使用される SAML 属性や、それらと AWS の信頼ポリシー条件コンテキストキーのマッピングを一覧で示します。これらのキーを使用して、ロールへのアクセスを制御できます。そのためには、SAML アクセスリクエストに付随するアサーションに含まれる値とキーを比較します。

### Important

これらのキーは、IAM 信頼ポリシー (誰がロールを利用するかを定義するポリシー) でのみ利用でき、アクセス許可ポリシーには適用できません。

eduPerson および eduOrg 属性の表では、値は文字列または文字列のリストとして型付けされています。文字列値の場合、StringEquals または StringLike 条件を使用して、IAM 信頼ポリシーでこれらの値をテストできます。文字列のリストを含む値の場合、ForAnyValue および ForAllValues [ポリシー set 演算子](#)を使用して、信頼ポリシーで値をテストできます。

### Note

AWS コンテキストキーごとに含めることができるクレームは 1 つだけです。複数含めた場合は、1 つのクレームのみが対応付けられます。

## eduPerson 属性と eduOrg 属性

eduPerson 属性または eduOrg 属性 (Name キー)	この AWS コンテキストキー (FriendlyName キー) へのマッピング	タイプ
urn:oid:1.3.6.1.4.1.5923.1.1.1.1	eduPerson Affiliation	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.2	eduPersonNickname	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.3	eduPersonOrgDN	文字列
urn:oid:1.3.6.1.4.1.5923.1.1.1.4	eduPerson OrgUnitDN	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.5	eduPerson PrimaryAffiliation	文字列
urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eduPerson PrincipalName	文字列
urn:oid:1.3.6.1.4.1.5923.1.1.1.7	eduPerson Entitlement	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPerson PrimaryOrgUnitDN	文字列
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPerson ScopedAffiliation	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.10	eduPerson TargetedID	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.1.1.11	eduPerson Assurance	文字列のリスト

eduPerson 属性または eduOrg 属性 (Name キー)	この AWS コンテキストキー (FriendlyName キー) へのマッピング	タイプ
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIdentityAuthNPolicyURI	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	文字列のリスト
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhitePagesURI	文字列のリスト
urn:oid:2.5.4.3	cn	文字列のリスト

### Active Directory の属性

AD 属性	この AWS コンテキストキーへのマッピング	タイプ
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	文字列
http://schemas.xmlsoap.org/claims/CommonName	commonName	文字列
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	文字列
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	文字列

AD 属性	この AWS コンテキストキーへのマッピング	タイプ
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	文字列
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	文字列

### X.500 属性

X.500 属性	この AWS コンテキストキーへのマッピング	タイプ
2.5.4.3	commonName	文字列
2.5.4.4	surname	文字列
2.4.5.42	givenName	文字列
2.5.4.45	x500UniqueIdentifier	文字列
0.9.2342.19200300100.1.1	uid	文字列
0.9.2342.19200300100.1.3	mail	文字列
0.9.2342.19200300.100.1.45	organizationStatus	文字列

## SAML 2.0 フェデレーテッドユーザーが AWS Management Console にアクセス可能にする

フェデレーテッドユーザーが AWS Management Console にアクセスできるように、SAML 2.0 互換 ID プロバイダー (IdP) および AWS を設定するロールを使用します。このロールは、コンソールでタスクを実行するユーザーアクセス権限を付与します。AWS にアクセスするためのその他の方法を SAML フェデレーテッドユーザーに付与する場合は、以下のトピックの 1 つを参照してください。

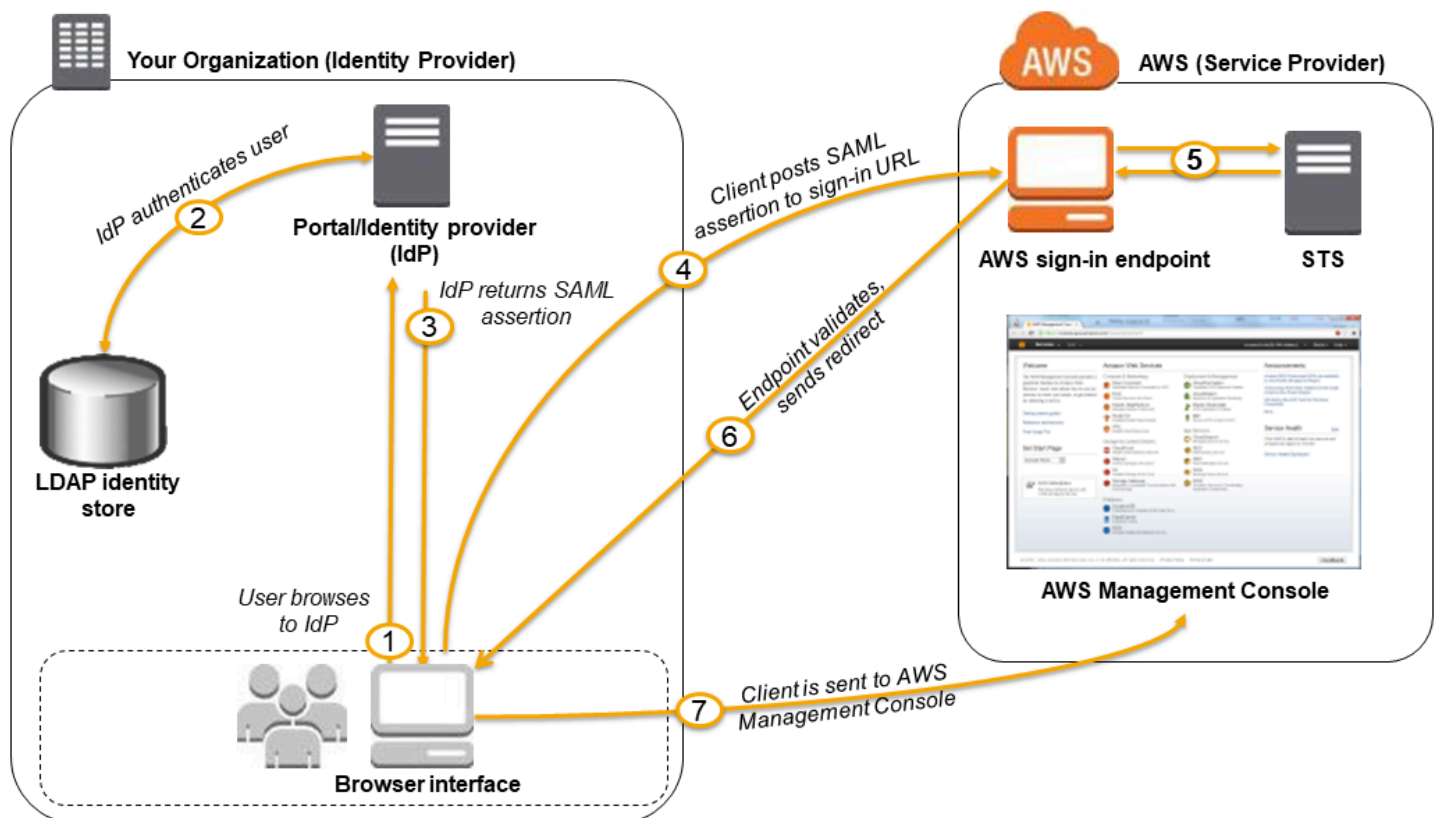
- AWS CLI: [IAM ロール \(AWS CLI\) の切り替え](#)
- [IAM ロールへの切り替え \(Tools for Windows PowerShell\)](#) Tools for Windows PowerShell
- AWS API: [IAM ロール \(AWS API\) の切り替え](#)

## 概要

次の図は、SAML 対応のシングルサインオンについて処理の流れを示しています。

### Note

この SAML の使用方法の場合、ワークフローでユーザーに代わって AWS Management Consoleを開くため、[SAML 2.0 フェデレーション](#) に示されている一般的な使用方法とは異なります。これには、AssumeRoleWithSAML API を直接呼び出す代わりに、AWS サインインエンドポイントを使用する必要があります。エンドポイントはユーザーの代わりに API を呼び出し、URL を返すと、それによってユーザーのブラウザが AWS Management Consoleへ自動的にリダイレクトされます。



この図表は以下のステップを示しています。

1. ユーザーは組織のポータルにアクセスして、AWS Management Console に移動するオプションを選択します。一般的に、組織のポータルは、組織と AWS 間の信頼の交換を処理する IdP の機能です。たとえば、Active Directory フェデレーションサービスでは、ポータル URL は次のようになります。https://*ADFSServiceName*/adfs/ls/IdpInitiatedSignOn.aspx
2. ポータルが組織内のユーザーの ID を確認します。
3. ポータルは、ユーザーを識別するアサーションを含む SAML アサーションレスポンスを生成し、ユーザーの属性を含めます。コンソールセッションが有効な期間を指定する SessionDuration という SAML アサーションの属性を含むよう IdP を設定できます。[セッションタグ](#)として属性を渡すように IdP を設定することもできます。ポータルはこのレスポンスをクライアントブラウザに送信します。
4. クライアントブラウザは AWS のシングルサインオンエンドポイントにリダイレクトされ、SAML アサーションを投稿します。
5. エンドポイントは、ユーザーの代わりに一時的なセキュリティ認証情報をリクエストし、コンソールのサインイン URL を作成します。
6. AWS は、サインイン URL をクライアントにリダイレクトとして送信します。
7. クライアントのブラウザは AWS Management Console にリダイレクトされます。複数の IAM ロールに対応付けられた属性を SAML 認証レスポンスが含む場合は、最初にコンソールへのアクセスするためのロールを選択する画面が表示されます。

ユーザーの立場では、この処理を意識することはありません。ユーザーは組織の内部ポータルから AWS Management Console に移動するだけで、AWS 認証情報を指定する必要はありません。

以下のセクションでは、この動作を設定する方法の概要と、詳細なステップへのリンクをご紹介します。

## ネットワークを AWS の SAML プロバイダーとして設定する

組織のネットワーク内で、組織の ID ストア (Windows Active Directory など) が、Windows Active Directory Federation Services や Shibboleth などの SAML ベースの IdP と連携するように設定します。IdP を使用して、組織を IdP として記述し、認証キーを含むドキュメントメタデータを生成します。また、AWS Management Console ルートユーザーに対するユーザーリクエストを AWS SAML エンドポイントにルーティングして、SAML アサーションを使って認証するように、組織のポータルを設定します。metadata.xml ファイルを生成するための IdP の設定方法は、IdP によって異なります。手順については IdP の文書を参照してください。また、[サードパーティーの SAML ソリュー](#)

[シオンプロバイダーを AWS に統合する](#) には、サポートされる数多くの SAML プロバイダーのウェブドキュメントへのリンクが掲載されています。

IAMで SAML プロバイダーを作成するには

次に、AWS Management Console にサインインし、IAM コンソールへ移動します。ここで、新しい SAML プロバイダーを作成します。これは、組織の IdP に関する情報を保持する IAM のエンティティです。このプロセスの一環として、前のセクションで組織の IdP ソフトウェアによって生成されたメタデータドキュメントをアップロードします。詳細については、「[IAM で SAML ID プロバイダーを作成する](#)」を参照してください。

フェデレーションユーザーのアクセス許可を AWS で設定する

次のステップでは、IAM と組織の IdP の間の信頼関係を確立する IAM ロールを作成します。このロールは、フェデレーションの目的で IdP をプリンシパル (信頼されたエンティティ) として識別します。ロールは、組織の IdP によって認証されたユーザーが AWS で何を実行できるかも定義します。このロールは、IAM コンソールを使用して作成できます。ロールを引き受けることができるユーザーを示す信頼ポリシーを作成するときは、前述の IAM で作成した SAML プロバイダーを指定します。また、ユーザーがロールを引き受けるために一致する必要がある 1 つ以上の SAML 属性も指定します。たとえば、SAML の [eduPersonOrgDN](#) 値が ExampleOrg であるユーザーのみにサインインを許可するように指定できます。ロールウィザードは、そのロールが AWS Management Console へのサインインだけで引き受けられるように、`saml:aud` 属性をテストする条件を自動的に追加します。ロールの信頼ポリシーは次のようなものです。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {"StringEquals": {
      "saml:edupersonorgdn": "ExampleOrg",
      "saml:aud": "https://signin.aws.amazon.com/saml"
    }}
  ]
}
```

**Note**

ロール信頼ポリシーで使用される SAML IDP は、そのロールと同じアカウントにある必要があります。

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` で `saml:aud` 属性のリージョンナルエンドポイントを含めることができます。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。

ロールの[アクセス許可ポリシー](#)では、任意のロール、ユーザー、グループに付与するアクセス許可を指定します。たとえば、組織のユーザーが Amazon EC2 インスタンスを管理することを許可する場合、アクセス許可ポリシーで明示的に Amazon EC2 アクションを許可します。これは、Amazon EC2 Full Access 管理ポリシーなどの[管理ポリシー](#)を割り当てることでも行えます。

SAML IdP のロールの作成に関する詳細については、「[SAML 2.0 フェデレーション用のロールを作成する \(コンソール\)](#)」を参照してください。

#### 設定の完了と SAML アサーションの作成

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` または `https://signin.aws.amazon.com/static/saml-metadata.xml` にある `saml-metadata.xml` ファイルをインストールし、AWS がお客様のサービスプロバイダーであることを SAML IdP に通知します。実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。

ファイルのインストール方法は IdP によって異なります。プロバイダーによっては、URL の入力を選択できる場合があります。この場合、IdP がお客様の代わりにファイルを取得してインストールします。また、URL からファイルをダウンロードし、ローカルファイルとして指定する必要があるプロバイダーもあります。詳細については IdP の文書を参照してください。また、[サードパーティーの SAML ソリューションプロバイダーを AWS に統合する](#) には、サポートされる数多くの SAML プロバイダーのウェブドキュメントへのリンクが掲載されています。

また、認証レスポンスの一部として、IdP から AWS へ SAML 属性として渡す情報も設定します。この情報のほとんどは、ポリシーで評価できる条件コンテキストキーとして AWS に表示されます。これらの条件キーにより、適切なコンテキストで許可されたユーザーのみに、AWS リソースにアクセスするアクセス許可が付与されます。コンソールを使用するタイミングを制限する時間ウィンドウを



指定できます。ユーザーが認証情報を更新する前にコンソールにアクセスできる最大時間 ( 最大 12 時間 ) を指定できます。詳細については、[認証レスポンス用の SAML アサーションを設定する](#) を参照してください

## IAM の一時的な認証情報

AWS Security Token Service (AWS STS) を使用して、AWS リソースへのアクセスをコントロールできる一時的セキュリティ認証情報を持つ、信頼されたユーザーを作成および提供することができます。一時的セキュリティ認証情報の機能は、長期的なアクセスキー認証情報とほとんど同じですが、次の相違点があります。

- 一時的セキュリティ認証情報は、その名前が示すとおり、使用期限が短くなっています。有効期限は数分から数時間に設定できます。認証情報が失効すると、AWS はそれらを認識しなくなります。また、その認証情報によって作成された API リクエストによるあらゆるタイプのアクセスが許可されなくなります。
- 一時的セキュリティ認証情報はユーザーとともに保存されることはなく、ユーザーのリクエストに応じて動的に生成され、提供されます。一時的セキュリティ認証情報が失効すると ( または失効する前でも )、ユーザーは新しい認証情報をリクエストできます。ただし、リクエストするユーザーがまだその権限を持っている場合に限りです。

そのため、一時的な認証情報には、長期の認証情報よりも次の利点があります。

- アプリケーションの長期の AWS セキュリティ認証情報を配布したり埋め込んだりする必要がありません。
- ユーザーに対して AWS ID を定義せずに AWS リソースへのアクセスを許可できます。一時的認証情報は [ロール](#) および [ID フェデレーション](#) の基本となります。
- 一時的セキュリティ認証情報の有効期限は限られているので、認証情報が不要になった際に更新したり、明示的に取り消したりする必要がありません。一時的セキュリティ認証情報の有効期限が切れると、再利用することはできません。認証情報が有効な期間を、最大限度まで指定できます。

## AWS STS と AWS リージョン

一時的な認証情報は AWS STS によって生成されます。デフォルトでは、AWS STS は `https://sts.amazonaws.com` に 1 つのエンドポイントのあるグローバルサービスです。ただし、他のサポートされているリージョンにあるエンドポイントへの AWS STS API 呼び出しを実行することもできます。地理的に近い場所にあるリージョンのサーバーに対してリクエストを送信することによっ

て、レイテンシー (サーバーのラグ) を低減できます。認証情報を取得したリージョンに関係なく、認証情報はグローバルに使用できます。詳細については、「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

## 一時的な認証情報の一般的なシナリオ

一時的な認証情報は、ID フェデレーション、委任、クロスアカウントアクセス、および IAM ロールが使用されるシナリオで便利です。

### ID フェデレーション

AWS 以外の外部システムでユーザー ID を管理し、それらのシステムのアクセス許可からサインインするユーザーに、AWS タスクの実行や AWS リソースへのアクセスの権限を付与できます。IAM では、2 種類のアイデンティティフェデレーションがサポートされます。いずれの場合も、ID は AWS の外部に格納されます。異なる点は、外部システムが存在する場所 (データセンターまたはウェブの外部サードパーティ) です。外部 ID プロバイダーについては、「[ID プロバイダーとフェデレーション](#)」を参照してください。

- SAML フェデレーション - 組織のネットワークのユーザーを認証し、新しい AWS ID を作成したり、異なるサインイン認証情報でサインインすることを求めたりすることなく、それらのユーザーに AWS へのアクセスを提供できます。これは、一時アクセスに対するシングルサインオンのアプローチとして知られています。AWS STS は Security Assertion Markup Language (SAML) 2.0 のようなオープンスタンダードをサポートしており、Microsoft AD FS を通じて Microsoft Active Directory を活用することが可能です。また、SAML 2.0 を使用して、ユーザー ID フェデレーション用の独自のソリューションを管理することもできます。詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。
- カスタムフェデレーションブローカー - AWS組織の認証システムを使用して リソースへのアクセスを許可することができます。シナリオの例については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。
- SAML 2.0 を使用したフェデレーション - AWS 組織の認証システムと SAML を使用して、リソースへのアクセスを許可することができます。詳細とシナリオの例については、「[SAML 2.0 フェデレーション](#)」を参照してください。
- OpenID Connect (OIDC) フェデレーション - モバイルまたはウェブアプリケーションに対して、ユーザーに一般的なサードパーティー ID プロバイダー (Login with Amazon、Facebook、Google、OIDC 2.0 互換の任意のプロバイダーなど) を使用したサインインを求めることができます。カスタムサインインコードを作成したり、独自のユーザー ID を管理したりする必要はありません。OIDC フェデレーションを使用すると、IAM ユーザーアクセスキーな

どの長期的セキュリティ認証情報をアプリケーションに配布する必要がないため、AWS アカウントを安全に保つことができます。詳細については、「[OIDC フェデレーション](#)」を参照してください。

AWS STS OIDC フェデレーションでは、Login with Amazon、Facebook、Google、および任意の OpenID Connect (OIDC) 互換の ID プロバイダーがサポートされています。

#### Note

モバイルアプリケーションに対しては、Amazon Cognito の使用をお勧めします。このサービスをモバイル開発用の AWS SDK と共に使用して、ユーザーの一意の ID を作成し、AWS リソースへの安全なアクセスのためにユーザーを認証できます。Amazon Cognito では、AWS STS と同じ ID プロバイダーがサポートされます。さらに、認証されていない (ゲスト) アクセスもサポートされ、ユーザーがサインインしたときにユーザーデータを移行することができます。また、Amazon Cognito には、ユーザーがデバイスを変えてもデータが保持されるように、ユーザーデータを同期するための API オペレーションも用意されています。詳細については、Amplify ドキュメントの「[Amplify による認証](#)」を参照してください。

## クロスアカウントアクセスのロール

多くの組織は、複数の AWS アカウント を保持しています。ロールとクロスアカウントアクセスを使用すると、1つのアカウントでユーザー ID を定義し、その ID を使用して、組織に属している他のアカウントの AWS リソースにもアクセスできるようにすることができます。これは、一時アクセスに対する委任アプローチとして知られています。クロスアカウントロールの作成の詳細については、「[IAM ユーザーにアクセス許可を委任するロールの作成](#)」を参照してください。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

## Amazon EC2 の ロール

Amazon EC2 インスタンスでアプリケーションを実行し、これらのアプリケーションが AWS リソースにアクセスする必要がある場合は、アプリケーションの起動時に一時的セキュリティ認証情報をインスタンスに提供できます。これらの一時的なセキュリティ認証情報は、インスタンスで実行されるすべてのアプリケーションが使用できるので、インスタンスに長期的な認証情報を保存する必要がありません。詳細については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

## その他の AWS サービス

一時的セキュリティ認証情報を使用して、ほとんどの AWS サービスにアクセスできます。一時的セキュリティ認証情報を受け入れるサービスのリストは、「[IAM と連携する AWS のサービス](#)」を参照してください。

## 一時的なセキュリティ認証情報のリクエスト

一時的なセキュリティ認証情報をリクエストするには、AWS API で AWS Security Token Service (AWS STS) を使用できます。これには、AWS リソースへのアクセスを制御できる一時的セキュリティ認証情報を持つ、信頼されたユーザーを作成および提供するオペレーションが含まれます。AWS STS の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。ロールを引き受けることで一時的なセキュリティ認証情報をリクエストするための別の方法については、「[IAM ロールを使用する](#)」を参照してください。

API オペレーションを呼び出すには、いずれかの [AWS SDK](#) を使用することができます。SDK は、Java、.NET、Python、Ruby、Android、iOS など、さまざまなプログラミング言語や環境で使用できます。SDK は、リクエストへの暗号を使用した署名、必要に応じてリクエストの再試行、エラーレスポンスの処理などのタスクを処理します。また、[AWS Security Token Service API リファレンス](#) で説明されている AWS STS Query API を使用することもできます。最後に、AWS STS コマンドは [AWS Command Line Interface](#) および [AWS Tools for Windows PowerShell](#) という 2 つのコマンドラインツールでサポートされています。

AWS STS API オペレーションは、アクセスキーペアやセキュリティトークンを含む一時的セキュリティ認証情報で新しいセッションを作成します。アクセスキーペアは、アクセスキー ID とシークレットキーで構成されます。ユーザー（またはユーザーが実行しているアプリケーション）はこれらの認証情報を使用して、リソースにアクセスできます。AWS STS API オペレーションを使用して、ロールセッションを作成し、プログラムでセッションポリシーとセッションタグを渡すことができます。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーの詳細については、「[セッションポリシー](#)」を参照してください。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

### Note

AWS STS API オペレーションから返されるセッショントークンのサイズは固定ではありません。最大サイズを仮定しないことを強くお勧めします。一般的なトークンのサイズは 4096 バイト未満ですが、変化する可能性があります。

## AWS リージョンでの AWS STS の使用

AWS STS API 呼び出しは、グローバルエンドポイントにも、リージョンのエンドポイントの 1 つに対しても送信できます。より近くのエンドポイントを選択した場合、レイテンシーを軽減し、API 呼び出しのパフォーマンスが向上します。また、元のエンドポイントとの通信ができなくなった場合は、代替リージョンのエンドポイントに呼び出しを送信することもできます。各種 AWS SDK の 1 つを使用している場合、API コールを行う前に SDK メソッドを使用してリージョンを選択します。手動で HTTP API リクエストを組み立てる場合、独自で正しいエンドポイントにリクエストを送信する必要があります。詳細については、[リージョンとエンドポイントの「AWS STS」セクション](#)および「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

AWS 環境およびアプリケーションで使用する一時的な認証情報の取得に使用できる API オペレーションを、次に示します。

### [AssumeRole](#) — クロスアカウントの委任とカスタム ID ブローカーを介したフェデレーション

この AssumeRole API オペレーションは、既存の IAM ユーザーに、まだアクセスできない AWS リソースへのアクセスを許可する際に役立ちます。例えば、ユーザーが別の AWS アカウントのリソースにアクセスする必要がある場合があります。また、特権アクセスを一時的に得るための方法 (多要素認証 (MFA) など) としても役立ちます。アクティブなユーザーの認証情報を使用してこの API を呼び出す必要があります。誰がこの操作を呼び出すことができるのかについては、[AWS STS API オペレーションの比較](#)を参照してください。詳細については、[IAM ユーザーにアクセス許可を委任するロールの作成](#)および[MFA 保護 API アクセスの設定](#)を参照してください。

この呼び出しは、有効な AWS セキュリティ認証情報を使用して実行される必要があります。この呼び出しを行うときに、以下の情報を渡します。

- アプリが引き受ける必要のあるロールの Amazon リソースネーム (ARN)。
- ( オプション ) 一時的なセキュリティ認証情報の期間を指定する期間。DurationSeconds パラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。このパラメータを渡さない場合、一時的な認証情報の有効期限は 1 時間です。この API の DurationSeconds パラメータは、コンソールセッションの期間を指定する SessionDuration HTTP パラメータとは別のものです。コンソールのサインイントークンの場合は、フェデレーションエンドポイントに対するリクエストで SessionDuration HTTP パラメータを使用します。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。



- ロールセッション名 ロールが異なるプリンシパルによって使用される場合にセッションを識別するには、この文字列値を使用します。セキュリティ上の理由から、管理者は [AWS CloudTrail ログ](#)内のこのフィールドを表示し、AWS でアクションを実行したユーザーを確認できます。管理者が、ロールを引き受けるときにセッション名として IAM ユーザー名を指定するように要求している場合があります。詳細については、「[sts:RoleSessionName](#)」を参照してください。
- (オプション) 送信元アイデンティティ。ユーザーがロールを引き受けるときに、ソース ID を指定するように要求できます。ソース ID を設定した後は、値を変更できません。これは、ロールセッション中に実行されるすべてのアクションのリクエストに存在します。ソース ID 値は [ロールの連鎖](#)セッション間で持続します。ソース ID 情報は、AWS CloudTrail ログを使用して、ロールでアクションを実行したユーザーを特定します。ソースアイデンティティの使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。
- (オプション) インラインまたはマネージド ポリシーセッション。これらのポリシーでは、ロールセッションに割り当てられているロールのアイデンティティベースのポリシーのアクセス許可を制限しています。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。
- (オプション) セッションタグ。ロールを引き受け、一時的な認証情報を使用してリクエストを行うことができます。この場合、セッションのプリンシパルタグには、ロールのタグと渡されたセッションタグが含まれます。一時的な認証情報を使用してこの呼び出しを行うと、新しいセッションは呼び出し元のセッションから推移的なセッションタグも継承します。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。
- (オプション) MFA 情報。Multi-Factor Authentication (MFA) を使用するように設定されている場合、MFA デバイスの ID と、このデバイスによって提供されるワンタイムコードを含めます。
- (オプション) サードパーティーへのアカウントにアクセス権を委任するときに使用できる ExternalId 値。この値を利用して、指定されたサードパーティーだけがロールにアクセスできるようにします。詳細については、「[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#)」を参照してください。

以下の例は、AssumeRole を使用したリクエストと応答のサンプルを示します。このサンプルリクエストは、含まれている [セッションポリシー](#)、[セッションタグ](#)、[外部 ID](#)、および [ソース ID](#) を使用して、指定された期間の demo ロールを引き受けます。結果のセッションには John-session という名前が付けられます。

## Example リクエストの例

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=AssumeRole  
&RoleSessionName=John-session  
&RoleArn=arn:aws::iam::123456789012:role/demo  
&Policy=%7B%22Version%22%3A%22012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A%20%22Stmnt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A*%22%2C%22Resource%22%3A%20%22*%22%7D%5D%7D  
&DurationSeconds=1800  
&Tags.member.1.Key=Project  
&Tags.member.1.Value=Pegasus  
&Tags.member.2.Key=Cost-Center  
&Tags.member.2.Value=12345  
&ExternalId=123ABC  
&SourceIdentity=DevUser123  
&AUTHPARAMS
```

上記の例に示すポリシーの値は、次のポリシーの URL エンコードされたバージョンです。

```
{"Version":"2012-10-17","Statement":  
[{"Sid":"Stmnt1","Effect":"Allow","Action":"s3:*","Resource":"*"}]}
```

この例の AUTHPARAMS パラメータは 署名のプレースホルダーです。署名は AWS HTTP API リクエストに含める必要がある認証情報です。[AWS SDK](#) を使用して API リクエストを作成することをお勧めします。その利点の 1 つは、SDK がリクエストの署名を処理することです。API リクエストを手動で作成し、署名する必要がある場合は、[Amazon Web Services 全般のリファレンス] の「[署名バージョン 4 を使用して AWS リクエストに署名する](#)」を参照してリクエストに署名する方法を確認してください。

一時的セキュリティ認証情報に加えて、レスポンスにはフェデレーテッドユーザーの Amazon リソースネーム (ARN)、および認証情報の有効期限が含まれています。

## Example レスポンスの例

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
<AssumeRoleResult>  
<SourceIdentity>DevUser123</SourceIdentity>  
<Credentials>  
<SessionToken>  
AQoDYXdzEPT//////////wEXAMPLEtc764bNrc9SAPBSM22wD0k4x4HIZ8j4FZTWdQW
```

```

LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTf1fKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
9HFv1Rd8Tx6q6fE8YQcHNvXAKiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL641IZbqBAz
+scqKmlzm8FDrypNC9Yjc8fP0Ln9FX9KSYvKTr4rvx3iS1LTJabIQwj2ICCR/oLxBA==
</SessionToken>
<SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
</SecretAccessKey>
<Expiration>2019-07-15T23:28:33.359Z</Expiration>
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
<AssumedRoleUser>
  <Arn>arn:aws:sts::123456789012:assumed-role/demo/John</Arn>
  <AssumedRoleId>AR0123EXAMPLE123:John</AssumedRoleId>
</AssumedRoleUser>
<PackedPolicySize>8</PackedPolicySize>
</AssumeRoleResult>
<ResponseMetadata>
  <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</AssumeRoleResponse>

```

### Note

AWS 変換では、渡されたセッションポリシーとセッションタグが、個別の制限を持つひとまとめのバイナリ形式に圧縮されます。プレーンテキストが他の要件を満たしていても、この制限ではリクエストが失敗する可能性があります。PackedPolicySize レスポンス要素は、リクエストのポリシーとタグがサイズ制限にどの程度近づいているかをパーセントで示します。

## [AssumeRoleWithWebIdentity](#) — ウェブベースの ID プロバイダーを介したフェデレーション

AssumeRoleWithWebIdentity API オペレーションでは、パブリック ID プロバイダーを経由して認証されたフェデレーテッドユーザーの一時的なセキュリティ認証情報のセットが返ります。パブリック ID プロバイダーの例としては、Login with Amazon、Facebook、Google、または OpenID Connect (OIDC) に対応している任意の ID プロバイダーがあります。このオペレーションは、AWS へのアクセスを必要とするモバイルアプリケーションやクライアントベースのウェブアプリケーションを作成する際に役立ちます。このオペレーションを使用すると、ユーザーには自分の AWS または



IAM アイデンティティは必要ありません。詳細については、「[OIDC フェデレーション](#)」を参照してください。

`AssumeRoleWithWebIdentity` を直接呼び出すのではなく、モバイル開発用の AWS SDK で Amazon Cognito および Amazon Cognito 認証情報プロバイダーを使用することをお勧めします。詳細については、Amplify ドキュメントの「[Amplify による認証](#)」を参照してください。

Amazon Cognito を使用していない場合は、AWS STS の `AssumeRoleWithWebIdentity` アクションを呼び出します。これは無署名の呼び出しです。つまり、アプリが AWS セキュリティ認証情報にアクセスできなくても呼び出すことができます。この呼び出しを行うときに、以下の情報を渡します。

- アプリが引き受ける必要のあるロールの Amazon リソースネーム (ARN)。アプリで複数のサインイン方法をサポートしている場合は、ID プロバイダーごとに 1 つずつ、複数のロールを定義する必要があります。`AssumeRoleWithWebIdentity` を呼び出すときは、ユーザーがサインインしたプロバイダーに固有のロールの ARN を含める必要があります。
- アプリがユーザーを認証した後、IdP から取得したトークン。
- [セッションタグ](#)としてトークンに属性を渡すように IdP を設定できます。
- ( オプション ) 一時的なセキュリティ認証情報の期間を指定する期間。`DurationSeconds` パラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。このパラメータを渡さない場合、一時的な認証情報の有効期限は 1 時間です。この API の `DurationSeconds` パラメータは、コンソールセッションの期間を指定する `SessionDuration` HTTP パラメータとは別のものです。コンソールのサインイントークンの場合は、フェデレーションエンドポイントに対するリクエストで `SessionDuration` HTTP パラメータを使用します。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。
- ロールセッション名 ロールが異なるプリンシパルによって使用される場合にセッションを識別するには、この文字列値を使用します。セキュリティ上の理由から、管理者は [AWS CloudTrail ログ](#)内のこのフィールドを表示し、AWS でアクションを実行したユーザーを確認できます。管理者が、ロールを引き受けるときにセッション名に特定の値を指定することを要求している場合があります。詳細については、「[sts:RoleSessionName](#)」を参照してください。
- ( オプション ) 送信元アイデンティティ。フェデレーションユーザーは、ロールを引き受けるときにソース ID を指定するように要求できます。ソース ID を設定した後は、値を変更できません。これは、ロールセッション中に実行されるすべてのアクションのリクエストに存在します。ソース ID 値は[ロールの連鎖](#)セッション間で存続します。ソース ID 情報は、AWS CloudTrail ログを使用して、ロールでアクションを実行したユーザーを特定します。ソースアイデンティティの使

用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

- (オプション) インラインまたはマネージド ポリシーセッション。これらのポリシーでは、ロールセッションに割り当てられているロールのアイデンティティベースのポリシーのアクセス許可を制限しています。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。

#### Note

AssumeRoleWithWebIdentity に対する呼び出しは署名 (暗号化) されていません。そのため、信頼されている経路を通じてリクエストが送信される場合のみ、オプションのセッションポリシーを含める必要があります。そうでない場合、他のユーザーが制限を削除するようにポリシーを変更できます。

AssumeRoleWithWebIdentity を呼び出すとき、AWS はトークンの信頼性を確認します。たとえば、プロバイダーに応じて、AWS はプロバイダーを呼び出して、アプリが渡したトークンを含める場合があります。ID プロバイダーがトークンを確認した場合、AWS は次の情報を返します。

- 一時的なセキュリティ認証情報一式。一時的なセキュリティ認証情報は、アクセスキー ID、シークレットアクセスキー、およびセッショントークンで構成されています。
- 引き受けたロールのロール ID と ARN。
- 一意のユーザー ID を含む SubjectFromWebIdentityToken 値。

一時的なセキュリティ認証情報がある場合、それを使って AWS API を呼び出すことができます。これは、長期的なセキュリティ認証情報を使用して AWS API 呼び出しを行うのと同じ処理です。異なる点は、AWS で一時的なセキュリティ認証情報が有効であることを確認できる、セッショントークンを含める必要があることです。

アプリで認証情報をキャッシュする必要があります。先述したように、認証情報はデフォルトで 1 時間後に無効になります。[SDK の AmazonSTSCredentialsProviderAWS](#) オペレーションを使用しない場合、再度 AssumeRoleWithWebIdentity を呼び出すかどうかはお客様およびお客様のアプリによります。古い認証情報が失効する前に、このオペレーションを呼び出して新しい一時的なセキュリティ認証情報を取得します。

## [AssumeRoleWithSAML](#) — SAML 2.0 と互換性のあるエンタープライズ ID プロバイダーを介したフェデレーション

AssumeRoleWithSAML API オペレーションでは、組織の既存の ID システムによって認証されたフェデレティッドユーザーの一時的なセキュリティ認証情報のセットが返ります。また、ユーザーは [SAML 2.0](#) (Security Assertion Markup Language) を使用して AWS に認証および許可情報を渡す必要があります。この API オペレーションは、SAML アサーションを生成できるソフトウェアに認証システム (Windows Active Directory、OpenLDAP など) を統合している組織で役立ちます。このような統合はユーザー ID とアクセス許可に関する情報を提供します (Active Directory Federation Services や Shibboleth など)。詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。

### Note

AssumeRoleWithSAML に対する呼び出しは署名 (暗号化) されていません。そのため、信頼されている経路を通じてリクエストが送信される場合のみ、オプションのセッションポリシーを含める必要があります。そうでない場合、他のユーザーが制限を削除するようにポリシーを変更できます。

これは無署名の呼び出しです。つまり、アプリが AWS セキュリティ認証情報にアクセスできなくても呼び出すことができます。この呼び出しを行うときに、以下の情報を渡します。

- アプリが引き受ける必要のあるロールの Amazon リソースネーム (ARN)。
- ID プロバイダーを表す IAM で作成された SAML プロバイダーの ARN。
- アプリからのサインインリクエストに対する認証レスポンスで、SAML ID プロバイダーによって提供される、base64 でエンコードされた SAML アサーション。
- [セッションタグ](#)として属性を SAML アサーションに渡すように IdP を設定できます。
- ( オプション ) 一時的なセキュリティ認証情報の期間を指定する期間。DurationSeconds パラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。このパラメータを渡さない場合、一時的な認証情報の有効期限は 1 時間です。この API の DurationSeconds パラメータは、コンソールセッションの期間を指定する SessionDuration HTTP パラメータとは別のものです。コンソールのサインイントークンの場合は、フェデレーションエンドポイントに対するリクエストで SessionDuration HTTP パラメータを使用します。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

- (オプション) インラインまたはマネージド ポリシーセッション。これらのポリシーでは、ロールセッションに割り当てられているロールのアイデンティティベースのポリシーのアクセス許可を制限しています。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。
- ロールセッション名 ロールが異なるプリンシパルによって使用される場合にセッションを識別するには、この文字列値を使用します。セキュリティ上の理由から、管理者は [AWS CloudTrail ログ](#)内のこのフィールドを表示し、AWS でアクションを実行したユーザーを確認できます。管理者が、ロールを引き受けるときにセッション名に特定の値を指定することを要求している場合があります。詳細については、「[sts:RoleSessionName](#)」を参照してください。
- (オプション) 送信元アイデンティティ。フェデレーティッドユーザーは、ロールを引き受けるときにソース ID を指定するように要求できます。ソース ID を設定した後は、値を変更できません。これは、ロールセッション中に実行されるすべてのアクションのリクエストに存在します。ソース ID 値は[ロールの連鎖](#)セッション間で持続します。ソース ID 情報は、AWS CloudTrail ログを使用して、ロールでアクションを実行したユーザーを特定します。ソースアイデンティティの使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

AssumeRoleWithSAML を呼び出すときに、AWS は SAML アサーションの信頼性を確認します。ID プロバイダーがアサーションを確認した場合、AWS は次の情報を返します。

- 一時的なセキュリティ認証情報一式。一時的なセキュリティ認証情報は、アクセスキー ID、シークレットアクセスキー、およびセッショントークンで構成されています。
- 引き受けたロールのロール ID と ARN。
- SAML アサーションの Audience 要素の Recipient 属性値を含む SubjectConfirmationData 値。
- SAML アサーションの Issuer 要素の値を含む Issuer 値。
- Issuer 値、AWS アカウント ID、SAML プロバイダーのフレンドリ名から構築されたハッシュ値を含む NameQualifier 要素。Subject 要素と結合する場合、フェデレーティッドユーザーを一意に識別できます。
- SAML アサーションの Subject 要素内の NameID 要素の値を含む Subject 要素。
- SubjectType 要素の形式を示す Subject 要素。値は、persistent、transient、または SAML アサーションで使用されている Format および Subject 要素の完全な NameID URI と

することができます。NameID 要素の Format 属性の詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。

一時的なセキュリティ認証情報がある場合、それを使って AWS API を呼び出すことができます。これは、長期的なセキュリティ認証情報を使用して AWS API 呼び出しを行うのと同じ処理です。異なる点は、AWS で一時的なセキュリティ認証情報が有効であることを確認できる、セッショントークンを含める必要があることです。

アプリで認証情報をキャッシュする必要があります。認証情報はデフォルトで 1 時間後に無効になります。[SDK の AmazonSTSCredentialsProviderAWS](#) アクションを使用していない場合、再度 AssumeRoleWithSAML を呼び出すかどうかはお客様およびお客様のアプリによります。古い認証情報が失効する前に、このオペレーションを呼び出して新しい一時的なセキュリティ認証情報を取得します。

## [GetFederationToken](#) — カスタム ID ブローカーを介したフェデレーション

GetFederationToken API オペレーションでは、フェデレーティッドユーザー用の一時的セキュリティ認証情報が返ります。この API は、デフォルトの有効期限が大幅に長い (1 時間ではなく 12 時間) という点で AssumeRole とは異なります。また、DurationSeconds パラメータを使用して、一時的なセキュリティ認証情報が有効である期間を選択することもできます。結果として得られる認証情報は、900 秒 (15 分) から 129,600 秒 (36 時間) までの指定された期間内で有効です。有効期限を長くすると、新しい認証情報を頻繁に取得する必要がなくなるため、AWS への呼び出し回数を減少させることができます。

このリクエストを行うときは、特定の IAM ユーザーの認証情報を使用します。一時的セキュリティ認証情報のアクセス許可は、GetFederationToken を呼び出したときに渡すセッションポリシーによって決まります。結果として得られるセッションのアクセス許可は、IAM ユーザーポリシーとユーザーが渡すセッションポリシーの共通部分です。セッションポリシーを使用して、フェデレーションをリクエストしている IAM ユーザーのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。

GetFederationToken オペレーションによって返される一時的な認証情報を使用すると、セッションのプリンシパルタグには、ユーザーのタグと渡されたセッションタグが含まれます。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

GetFederationToken 呼び出しは、セッショントークン、アクセスキー、シークレットキー、失効情報で構成される一時的セキュリティ認証情報を返します。組織内でアクセス権限



を管理する (たとえば、プロキシアプリケーションを使用してアクセス権限を割り当てる) 場合、GetFederationToken を使用できます。

以下の例に、GetFederationToken を使用したリクエストと応答のサンプルを示します。このリクエスト例では、指定された期間の呼び出し元ユーザーを[セッションポリシー](#) ARN および[セッションタグ](#)とフェデレートします。結果のセッションには Jane-session という名前が付けられます。

#### Example リクエストの例

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetFederationToken  
&Name=Jane-session  
&PolicyArns.member.1.arn==arn%3Aaws%3Aiam%3A%3A123456789012%3Apolicy%2FRole1policy  
&DurationSeconds=1800  
&Tags.member.1.Key=Project  
&Tags.member.1.Value=Pegasus  
&Tags.member.2.Key=Cost-Center  
&Tags.member.2.Value=12345  
&AUTHPARAMS
```

前述の例に示すポリシー ARN には、次の URL エンコードされた ARN が含まれています。

```
arn:aws:iam::123456789012:policy/Role1policy
```

また、例の &AUTHPARAMS パラメータは、認証情報のプレースホルダーとして使用されることに注意してください。これは、署名であり、AWS HTTP API リクエストに含める必要があります。[AWS SDK](#) を使用して API リクエストを作成することをお勧めします。その利点の 1 つは、SDK がリクエストの署名を処理することです。API リクエストを手動で作成し、署名する必要がある場合は、「Amazon Web Services 全般のリファレンス」の「[署名バージョン 4 を使用して AWS リクエストに署名する](#)」を参照してリクエストに署名する方法を確認してください。

一時的セキュリティ認証情報に加えて、レスポンスにはフェデレーテッドユーザーの Amazon リソースネーム (ARN)、および認証情報の有効期限が含まれています。

#### Example レスポンスの例

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
<GetFederationTokenResult>  
<Credentials>
```

```
<SessionToken>
AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTwdQW
LWskWHGBuFqwAeMicRXmxfpSPfIeoIYRqTf1fKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPKyQDYwT7WZ0wq5V5XDvp75YU
9HFv1Rd8Tx6q6fE8YQcHNvXAKiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
+scqKmlzm8FDrypNC9Yjc8fP0Ln9FX9KSYvKTr4rvx3iS1lTJabIQwj2ICCEXAMPLE==
</SessionToken>
<SecretAccessKey>
wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
</SecretAccessKey>
<Expiration>2019-04-15T23:28:33.359Z</Expiration>
<AccessKeyId>AKIAIOSFODNN7EXAMPLE;</AccessKeyId>
</Credentials>
<FederatedUser>
  <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
  <FederatedUserId>123456789012:Jean</FederatedUserId>
</FederatedUser>
<PackedPolicySize>4</PackedPolicySize>
</GetFederationTokenResult>
<ResponseMetadata>
  <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</GetFederationTokenResponse>
```

### Note

AWS 変換では、渡されたセッションポリシーとセッションタグが、個別の制限を持つひとまとめのバイナリ形式に圧縮されます。プレーンテキストが他の要件を満たしていても、この制限ではリクエストが失敗する可能性があります。PackedPolicySize レスポンス要素は、リクエストのポリシーとタグがサイズ制限にどの程度近づいているかをパーセントで示します。

AWS ではリソースレベルでアクセス許可を付与する（たとえば、Amazon S3 バケットにリソースベースのポリシーをアタッチする）ことを推奨しています。Policy パラメータは省略できます。ただし、フェデレーティッドユーザーのポリシーを含まない場合、一時的セキュリティ認証情報ではアクセス権限が付与されません。この場合、リソースポリシーを使用してフェデレーティッドユーザーに AWS リソースへのアクセスを許可する必要があります。

例えば、AWS アカウント 番号が 111122223333 であり、Susan にアクセスを許可しようとしている Amazon S3 バケットがあるとします。Susan の一時的セキュリティ

ティ認証情報にはバケットのポリシーが含まれていません。この場合、Susan の ARN (arn:aws:sts::111122223333:federated-user/Susan など) と一致する ARN があるポリシーがバケットにあるようにする必要があります。

## [GetSessionToken](#) — 信頼されていない環境にあるユーザー向けの一時的認証情報

この GetSessionToken API オペレーションでは、既存の IAM ユーザーに一時的セキュリティ認証情報のセットが返ります。この API は、MFA が IAM ユーザーに対して有効なときに AWS リクエストを作成するなど、セキュリティを強化するために役立ちます。認証情報は一時的なものであるため、安全性の低い環境からリソースにアクセスする IAM ユーザーがいる場合、これによりセキュリティが強化されます。安全性の低い環境の例には、モバイルデバイスやウェブブラウザが含まれます。詳細については、[一時的なセキュリティ認証情報のリクエスト](#) または [AWS Security Token Service API リファレンス](#) の「GetBucketEncryption」を参照してください。

デフォルトでは、IAM ユーザーの一時的なセキュリティ認証情報は、最大 12 時間有効です。ただし、DurationSeconds パラメータを使用して最短 15 分、最長 36 時間の有効期間をリクエストできます。セキュリティ上の理由から、AWS アカウントのルートユーザーのトークンは有効期間が 1 時間に制限されます。

GetSessionToken は、セッショントークン、アクセスキー ID、およびシークレットアクセスキーから構成される一時的なセキュリティ認証情報を返します。以下の例は、GetSessionToken を使用したリクエストと応答のサンプルを示します。レスポンスには、一時的なセキュリティ認証情報の有効期限も含んでいます。

### Example リクエストの例

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetSessionToken  
&DurationSeconds=1800  
&AUTHPARAMS
```

この例の AUTHPARAMS パラメータは署名のプレースホルダーです。署名は AWS HTTP API リクエストに含める必要がある認証情報です。[AWS SDK](#) を使用して API リクエストを作成することをお勧めします。その利点の 1 つは、SDK がリクエストの署名を処理することです。API リクエストを手動で作成し、署名する必要がある場合は、「Amazon Web Services 全般のリファレンス」の「[署名バージョン 4 を使用して AWS リクエストに署名する](#)」を参照してリクエストに署名する方法を確認してください。



## Example レスポンスの例

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetSessionTokenResult>
    <Credentials>
      <SessionToken>
        AQoEXAMPLEH4aoAH0gNCAPyJxz4B1CFFxWNE10PTgk5TthT+FvwqnKwRc0IfRrh3c/L
        To6UddyJw00vEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgrmpRV3z
        rkuWJ0gQs8IZZaIv2BXIa2R40lglkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp
        Z3CYWFXG8C5zqx37wn0E49mRl/+0tkIKG07fAE
      </SessionToken>
      <SecretAccessKey>
        wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY
      </SecretAccessKey>
      <Expiration>2011-07-11T19:55:29.611Z</Expiration>
      <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
    </Credentials>
  </GetSessionTokenResult>
  <ResponseMetadata>
    <RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>
  </ResponseMetadata>
</GetSessionTokenResponse>
```

オプションで、GetSessionToken リクエストに、AWS の多要素認証 (MFA) で確認する SerialNumber および TokenCode 値を含めることができます。指定された値が有効であれば、AWS STS は、MFA 認証の状態を含む一時的なセキュリティ認証情報を提供します。この一時的なセキュリティ認証情報を使用して、MFA で保護された API オペレーションまたは AWS ウェブサイトに、MFA 認証が有効である限りアクセスできます。

以下の例は、MFA 認証コードとデバイスのシリアルナンバーを含む GetSessionToken リクエストを示しています。

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=7200
&SerialNumber=YourMFADeviceSerialNumber
&TokenCode=123456
&AUTHPARAMS
```

**Note**

AWS STS の呼び出しは、グローバルエンドポイントまたは AWS アカウント をアクティブ化しているリージョンのエンドポイントに対して行えます。詳細については、[リージョンとエンドポイントの「AWS STS」セクション](#)を参照してください。

この例の AUTHPARAMS パラメータは 署名のプレースホルダーです。署名は AWS HTTP API リクエストに含める必要がある認証情報です。[AWS SDK](#) を使用して API リクエストを作成することをお勧めします。その利点の 1 つは、SDK がリクエストの署名を処理することです。API リクエストを手動で作成し、署名する必要がある場合は、[Amazon Web Services 全般のリファレンス] の「[署名バージョン 4 を使用して AWS リクエストに署名する](#)」を参照してリクエストに署名する方法を確認してください。

## AWS STS API オペレーションの比較

次の表は、一時的なセキュリティ認証情報を返す、AWS STS の API オペレーションの機能を比較したものです。ロールを引き受けることで一時的なセキュリティ認証情報をリクエストするための別の方法については、「[IAM ロールを使用する](#)」を参照してください。セッションタグを渡すことができるさまざまな AWS STS API オペレーションについては、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

### API オプションの比較

AWS STS API	呼び出し元	認証情報の有効期間 (最小   最大   デフォルト)	MFA サポート <sup>1</sup>	セッションポリシーのサポート <sup>2</sup>	得られた一時的な認証情報に対する制限
<a href="#">AssumeRole</a>	既存の一時的なセキュリティ認証情報を持つ IAM ユーザーまたは IAM ロール	15 分   最大セッション期間設定 <sup>3</sup>   1 時間	はい	はい	GetFederationToken または GetSessionToken を呼び出せません。

AWS STS API	呼び出し元	認証情報の有効期間 (最小   最大   デフォルト)	MFA サポート <sup>1</sup>	セッションポリシーのサポート <sup>2</sup>	得られた一時的な認証情報に対する制限
<a href="#">AssumeRoleWithSAML</a>	任意のユーザー。呼び出し元は、既知の ID プロバイダーからの認証を示す SAML 認証レスポンスを渡す必要があります。	15 分   最大セッション期間設定 <sup>3</sup>   1 時間	いいえ	はい	GetFederationToken または GetSessionToken を呼び出せません。
<a href="#">AssumeRoleWithWebIdentity</a>	任意のユーザー。呼び出し元は、既知の ID プロバイダーからの認証を示す OIDC 準拠の JWT トークンを渡す必要があります。	15 分   最大セッション期間設定 <sup>3</sup>   1 時間	いいえ	はい	GetFederationToken または GetSessionToken を呼び出せません。

AWS STS API	呼び出し元	認証情報の有効期間 (最小   最大   デフォルト)	MFA サポート <sup>1</sup>	セッションポリシーのサポート <sup>2</sup>	得られた一時的な認証情報に対する制限
<a href="#">GetFederationToken</a>	IAM ユーザーまたは AWS アカウントのルートユーザー	IAM ユーザー: 15 分   36 時間   12 時間  ルートユーザー: 15 分   1 時間   1 時間	いいえ	はい	AWS CLI または AWS API を使用して IAM オペレーションを呼び出すことはできません。この制限はコンソールセッションには適用されません。  GetCallerIdentity 以外の AWS STS オペレーションは呼び出せません。 <sup>4</sup>  コンソールへの SSO は許可されています。 <sup>5</sup>
<a href="#">GetSessionToken</a>	IAM ユーザーまたは AWS アカウントのルートユーザー	IAM ユーザー: 15 分   36 時間   12 時間  ルートユーザー: 15 分   1 時間   1 時間	はい	いいえ	リクエストに MFA 情報が含まれていない場合は、IAM API オペレーションを呼び出せません。  AssumeRole または GetCallerIdentity を除く AWS STS API オペレーションを呼び出せません。  コンソールへの SSO は許可されていません。 <sup>6</sup>

<sup>1</sup> MFA サポート。AssumeRole および GetSessionToken API オペレーションを呼び出すときに、多要素認証 (MFA) デバイスに関する情報を含めることができます。そうすることで、API 呼び出しによって得られた一時的なセキュリティ認証情報を、MFA デバイスで認証されたユーザーだけが使用できるようにできます。詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

<sup>2</sup> セッションポリシーのサポート。セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時セッションをプログラムで作成する際にパラメータとして渡すポリシーです。このポリシーでは、セッションに割り当てられているロールまたはユーザーのアイデンティティベースのポリシーのアクセス許可を制限しています。結果として得られるセッションのアクセス許可は、エンティティの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。

<sup>3</sup> [Maximum session duration setting] (最大セッション期間設定)。DurationSeconds パラメータを使用して、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定できます。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。

<sup>4</sup> GetCallerIdentity。このオペレーションを実行するためのアクセス許可は必要ありません。管理者が、sts:GetCallerIdentity アクションへのアクセスを明示的に拒否するポリシーを IAM ユーザーまたはロールに追加しても、このオペレーションは実行できます。IAM ユーザーまたはロールがアクセスを拒否されたときに同じ情報が返されるため、アクセス許可は必要ありません。レスポンスの例については、「[iam:DeleteVirtualMFADevice を実行する権限がありません](#)」を参照してください。

<sup>5</sup> コンソールへのシングルサインオン (SSO)。SSO をサポートするために、AWS でフェデレーションエンドポイント (<https://signin.aws.amazon.com/federation>) を呼び出し、一時的なセキュリティ認証情報を渡すことができます。エンドポイントから返されるトークンを使用すると、パスワードを要求せずにユーザーを直接コンソールにサインインさせる URL を作成できます。詳細とサンプルスクリプトについては、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Console にアクセス可能にする](#)」および AWS セキュリティブログの「[AWS へのクロスアカウントアクセスを有効にする方法](#)」を参照してください。

<sup>5</sup> 一時的な認証情報を取得した後、この認証情報をフェデレーションのシングルサインオンエンドポイントに渡して AWS Management Console にアクセスすることはできません。詳細については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

## AWS リソースを使用した一時的な認証情報の使用

一時的な認証情報を使用して、AWS または AWS CLI API ( [AWS SDK](#) を使用 ) を使用して AWS リソースのプログラム要求を行うことができます。一時的な認証情報は、IAM ユーザーの認証情報など、長期的なセキュリティ認証情報を使用するのと同じアクセス許可を提供します。ただし、いくつかの違いがあります。

- 一時的セキュリティ認証情報を使用して呼び出しを行うときは、一時的な認証情報と共に返されるセッショントークンを呼び出しに含める必要があります。AWS はセッショントークンを使用して、一時的セキュリティ認証情報を検証します。
- 一時的な認証情報は、指定した期間が過ぎると失効します。一時認証情報が有効期限切れになると、その認証情報を使用する呼び出しはすべて失敗するため、新しい一時認証情報を生成する必要があります。一時的な認証情報は、最初に指定された間隔を超えて延長または更新することはできません。
- 一時的な認証情報を使用してリクエストを行う場合、プリンシパルに一連のタグが含まれる場合があります。これらのタグは、引き受けるロールにアタッチされたセッションタグとタグから取得されます。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

[AWS SDK](#)、[AWS Command Line Interface](#) (AWS CLI) または [Tools for Windows PowerShell](#) を使用している場合、一時的なセキュリティ認証情報を取得および使用方法はコンテキストによって異なります。コード、AWS CLI、または Tools for Windows PowerShell コマンドを EC2 インスタンス内で実行している場合は、Amazon EC2 のロールを利用できます。それ以外の場合は、[AWS STS API](#) を呼び出して一時的な認証情報を取得した後、その情報を明示的に使用して AWS のサービス呼び出すことができます。

### Note

AWS Security Token Service (AWS STS) を使用して、AWS リソースへのアクセスをコントロールできる一時的セキュリティ認証情報を持つ、信頼されたユーザーを作成および提供することができます。AWS STS の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。AWS STS は、デフォルトのエンドポイントが <https://sts.amazonaws.com> にあるグローバルサービスです。このエンドポイントや他のエンドポイントから取得した認証情報はグローバルに有効ですが、このエンドポイントは米国東部 (バージニア北部) リージョンにあります。これらの認証情報は、どのリージョンのサービスとリソースでも機能します。サポートされているリージョンのいずれかにあるエンドポイントへの AWS STS API 呼び出しを実行することもできます。地理的に近い場所にあるリージョンのサーバーに対し

てリクエストを実行することによって、レイテンシーを低減できます。認証情報を取得したリージョンに関係なく、認証情報はグローバルに使用できます。詳細については、「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

## 目次

- [Amazon EC2 インスタンスでの一時的な認証情報の使用](#)
- [AWS SDK での一時的セキュリティ認証情報の使用](#)
- [AWS CLI で一時的なセキュリティ認証情報を使用する](#)
- [API オペレーションで一時的なセキュリティ認証情報を使用する](#)
- [詳細情報](#)

## Amazon EC2 インスタンスでの一時的な認証情報の使用

AWS CLI コマンドまたはコードを EC2 インスタンス内で実行する場合、認証情報を取得するために推奨される方法は、[Amazon EC2](#) のロールを使用する方法です。つまり、EC2 インスタンスで実行されているアプリケーションに付与するアクセス許可を指定する IAM ロールを作成します。インスタンスを起動するときに、このロールとインスタンスを関連付けます。

これにより、インスタンスで実行されるアプリケーション、AWS CLI、Tools for Windows PowerShell コマンドがインスタンスメタデータから自動的に一時的セキュリティ認証情報を取得できるようになります。一時的なセキュリティ認証情報を明示的に取得する必要はありません。AWS SDK、AWS CLI、および Tools for Windows PowerShell は EC2 インスタンスメタデータサービス (IMDS) から認証情報を自動的に取得して使用します。一時的なセキュリティ認証情報には、インスタンスに関連付けられたロールに定義されているアクセス権限が付与されます。

詳細情報および例については、次のセクションを参照してください。

- [IAM ロールを使用した Amazon Elastic Compute Cloud の AWS リソースへのアクセスの許可](#) — AWS SDK for Java
- [IAM ロールを使用したアクセスの許可](#) — AWS SDK for .NET
- [ロールの作成](#) — AWS SDK for Ruby



## AWS SDK での一時的セキュリティ認証情報の使用

コード内で一時的なセキュリティ認証情報を使用するには、AssumeRole などの AWS STS API をプログラムから呼び出し、結果として得られる認証情報とセッショントークンを抽出します。その後、これらの値を AWS への後続の呼び出しの認証情報として使用します。以下の例は、AWS SDK を使用している場合に、一時的なセキュリティ認証情報を使用する方法を示す疑似コードです。

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
    assumeRoleResult.AccessKeyId,
    assumeRoleResult.SecretAccessKey,
    assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

Python で記述された例 ([AWS SDK for Python \(Boto\)](#) を使用) については、「[IAM ロール \(AWS API\) の切り替え](#)」を参照してください。この例では、AssumeRole を呼び出して一時的なセキュリティ認証情報を取得し、それらの認証情報を使用して Amazon S3 を呼び出します。

AssumeRole、GetFederationToken、およびその他の API オペレーションを呼び出す方法の詳細については、「[AWS Security Token Service API リファレンス](#)」を参照してください。結果から一時的なセキュリティ認証情報とセッショントークンを取得する方法の詳細については、お使いの SDK のドキュメントを参照してください。すべての AWS SDK に関するドキュメントは、主要な [AWS のドキュメントページ](#) の「SDK とツールキット」セクションにあります。

古い認証情報が失効する前に、新しい認証情報を取得する必要があります。一部の SDK では、ユーザーの代わりに認証情報の更新処理を管理するプロバイダーを使用できます。お使いの SDK のドキュメントを確認してください。

## AWS CLI で一時的なセキュリティ認証情報を使用する

AWS CLI で一時的なセキュリティ認証情報を使用できます。これは、ポリシーをテストする場合に便利です。

[AWS CLI](#) を使用して、[AWS STS API](#) (AssumeRole や GetFederationToken など) を呼び出し、結果の出力をキャプチャします。次の例は、出力をファイルに送信する AssumeRole の呼び出しを示しています。この例では、profile パラメータは AWS CLI 設定ファイル内のプロファイルであると想定されています。また、ロールを引き受けるアクセス許可を持つ IAM ユーザーの認証情報を参照することも想定されます。



```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-name "RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

コマンドが完了したら、ルーティングした場所からアクセスキー ID、シークレットアクセスキー、セッショントークンを抽出できます。この操作は、手動またはスクリプトを使用して行うことができます。次に、これらの値を環境変数に割り当てます。

AWS CLI コマンドを実行すると、AWS CLI によって特定の順序で資格情報が検索されます。最初は環境変数で、次に構成ファイルで検索されます。したがって、一時的な認証情報を環境変数に設定すると、AWS CLI でこれらの認証情報がデフォルトで使用されます（コマンドで `profile` パラメータを指定すると、AWS CLI は環境変数をスキップします。代わりに、AWS CLI は構成ファイルを調べます。これにより、必要に応じて環境変数の資格情報をオーバーライドできます。）

以下の例では、環境変数に一時的なセキュリティ認証情報を設定した後で AWS CLI コマンドを呼び出しています。AWS CLI コマンドには `profile` パラメータが指定されていないため、AWS CLI で最初に環境変数内で認証情報が検索されます。その結果、一時的な認証情報が使用されます。

## Linux

```
$ export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
$ aws ec2 describe-instances --region us-west-1
```

## Windows

```
C:\> SET AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>
C:\> aws ec2 describe-instances --region us-west-1
```

## API オペレーションで一時的なセキュリティ認証情報を使用する

AWS に対する直接 HTTPS API リクエストを行う場合、AWS Security Token Service (AWS STS) から取得した一時的セキュリティ認証情報でそれらのリクエストを署名できます。これを行うには AWS STS から受け取るアクセスキー ID とシークレットアクセスキーを使用します。長期的な認証情報を使用してリクエストに署名するのと同じ方法で、アクセスキー ID とシークレットアクセスキーを使用します。また、AWS STS から渡されるセッショントークンを API リクエストに追加します。セッショントークンは、HTTP ヘッダーに追加するか、`X-Amz-Security-Token` という名前のクエリ文字列パラメータに追加します。セッショントークンを追加するのは、HTTP ヘッダーまた

はクエリ文字列パラメータのいずれかです。両方には追加しません。HTTPS API リクエストの署名の詳細については、「AWS 全般のリファレンス」の「[AWS API リクエストの署名](#)」を参照してください。

## 詳細情報

他の AWS のサービスで AWS STS を使用方法の詳細については、以下のブログ記事を参照してください。

- Amazon S3。「Amazon Simple Storage Service ユーザーガイド」の「[IAM ユーザー一時クレデンシャルを使用したリクエストの作成](#)」または「[フェデレーションユーザー一時クレデンシャルを使用したリクエストの作成](#)」を参照してください。
- Amazon SNS。「Amazon Simple Notification Service デベロッパーガイド」の「[Amazon SNS でのアイデンティティベースのポリシーの使用](#)」を参照してください。
- Amazon SQS。「Amazon Simple Queue Service デベロッパーガイド」の「[Amazon SQS での Identity and access management](#)」を参照してください。
- Amazon SimpleDB『[Amazon SimpleDB 開発者ガイド](#)』の「[一時的なセキュリティクレデンシャルの使用](#)」を参照してください。

## 一時的なセキュリティ認証情報のアクセス権を制御する

AWS Security Token Service (AWS STS) を使用して、AWS リソースへのアクセスをコントロールできる一時的なセキュリティ認証情報を持つ、信頼されたユーザーを作成および提供することができます。AWS STS の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。AWS STS によって一時的なセキュリティ証明書が発行された後、それらの証明書は期限切れになるまで有効であり、取り消すことはできません。ただし、一時的なセキュリティ認証情報に割り当てられたアクセス権限は、認証情報を使用するリクエストがなされるたびに評価されるため、発行後にアクセス権を変更することで、認証情報を取り消すのと同等の効果を得ることができます。

以下のトピックでは、AWS のアクセス許可とポリシーに関する実用的な知識があることを前提としています。これらのトピックの詳細については、「[AWS リソースのアクセス管理](#)」を参照してください。

### トピック

- [AssumeRole、AssumeRoleWithSAML、AssumeRoleWithWebIdentity のアクセス権限](#)
- [引き受けたロールで実行されるアクションのモニタリングと制御](#)
- [GetFederationToken のアクセス権限](#)

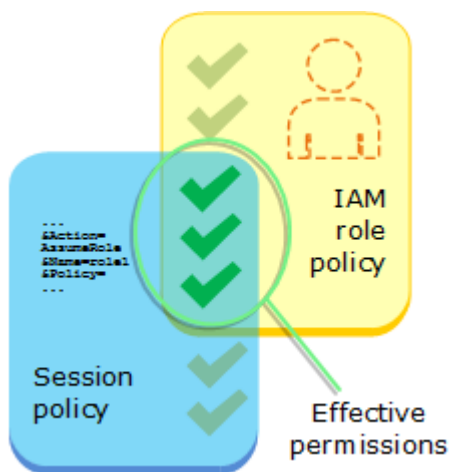
- [GetSessionToken のアクセス権限](#)
- [一時的なセキュリティ認証情報のアクセス権限を無効にする](#)
- [一時的なセキュリティ認証情報を作成するためのアクセス権限の付与](#)
- [ID 対応コンソールセッションを使用するアクセス許可の付与](#)

## AssumeRole、AssumeRoleWithSAML、AssumeRoleWithWebIdentity のアクセス権限

引き受けるロールのアクセス許可ポリシーによって、AssumeRole、AssumeRoleWithSAML、および AssumeRoleWithWebIdentity によって返る一時的なセキュリティ認証情報のアクセス許可が決まります。これらのアクセス権限は、ロールを作成または更新するときに定義します。

必要に応じて、インラインまたはマネージド [セッションポリシー](#)

を AssumeRole、AssumeRoleWithSAML、または AssumeRoleWithWebIdentity API オペレーションのパラメータとして渡すことができます。セッションポリシーは、ロールの一時的な認証情報セッションのアクセス許可を制限します。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。ロールの一時的な認証情報を以降の AWS API コールで使用し、ロールを所有するアカウント内のリソースにアクセスできます。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。AWS でロールの有効なアクセス許可を決定する方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。



AssumeRole を最初に呼び出した認証情報にアタッチされたポリシーは、「許可」または「拒否」承認の決定を行うときに AWS によって評価されません。ユーザーは引き受けたロールによって割り当てられたアクセス権限に従って、元のアクセス権限を一時的に放棄します。AssumeRoleWithSAML および AssumeRoleWithWebIdentity API オペレーションの場合、API の呼び出し元が AWS ID ではないため、評価するポリシーはありません。

## 例: AssumeRole を使用してアクセス権限を割り当てる

AssumeRole API オペレーションをさまざまなポリシーと共に使用できます。ここにいくつか例を挙げます。

### ロールのアクセス許可ポリシー

この例では、オプションの Policy パラメータでセッションポリシーを指定せずに、AssumeRole API オペレーションを呼び出します。一時的な認証情報に割り当てられるアクセス許可は、引き受けるロールのアクセス許可ポリシーによって決まります。次のアクセス許可ポリシーの例では、productionapp という名前の S3 バケットに含まれるすべてのオブジェクトを一覧表示するアクセス許可をロールに付与します。また、このバケット内のオブジェクトを取得、格納、削除することもロールに許可します。

### Example ロールアクセス権限ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

### パラメータとして渡されたセッションポリシー

前の例と同じロールを引き受けることをユーザーに許可すると仮定します。ただし、この場合、オブジェクトを取得して productionapp S3 バケットに配置するためのアクセス許可のみをロールセッションに付与する必要があります。オブジェクトの削除は許可しません。これを実現する 1 つの方法が、新しいロールを作成し、そのロールのアクセス許可ポリシーで目的のアクセス許可を指定する

ことです。もう 1 つの方法では、AssumeRole API を呼び出し、API オペレーションの一部としてオプションの Policy パラメータにセッションポリシーを含めます。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、委任されているロールのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。ロールセッションのアクセス許可の詳細については、「[セッションポリシー](#)」を参照してください。

新しいセッションの一時的な認証情報を取得した後、それらのアクセス許可を付与するユーザーに取得した認証情報を渡すことができます。

たとえば、API 呼び出しのパラメータとして以下のポリシーを渡すとします。このセッションを使用するユーザーには、次のアクションのみを実行するアクセス許可が付与されています。

- productionapp バケットのすべてのオブジェクトをリストします。
- productionapp バケットのオブジェクトを取得し格納します。

次のセッションポリシーでは、s3:DeleteObject アクセス許可は除外され、引き受けたセッションに s3:DeleteObject アクセス許可は付与されません。このポリシーでは、ロールの既存のアクセス許可ポリシーが上書きされるように、ロールセッションのアクセス許可の上限を設定します。

#### Example AssumeRole API コールで渡すセッションポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

## リソースベースのポリシー

一部の AWS リソースはリソースベースのポリシーをサポートし、このポリシーは一時的なセキュリティ認証情報に影響するアクセス許可を定義する別の仕組みとして利用できます。リソースベースのポリシーをサポートしているのは、Amazon S3 バケット、Amazon SNS トピック、Amazon SQS キューなど、一部のリソースのみです。以下の例では、前の例を拡張して、productionapp という名前の S3 バケットを使用します。以下に挙げるポリシーが、バケットにアタッチされます。

productionapp バケットに以下のリソースベースのポリシーをアタッチすると、すべてのユーザーに対して、バケットからオブジェクトを削除するアクセス許可は拒否されます。(ポリシーの Principal 要素を参照してください。) ロールのアクセス許可ポリシーで DeleteObject アクセス許可が付与されていますが、これには引き受けたすべてのロールユーザーも含まれます。明示的な Deny ステートメントは Allow ステートメントより常に優先されます。

### Example バケットポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "*"},
    "Effect": "Deny",
    "Action": "s3:DeleteObject",
    "Resource": "arn:aws:s3:::productionapp/*"
  }
}
```

AWS による複数のポリシータイプの組み合わせと評価の詳細については、「[ポリシーの評価論理](#)」を参照してください。

## 引き受けたロールで実行されるアクションのモニタリングと制御

[IAM ロール](#)は、[アクセス許可](#)が割り当てられている IAM 内のオブジェクトです。IAM ID または AWS の外部からの ID を使用して [そのロールを引き受ける](#) と、ロールに割り当てられたアクセス許可を持つセッションを受け取ります。

AWS でアクションを実行すると、セッションに関する情報を AWS CloudTrail にログ記録して、アカウント管理者がモニタリングできるようになります。管理者は、AWS でアクションを実行する個人またはアプリケーションを識別するカスタム文字列を渡すように ID を要求するようにロールを設定できます。この ID 情報は、ソース ID として AWS CloudTrail に保存されます。管理者は CloudTrail のアクティビティを確認すると、ソース ID 情報を表示して、引き受けたロールセッションでアクションを実行したユーザーやアクションを判断できます。



ソースIDが設定されると、ロールセッション中に実行されるすべての AWS アクションの要求に存在します。設定された値は、ロールが AWS CLI または AWS API を介して別のロールを引き受けるために使用される場合、[ロール連鎖](#)と呼ばれます。設定される値は、ロールセッション中に変更できません。管理者は、ソース ID の存在または値に基づいて詳細なアクセス許可を構成して、共有ロールで実行される AWS アクションをさらに制御できます。ソース ID 属性を使用できるかどうか、必須かどうか、どの値を使用できるかを決定できます。

ソース ID の使用方法は、ロールセッション名およびセッションタグとは重要な点で異なります。ソース ID 値は、設定後は変更できません。また、ロールセッションで実行される追加のアクションでも保持されます。セッションタグとロールセッション名の使用方法は次のとおりです。

- セッションタグ – ロールを引き受けるとき、またはユーザーをフェデレートするときにセッションタグを渡すこともできます。セッションタグは、ロールを引き受けるときに存在します。タグ条件キーを使用して、タグに基づいてプリンシパルにアクセス許可を付与するポリシーを定義できます。次に、CloudTrail を使用して、ロールを引き受けるか、ユーザーをフェデレートするために行われたリクエストを表示できます。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。
- ロールセッション名 – ロール信頼ポリシーで `sts:RoleSessionName` 条件キーを使用して、ユーザーがロールを引き受けるときに特定のセッション名を提供するように要求できます。ロールセッション名は、ロールが異なるプリンシパルによって使用される場合に、ロールセッションを区別するために使用できます。ロールセッション名の詳細については、「[sts:RoleSessionName](#)」を参照してください。

ロールを引き受ける ID を制御する場合は、ソース ID を使用することをお勧めします。ソースアイデンティティは、CloudTrail ログをマイニングして、アクションを実行するためにロールを使用したユーザーを特定する場合にも役立ちます。

## トピック

- [ソース ID を使用するための設定](#)
- [ソースアイデンティティについて知っておくべきこと](#)
- [ソース ID を設定するために必要なアクセス許可](#)
- [ロールを引き受けるときのソース ID の指定](#)
- [AssumeRole でのソース ID の使用](#)
- [AssumeRoleWithSAML を使用したソース ID の使用](#)
- [AssumeRoleWithWebIdentity によるソース ID の使用](#)

- [ソース ID 情報を使用してアクセスを制御する](#)
- [CloudTrail でのソースアイデンティティの表示](#)

## ソース ID を使用するための設定

ソース ID を使用するように設定する方法は、ロールを引き受けるときに使用される方法によって異なります。例えば、IAMユーザーは、AssumeRole オペレーションを直接使用してロールを引き受ける場合があります。エンタープライズ ID (ワークフォースIDとも呼ばれる) がある場合、AWS を使用して AssumeRoleWithSAML リソースにアクセスする可能性があります。エンドユーザーがモバイルまたはWebアプリケーションにアクセスする場合、AssumeRoleWithWebIdentity を使用してアクセスする可能性があります。次に、既存の環境でソース ID 情報を利用するためのを設定する方法を理解するのに役立つ概要のワークフローの概要を示します。

1. テストユーザーおよびロールの構成 — 運用前環境を使用して、テストユーザーおよびロールを構成し、ソースアイデンティティを設定できるようにポリシーを構成します。

フェデレーション ID に ID プロバイダー (IdP) を使用する場合は、アサーションまたはトークンでソース ID に選択したユーザー属性を渡すように IdP を設定します。

2. ロールを引き受けるには – ロールを想定し、テスト用に設定したユーザーとロールにソース ID を渡します。
3. CloudTrail の確認 — CloudTrail ログでテストロールのソース ID 情報を確認します。
4. ユーザーのトレーニング — 運用前の環境でテストした後、必要に応じて、ソース ID 情報を渡す方法をユーザーが把握していることを確認します。本番環境でソース ID の提供をユーザーに要求する期限を設定します。
5. 本番ポリシーの設定 : 本番環境のポリシーを構成し、本番ユーザーおよびロールに追加します。
6. アクティビティのモニタリング — CloudTrail ログを使用して、本番ロールのアクティビティをモニタリングします。

## ソースアイデンティティについて知っておくべきこと

ソース ID を使用する際には、次の点に注意してください。

- ID プロバイダー (IdP) に接続されているすべてのロールの信頼ポリシーに sts:SetSourceIdentity アクセス許可が必要です。ロール信頼ポリシーでこのアクセス許可を持たないロールの場合、AssumeRole\* オペレーションは失敗します。各ロールのロール信頼ポリシーを更新しない場合は、ソース ID を渡すために個別の IdP インスタンスを使用できます。その



後、個別の IdP に接続されているロールにのみ `sts:SetSourceIdentity` アクセス許可を追加します。

- アイデンティティがソースアイデンティティを設定する場合、`sts:SourceIdentity` キーはリクエストに存在します。ロールセッション中に実行される後続のアクションでは、`aws:SourceIdentity` キーがリクエストに存在します。AWS は、`sts:SourceIdentity` または `aws:SourceIdentity` キーのソース ID の値を制御しません。ソース ID を要求する場合は、ユーザーまたは IdP に提供する属性を選択する必要があります。セキュリティ上の理由から、これらの値の提供方法を制御できることを確認する必要があります。
- ソース ID の値は、2 ～ 64 文字の長さである必要があります。英数字、アンダースコア、および、`., += @ -` (ハイフン) のみを含めることができます。テキスト `aws:` から開始するタグキーや値を作成することはできません。このプレフィックスは AWS の内部使用のために予約されています。
- AWS サービスまたはサービスにリンクされたロールがフェデレーション ID またはワーカー ID に代わってアクションを実行する場合、ソース ID 情報は CloudTrail によってキャプチャされません。

#### Important

AWS Management Console でロールを切り替えるには、ロールを引き受けるときにソース ID を設定する必要があります。このようなロールを引き受けるには、AWS CLI または AWS API を呼び出して `AssumeRole` オペレーションを実行し、ソース ID パラメーターを指定します。

ソース ID を設定するために必要なアクセス許可

API オペレーションに一致するアクションに加えて、ポリシーには次のアクセス許可のみのアクションが必要です。

```
sts:SetSourceIdentity
```

- ソース ID を指定するには、プリンシパル (IAM ユーザーとロール) に `sts:SetSourceIdentity` へのアクセス許可が必要です。管理者は、ロールの信頼ポリシーとプリンシパルのアクセス許可ポリシーでこれを設定できます。
- 別のロールでロールを引き受ける場合、[ロールの連鎖](#)と呼ばれるアクセス許可 `sts:SetSourceIdentity` は、ロールを引き受けるプリンシパルのアクセス許可ポリシーと、

ターゲットロールのロール信頼ポリシーの両方で必要です。そうしない場合、ロールの引き受け操作は失敗します。

- ソース ID を使用している場合、ID プロバイダー (IdP) に接続されているすべてのロールの信頼ポリシーに `sts:SetSourceIdentity` アクセス許可が必要です。このアクセス許可なしで IdP に接続されているロールでは、`AssumeRole*` オペレーションは失敗します。各ロールのロール信頼ポリシーを更新しない場合は、ソース ID を渡すために個別の IdP インスタンスを使用し、`sts:SetSourceIdentity` アクセス許可は、個別の IdP に接続されているロールにのみ付与されます。
- アカウントの境界を越えてソースIDを設定するには、2箇所にも `sts:SetSourceIdentity` アクセス許可を含める必要があります。これは、元のアカウントのプリンシパルのアクセス許可ポリシーと、ターゲットアカウントのロールのロール信頼ポリシーにある必要があります。例えば、ロールが[ロールの連鎖](#)を使用して別のアカウントでロールを引き受けるために使用される場合、これを行う必要がある場合があります。

アカウント管理者として、アカウントの IAM ユーザー `DevUser` が同じアカウントの `Developer_Role` を引き継ぐことを許可したいとします。ただし、ユーザーがソース ID を自分の IAM ユーザー名に設定している場合にのみ、このアクションを許可します。その場合、IAM ユーザーに以下のポリシーをアタッチできます。

Example `DevUser` にアタッチされた ID ベースのポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role"
    },
    {
      "Sid": "SetAwsUserNameAsSourceIdentity",
      "Effect": "Allow",
      "Action": "sts:SetSourceIdentity",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role",
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": "${aws:username}"
        }
      }
    }
  ]
}
```

```
    }
  }
]
}
```

許容可能なソース ID 値を適用するには、次のロール信頼ポリシーを設定します。このポリシーは、IAM ユーザー DevUser にロールを引き受けてソースIDを設定するためのアクセス許可を与えます。sts:SourceIdentity条件キーは、許容可能なソース ID 値を定義します。

#### Example ソース ID のロール信頼ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevUserAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/DevUser"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "sts:SourceIdentity": "DevUser"
        }
      }
    }
  ]
}
```

ユーザーは、IAMユーザー DevUser の認証情報を使用して、次の DeveloperRole リクエストを使用して、AWS CLI を受け入れようとしています。

#### Example AssumeRole CLI リクエストの例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer_Role \
--role-session-name Dev-project \
--source-identity DevUser \
```

AWS がリクエストを評価するとき、リクエストコンテキストには `sts:SourceIdentity` の `DevUser` が含まれます。

## ロールを引き受けるときのソース ID の指定

AWS STS `AssumeRole*` API操作の1つを使用してロールの一時的なセキュリティクレデンシャルを取得するときに、ソースIDを指定できます。使用する API オペレーションは、ユースケースによって異なります。例えば、IAMロールを使用して、IAMユーザーに通常はアクセスできない AWS リソースへのアクセスを許可する場合は、`AssumeRole` オペレーションを使用できます。エンタープライズ ID フェデレーションを使用してワークフォースユーザーを管理する場合は、この `AssumeRoleWithSAML` オペレーションを使用できます。OIDC フェデレーションを使用して、エンドユーザーがモバイルまたはウェブアプリケーションにアクセスできるようにする場合は、`AssumeRoleWithWebIdentity` オペレーションを使用できます。このセクションでは、オペレーションごとにソース ID を使用する方法について説明します。一時的な認証情報の一般的なシナリオの詳細については、「[一時的な認証情報の一般的なシナリオ](#)」を参照してください。

## AssumeRole でのソース ID の使用

`AssumeRole` オペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。IAM ユーザーまたはロールの認証情報を使用して `AssumeRole` を呼び出すことができます。ロールを引き受けるときにセッションタグを渡すには、`--source-identity` AWS CLI オプションまたは `SourceIdentity` AWS API パラメータを使用します。次の例は、AWS CLI を使用してソースIDを指定する方法を示しています。

## Example AssumeRole CLI リクエストの例

```
aws sts assume-role \  
--role-arn arn:aws:iam::123456789012:role/developer \  
--role-session-name Audit \  
--source-identity Admin \  

```

## AssumeRoleWithSAML を使用したソース ID の使用

`AssumeRoleWithSAML` オペレーションを呼び出すプリンシパルは、SAMLベースのフェデレーションを使用して認証されます。このオペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。SAML ベースのフェデレーションを使用して AWS Management Console にアクセスする方法の詳細については、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Console にアクセス可能にする](#)」を参照してください。AWS CLI または AWS API アクセスの詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。Active Directory ユー

ザーの SAML フェデレーションを設定するチュートリアルについては、AWS セキュリティブログの「[Active Directory フェデレーションサービスを使用した AWS フェデレーション認証 \(ADFS\)](#)」を参照してください。

管理者は、企業ディレクトリのメンバーに AWS STS AssumeRoleWithSAML オペレーションを使用しての AWS フェデレーションを許可できます。これを行うには、以下のタスクを完了する必要があります。

1. [組織での SAML プロバイダーの構成。](#)
2. [IAM で SAML プロバイダーを作成するには](#)
3. [フェデレーションユーザーのロールとそのアクセス許可を AWS で設定します。](#)
4. [SAML IdP の設定を終了し、SAML 認証レスポンスのアサーションを作成する](#)

ソース ID の SAML 属性を設定するには、Attribute 属性が Name に設定されている `https://aws.amazon.com/SAML/Attributes/SourceIdentity` 要素を含めます。AttributeValue 要素を使用して、ソース ID の値を指定します。例えば、次の ID 属性をソース ID として渡すとします。

```
SourceIdentity:DiegoRamirez
```

これらの属性を渡すには、SAML アサーションに以下の要素を含めます。

Example SAML アサーションのスニペットの例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
<AttributeValue>DiegoRamirez</AttributeValue>
</Attribute>
```

### AssumeRoleWithWebIdentity によるソース ID の使用

AssumeRoleWithWebIdentity オペレーションを呼び出すプリンシパルは、OpenID Connect (OIDC) 準拠のフェデレーションを使用して認証されます。このオペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。AWS Management Console アクセスに OIDC フェデレーションを使用する方法の詳細については、「[OIDC フェデレーション](#)」を参照してください。

OpenID Connect (OIDC) からソース ID を渡すには、JSON ウェブトークン (JWT) にソース ID を含める必要があります。AssumeRoleWithWebIdentity リクエストを送信するときに、トークンの <https://aws.amazon.com/> source\_identity 名前空間にセッションタグを含めます。OIDC

トークンとクレームの詳細については、「Amazon Cognito 開発者ガイド」の「[ユーザープールでのトークンの使用](#)」を参照してください。

例えば、次のデコードされた JWT は、AssumeRoleWithWebIdentity ソースIDで Admin を呼び出すために使用されるトークンです。

Example デコードされた JSON ウェブトークンの例

```
{
  "sub": "johndoe",
  "aud": "ac_oic_client",
  "jti": "ZYUCeRMQVtqHypVPWAN3VB",
  "iss": "https://xyz.com",
  "iat": 1566583294,
  "exp": 1566583354,
  "auth_time": 1566583292,
  "https://aws.amazon.com/source_identity": "Admin"
}
```

ソース ID 情報を使用してアクセスを制御する

ソース ID が最初に設定されると、[sts:SourceIdentity](#) キーがリクエストに存在します。ソース ID が設定されると、[aws:SourceIdentity](#) キーは、ロールセッション中に行われた後続のすべての要求に存在します。管理者は、ソース ID 属性の存在または値に基づいて AWS アクションを実行するための条件付き承認を付与するポリシーを作成できます。

開発者にソースIDを設定して、本番環境の重要な AWS リソースへの書き込み権限を持つ重要なロールを引き受けるようにリクエストするとします。また、AWS を使用してワークフォース ID への AssumeRoleWithSAML アクセスを許可するとします。上級開発者の Saanvi と Diego にロールへのアクセス権のみを与えるため、ロールに対して次の信頼ポリシーを作成します。

Example ソース ID のロール信頼ポリシーの例 (SAML)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SAMLProviderAssumeRoleWithSAML",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-provider"
      }
    }
  ]
}
```

```

    },
    "Action": [
      "sts:AssumeRoleWithSAML"
    ],
    "Condition": {
      "StringEquals": {
        "SAML:aud": "https://signin.aws.amazon.com/saml"
      }
    }
  },
  {
    "Sid": "SetSourceIdentitySrEngs",
    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
    },
    "Action": [
      "sts:SetSourceIdentity"
    ],
    "Condition": {
      "StringLike": {
        "sts:SourceIdentity": [
          "Saanvi",
          "Diego"
        ]
      }
    }
  }
]
}

```

信頼ポリシーには、重要なロールを引き受けるために Saanvi または Diego のソース ID を必要とする、sts:SourceIdentity の条件が含まれています。

または、OIDC フェデレーションに OIDC プロバイダーを使用している場合、AssumeRoleWithWebIdentity によってユーザーが認証される場合、信頼ポリシーは次のようになります。

Example ソース ID のロール信頼ポリシーの例 (OIDC プロバイダー)

```

{
  "Version": "2012-10-17",

```

```
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws:iam::111122223333:oidc-provider/server.example.com"
    },
    "Action": [
      "sts:AssumeRoleWithWebIdentity",
      "sts:SetSourceIdentity"
    ],
    "Condition": {
      "StringEquals": {
        "server.example.com:aud": "oidc-audience-id"
      },
      "StringLike": {
        "sts:SourceIdentity": [
          "Saanvi",
          "Diego"
        ]
      }
    }
  }
]
```

## ロールチェーンとクロスアカウント要件

CriticalRole を想定したユーザーが別のアカウントで CriticalRole\_2 を想定できるようにしたいとします。引き受けるために取得されたロールセッションの資格情報 CriticalRole は、[ロールの連鎖](#)を 2 番目のロール、CriticalRole\_2 別のアカウントで。ロールは、アカウントの境界を越えて引き継がれています。したがって、sts:SetSourceIdentity のアクセス許可は、CriticalRole のアクセス許可ポリシーと CriticalRole\_2 のロール信頼ポリシーの両方で付与する必要があります。

## Example CriticalRole のアクセス権限ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleAndSetSourceIdentity",
      "Effect": "Allow",
      "Action": [
```



```
    "sts:AssumeRole",
    "sts:SetSourceIdentity"
  ],
  "Resource": "arn:aws:iam::222222222222:role/CriticalRole_2"
}
]
```

アカウントの境界を越えてソースIDの設定を保護するために、次のロール信頼ポリシーは、CriticalRole のロールプリンシパルのみを信頼してソース ID を設定します。

Example CriticalRole\_2 のロール信頼ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/CriticalRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi", "Diego"]
        }
      }
    }
  ]
}
```

ユーザーは、CriticalRole を引き受けることで取得したロールセッション資格情報を使用して、次の呼び出しを行います。ソース ID は、CriticalRole の仮定時に設定されているので、明示的に再度設定する必要はありません。ユーザーがソース ID を設定しようとした場合、CriticalRole が想定された場合、ロールの引き受けリクエストは拒否されます。

Example AssumeRole CLI リクエストの例

```
aws sts assume-role \
```

```
--role-arn arn:aws:iam::222222222222:role/CriticalRole_2 \  
--role-session-name Audit \  

```

呼び出し元プリンシパルがロールを引き受けると、リクエスト内のソース ID は、最初に引き受けたロールセッションから保持されます。したがって、aws:SourceIdentity キーと sts:SourceIdentity キーの両方がリクエストコンテキストに存在します

### CloudTrail でのソースアイデンティティの表示

CloudTrail を使用して、ロールを引き受けるか、ユーザーをフェデレートするために行われたリクエストを表示できます。AWS でアクションを実行するための役割またはユーザーのリクエストを表示することもできます。CloudTrail ログファイルには、引き受けたロールまたはフェデレーティッドユーザーセッションのプリンシパルタグに関する情報が含まれます。詳細については、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」を参照してください。

例えば、ユーザーが AWS STS AssumeRole リクエストを行い、ソースIDを設定するとします。sourceIdentity 情報は、CloudTrail ログの requestParameters キーで見つけることができます。

### Example AWS CloudTrail ログの requestParameters セクションの例

```
"eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AWSAccount",  
    "principalId": "AIDAJ45Q7YFFAREXAMPLE",  
    "accountId": "111122223333"  
  },  
  "eventTime": "2020-04-02T18:20:53Z",  
  "eventSource": "sts.amazonaws.com",  
  "eventName": "AssumeRole",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "203.0.113.64",  
  "userAgent": "aws-cli/1.16.96 Python/3.6.0 Windows/10 botocore/1.12.86",  
  "requestParameters": {  
    "roleArn": "arn:aws:iam::123456789012:role/DevRole",  
    "roleSessionName": "Dev1",  
    "sourceIdentity": "source-identity-value-set"  
  }  
}
```

ユーザーが引き受けたロールセッションを使用してアクションを実行する場合、ソース ID 情報は userIdentity キーを CloudTrail ログに追加します。

## Example AWS CloudTrail ログの userIdentity キーの例

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAJ45Q7YFFAREXAMPLE:Dev1",
    "arn": "arn:aws:sts::123456789012:assumed-role/DevRole/Dev1",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/DevRole",
        "accountId": "123456789012",
        "userName": "DevRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-02-21T23:46:28Z"
      },
      "sourceIdentity": "source-identity-value-present"
    }
  }
}
```

AWS STS CloudTrail ログの API イベントの例を見るには、「[CloudTrail ログの IAM API イベントの例](#)」を参照してください。CloudTrail ログファイルに含まれる情報の詳細については、[AWS CloudTrail ユーザーガイド](#) の「CloudTrail イベントリファレンス」を参照してください。

## GetFederationToken のアクセス権限

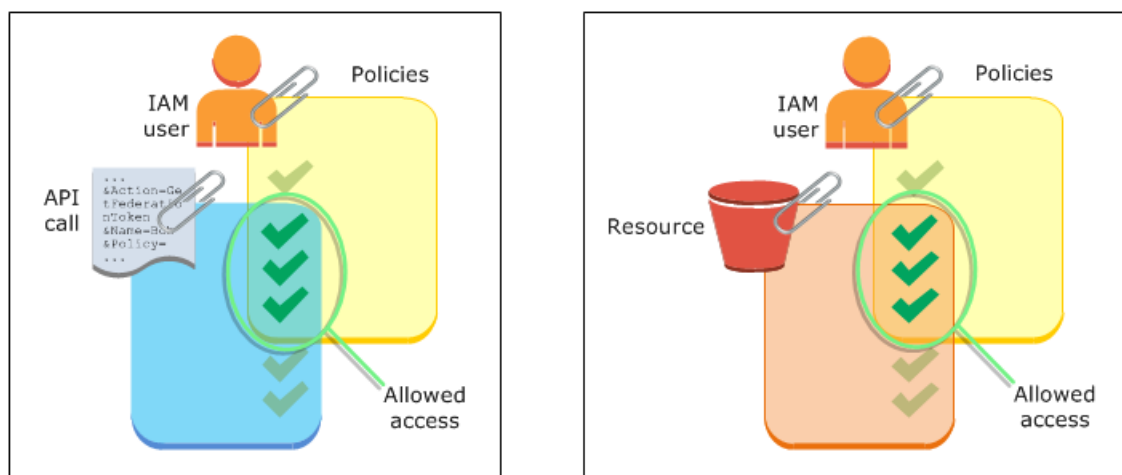
GetFederationToken オペレーションは、IAM ユーザーによって呼び出され、そのユーザーの一時的な認証情報を返します。このオペレーションでは、ユーザーをフェデレーションします。フェデレーテッドユーザーが割り当てられたアクセス権限は、次の 2 か所のどちらかで定義されます。

- GetFederationToken API コールのパラメータとして渡されるセッションポリシー。(こちらが普通です)。
- ポリシーの Principal 要素でフェデレーテッドユーザーを明示的に指名するリソースベースのポリシー (こちらはそれほど一般的ではありません)。

セッションポリシーは、一時セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。フェデレーテッドユーザーセッションを作成し、セッションポリシーを渡すと、結果として得られるセッションのアクセス許可は、ユーザーのアイデンティティベースのポリシーとセッションポリシーの共通部分です。セッションポリシーを使用して、フェデレーションされているユーザーのアイデンティティベースのポリシーによって許可されている以上のアクセス許可を付与することはできません。

ほとんどの場合、GetFederationToken API 呼び出しでポリシーを渡さなければ、返される一時的なセキュリティ認証情報はアクセス権を持ちません。ただし、リソースベースのポリシーでは、セッションに追加のアクセス許可を提供できます。セッションを許可されたプリンシパルとして指定するリソースベースのポリシーを使用してリソースにアクセスできます。

次の図は、ポリシーがどのように相互作用して、GetFederationToken の呼び出しによって返される一時的なセキュリティ認証情報のアクセス権限が決まるかを視覚的に示しています。



例: GetFederationToken を使用してアクセス権限を割り当てる

GetFederationToken API アクションをさまざまなポリシーと共に使用できます。ここにいくつかの例を挙げます。

ポリシーを IAM ユーザーにアタッチするには

この例では、2つのバックエンドウェブサービスに頼ったブラウザベースのクライアントアプリケーションがあります。1つのバックエンドサービスは、独自の ID システムを使用してクライアントアプリケーションを認証する独自の認証サーバーです。もう1つのバックエンドサービスは、クライアントアプリケーションの機能の一部を提供する AWS サービスです。クライアントアプリケーションはサーバーによって認証され、サーバーは適切なアクセス許可ポリシーを作成または取得します。サーバーは、続いて、GetFederationToken API を呼び出して一時的なセキュリティ認証情報を取

得し、その認証情報をクライアントアプリケーションに返します。クライアントアプリケーションは、その後、一時的なセキュリティ認証情報を使って AWS サービスに対してリクエストを直接行うことができます。このアーキテクチャでは、AWS 長期的認証情報を埋め込まなくても、クライアントアプリケーションが AWS リクエストを実行できます。

認証サーバーは、`GetFederationToken` という名前の IAM ユーザーの長期的なセキュリティ認証情報を使用して `token-app` API を呼び出します。ただし、長期的な IAM ユーザー認証情報はサーバーにとどまり、決してクライアントには配布されません。以下の例のポリシーは、`token-app` IAM ユーザーにアタッチされ、フェデレーションユーザー (クライアント) が必要とする最も広範なアクセス権限を定義します。フェデレーテッドユーザーの一時的なセキュリティ認証情報を取得するには、認証サービスに `sts:GetFederationToken` アクセス許可が必要となることに注意してください。

#### Note

AWS にはこの目的にかなったサンプル Java アプリケーションが用意されていて、「[ID 登録のためのトークン自動販売機 - サンプル Java ウェブアプリケーション](#)」からダウンロードできます。

Example `token-app` を呼び出す IAM ユーザー `GetFederationToken` にアタッチされたポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:GetFederationToken",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb>ListTables",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:ReceiveMessage",
      "Resource": "*"
    }
  ]
}
```

```
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:ListSubscriptions",
      "Resource": "*"
    }
  ]
}
```

前述のポリシーは、IAM ユーザーに複数のアクセス許可を付与します。ただし、このポリシー単独でフェデレーテッドユーザーにアクセス許可が付与されることはありません。この IAM ユーザーが `GetFederationToken` を呼び出して、API 呼び出しのパラメータとしてポリシーを渡さない場合、結果として、フェデレーションユーザーは有効なアクセス許可を持ちません。

#### パラメータとして渡されたセッションポリシー

フェデレーテッドユーザーに適切なアクセス許可を確実に割り当てる最も一般的な方法は、`GetFederationToken` API 呼び出しでセッションポリシーを渡すことです。前の例を拡張して、`GetFederationToken` が IAM ユーザー `token-app` の認証情報を使用して呼び出されると仮定します。次に、API 呼び出しのパラメータとして以下のセッションポリシーを渡すとしてします。結果として得られるフェデレーテッドユーザーは、`productionapp` という名前の Amazon S3 バケットのコンテンツを一覧表示するアクセス許可を持つことになります。ユーザーは、`productionapp` バケット内の項目に対して Amazon S3 `GetObject`、`PutObject`、および `DeleteObject` アクションを実行できません。

アクセス許可は IAM ユーザーポリシーとユーザーが渡すセッションポリシーとの共通部分であるため、フェデレーテッドユーザーにはこれらのアクセス許可が割り当てられます。

フェデレーテッドユーザーは、Amazon SNS、Amazon SQS、Amazon DynamoDB、または S3 バケットでアクションを行うことができませんでした (`productionapp` を除く)。これらのアクションは、このアクセス許可が `GetFederationToken` コールに関連付けられている IAM ユーザーに付与されている場合でも拒否されます。

#### Example `GetFederationToken` API 呼び出しのパラメータとして渡されるセッションポリシー

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": ["s3:ListBucket"],
    "Resource": ["arn:aws:s3:::productionapp"]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::productionapp/*"]
  }
]
```

## リソースベースのポリシー

一部の AWS リソースはリソースベースのポリシーをサポートし、このポリシーはフェデレーテッドユーザーにアクセス許可を直接付与するもう 1 つの仕組みとして利用できます。一部の AWS サービスのみでリソースに基づくポリシーをサポートしています。たとえば、Amazon S3 にはバケット、Amazon SNS にはトピック、Amazon SQS にはキューがあり、これらにポリシーをアタッチできます。リソースベースのポリシーをサポートするすべてのサービスのリストについては、「[IAM と連携する AWS のサービス](#)」を参照の上、テーブルの「リソースベースのポリシー」列を確認してください。リソースベースのポリシーを使用して、フェデレーションユーザーに直接アクセス許可を割り当てることができます。そのためには、リソースベースのポリシーの Principal 要素でフェデレーテッドユーザーの Amazon Resource Name (ARN) を指定します。以下の例では、これを表し、また productionapp という名前の S3 バケットを使用して前の例を拡張しています。

次のリソースベースのポリシーは、バケットにアタッチされています。このバケットポリシーでは、Carol という名前のフェデレーテッドユーザーにバケットへのアクセスを許可します。前に示したサンプルポリシーが token-app IAM ユーザーにアタッチされていると、Carol というフェデレーテッドユーザーには、s3:GetObject、s3:PutObject、s3:DeleteObject アクションを productionapp という名前のバケットに対して実行するアクセス許可が付与されています。これは、GetFederationToken API コールのパラメータとしてセッションポリシーが渡されていない場合にも当てはまります。Carol というフェデレーションユーザーが以下のリソースベースのポリシーによって明示的にアクセス権限を付与されているためです。

フェデレーテッドユーザーがアクセス許可を付与されるのは、IAM ユーザーとフェデレーテッドユーザーの両方に、明示的にアクセス許可が付与されている場合のみであることを忘れないでください。また、次の例のように、ポリシーの Principal 要素でフェデレーションユーザーを明示的に指名するリソースベースのポリシーでも (アカウント内で) 付与することができます。

#### Example フェデレーテッドユーザーにアクセスを許可するバケットポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {"AWS": "arn:aws:sts::account-id:federated-user/Carol"},
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::productionapp/*"]
    }
  ]
}
```

ポリシーの評価方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。

## GetSessionToken のアクセス権限

GetSessionToken API オペレーション、あるいは get-session-token CLI を呼び出す主な場面は、ユーザーを多要素認証 (MFA) で認証する必要がある場合です。MFA で認証されたユーザーが要求した場合にのみ、特定のアクションを許可するポリシーを作成することができます。MFA 認証チェックを正常に渡すには、ユーザーはまず GetSessionToken を呼び出し、オプションの SerialNumber および TokenCode パラメータを含める必要があります。ユーザーが MFA デバイスで正常に認証されている場合、GetSessionToken API オペレーションで返される認証情報には MFA コンテキストが含まれています。このコンテキストは、ユーザーが MFA で認証され、MFA 認証を必要とする API オペレーションへのアクセス権限があることを示します。

### GetSessionToken に必要なアクセス権限

ユーザーがセッショントークンを取得するために必要なアクセス権限はありません。GetSessionToken オペレーションの目的は、MFA を使用してユーザーを認証することです。認証オペレーションを制御するためにポリシーを使用することはできません。



ほとんどの AWS オペレーションを実行するためのアクセス権限を付与するには、ポリシーに同じ名前のアクションを追加します。たとえば、ユーザーを作成するには、CreateUser API オペレーション、create-user CLI コマンド、または AWS Management Console を使用する必要があります。これらのオペレーションを実行するには、CreateUser アクションにアクセスできるポリシーを持っている必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateUser",
      "Resource": "*"
    }
  ]
}
```

ポリシーに GetSessionToken アクションを含めることはできますが、これはユーザーが GetSessionToken オペレーションを実行する権限に影響を及ぼしません。

### GetSessionToken によるアクセス権限付与

GetSessionToken が IAM ユーザーの認証情報によって呼び出された場合、一時的なセキュリティ認証情報は IAM ユーザーと同じアクセス権限を持ちます。同様に、GetSessionToken が AWS アカウントのルートユーザー 認証情報によって呼び出された場合、一時的なセキュリティ認証情報はルートユーザーのアクセス許可を持ちます。

#### Note

GetSessionToken は、ルートユーザー認証情報を使用して呼び出さないようお勧めします。代わりに、[ベストプラクティス](#)に従って、必要なアクセス許可を持つ IAM ユーザーを作成します。AWS との日常的なやり取りには、これらの IAM ユーザーを使用します。

GetSessionToken を呼び出すときに取得する一時的な認証情報には、次の機能と制限がありません。

- フェデレーションのシングルサインオンエンドポイント <https://signin.aws.amazon.com/federation> に認証情報を渡すことで AWS Management Console にアクセスできません。詳細に

については、「[カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

- 認証情報を使用して IAM または AWS STS API オペレーションを呼び出すことはできません。認証情報を使用してその他の サービスの API オペレーションを呼び出すことはできますAWS。

[AWS STS API オペレーションの比較](#) で、この API オペレーションおよびその制限と機能を、一時的なセキュリティ認証情報を作成する他の API と比較してください。

GetSessionToken を使用した MFA 保護 API アクセスの詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

## 一時的なセキュリティ認証情報のアクセス権限を無効にする

一時的なセキュリティ認証情報は、期限が切れるまで有効です。これらの認証情報は、900 秒 (15 分) から最大 129,600 秒 (36 時間) までの指定された期間有効です。デフォルトのセッション時間は 43,200 秒 (12 時間) です。これらの認証情報は取り消すことができますが、認証情報を漏洩させて悪意あるアカウントの活動に利用されないようにロールのアクセス許可を変更する方法もあります。一時的なセキュリティ認証情報に割り当てられるアクセス許可は、AWS のリクエストの実行に使用されるたびに評価されます。認証情報からすべてのアクセス許可を削除すると、それらを使用している AWS のリクエストは失敗します。

ポリシーの更新が有効になるまでには、数分かかる場合があります。[ロールの一時的なセキュリティ認証情報を取り消し](#)、ロールを引き受けるすべてのユーザーに新しい認証情報の再認証およびリクエストを強制します。

AWS アカウントのルートユーザー のアクセス許可を変更することはできません。同様に、ルートユーザーとしてサインインしているときに GetFederationToken または GetSessionToken を呼び出して作成した一時的なセキュリティ認証情報のアクセス許可を変更することはできません。そのため、ルートユーザーとして GetFederationToken または GetSessionToken を呼び出さないようお勧めします。

### Important

IAM アイデンティティセンターのアクセス許可セットから作成された IAM のロールを編集することはできません。IAM アイデンティティセンターでユーザーのアクティブなアクセス許可セットセッションを取り消す必要があります。詳細については、「IAM Identity Center ユーザーガイド」の「[Revoke active IAM role sessions created by permission sets](#)」を参照してください。

## トピック

- [ルールに関連するすべてのセッションへのアクセスを拒否する](#)
- [特定のセッションへのアクセスを拒否する](#)
- [条件コンテキストキーを使用してユーザーセッションを拒否する](#)
- [リソースベースのポリシーでセッションユーザーを拒否する](#)

### ルールに関連するすべてのセッションへのアクセスを拒否する

次のような不審なアクセスが懸念される場合は、この方法を使用します。

- クロスアカウントアクセスを使用している別のアカウントのプリンシパル
- アカウント内の AWS リソースへのアクセス許可を持つ外部ユーザー ID
- OIDC プロバイダーを使用して、モバイルまたはウェブアプリケーションで認証されているユーザー

このプロシージャでは、ルールを引き受けるアクセス許可を持つすべてのユーザーのアクセス許可を拒否します。

AssumeRole、AssumeRoleWithSAML、また

は AssumeRoleWithWebIdentity、GetFederationToken、または GetSessionToken を呼び出して取得した一時的なセキュリティ認証情報に割り当てたアクセス許可を変更または削除するには、ルールのアクセス許可を定義するアクセス許可ポリシーを編集または削除します。

#### Important

また、プリンシパルのアクセスを許可するリソースベースのポリシーがある場合、そのリソースに対する明示的な拒否を追加する必要があります。詳細については、「[リソースベースのポリシーでセッションユーザーを拒否する](#)」を参照してください。

1. AWS Management Console にサインインし、IAM コンソール を開きます。
2. ナビゲーションペインで、編集するユーザーの名前を選択します。検索ボックスを使用してリストをフィルタリングします。
3. 該当するポリシーを選択します。
4. [アクセス許可] タブを選択します。

5. [JSON] タブを選択してポリシーを更新し、すべてのリソースとアクションを拒否します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

6. [確認] ページで、ポリシーの [概要] を確認してから、[変更の保存] を選択して作業を保存します。

ロールのポリシーを更新すると、変更内容はそのロールに関連付けられているすべての一時的なセキュリティ認証情報のアクセス許可に影響します。これには、ロールのアクセス許可ポリシーを変更する前に発行された認証情報も含まれます。ポリシーを更新すると、[ロールの一時的なセキュリティ認証情報を取り消し](#)、ロールが発行した認証情報に対するすべてのアクセス許可をすぐに取り消すことができます。

#### 特定のセッションへのアクセスを拒否する

deny-all (すべて拒否) ポリシーを使用して IdP から引き継がれるロールを更新したり、ロールを完全に削除したりすると、そのロールにアクセスできるすべてのユーザーのアクセスが中断されます。ロールに関連する他のすべてのセッションのアクセス許可に影響を与えずに、Principal 要素に基づいてアクセスを拒否できます。

Principal は、[条件コンテキストキー](#)または[リソースベースのポリシー](#)を使用してアクセス許可を拒否できます。

#### Tip

AWS CloudTrail ログを使用して、フェデレーションユーザーの ARN を確認できます。詳細については、「[AWS CloudTrail を使用してフェデレーションユーザーを簡単に識別する方法](#)」を参照してください。

## 条件コンテキストキーを使用してユーザーセッションを拒否する

条件コンテキストキーは、認証情報を作成した IAM ユーザーまたはロールのアクセス許可に影響を与えることなく一時的なセキュリティ認証情報へのアクセスを拒否したい場合に使用できます。

条件コンテキストキーの詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

### Note

また、プリンシパルのアクセスを許可するリソースベースのポリシーがある場合、これらの手順の完了後にリソースベースのポリシーで明示的な拒否を追加する必要があります。

ポリシーの更新後は、[ロールの一時的なセキュリティ認証情報を取り消し](#)、すべての発行された認証情報をすぐに取り消すことができます。

aws:PrincipalArn

条件コンテキストキー [aws:PrincipalArn](#) を使用して、特定のプリンシパル ARN のアクセスを拒否できます。これを行うには、ポリシーの Condition 要素で、一時的なセキュリティ認証情報が関連付けられている IAM ユーザー、ロール、またはフェデレーションユーザーの一意の識別子 (ID) を指定します。

1. IAM コンソールのナビゲーションペインで、編集するロールの名前を選択します。検索ボックスを使用してリストをフィルタリングします。
2. 該当するポリシーを選択します。
3. [アクセス許可] タブを選択します。
4. [JSON] タブを選択し、次の例に示すようにプリンシパル ARN の拒否ステートメントを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
```

```
"aws:PrincipalArn": [  
  "arn:aws:iam::222222222222:role/ROLENAME",  
  "arn:aws:iam::222222222222:user/USERNAME",  
  "arn:aws:sts::222222222222:federated-user/USERNAME"  
]  
}  
}  
]  
}
```

5. [確認] ページで、ポリシーの [概要] を確認してから、[変更の保存] を選択して作業を保存します。

#### aws:userid

条件コンテキストキー [aws:userid](#) を使用して、IAM ユーザーまたはロールに関連する一時的なセキュリティ認証情報のすべてまたは特定のセッションへのアクセスを拒否できます。これを行うには、ポリシーの Condition 要素で、一時的なセキュリティ認証情報が関連付けられている IAM ユーザー、ロール、またはフェデレーションユーザーの一意の識別子 (ID) を指定します。

次のポリシーは、条件コンテキストキー `aws:userid` を使用して一時的なセキュリティ認証情報のセッションへのアクセスを拒否する方法の例を示しています。

- AIDAXUSER1 は、IAM ユーザー用の一意の識別子を表します。IAM ユーザーの一意の識別子をコンテキストキー `aws:userid` の値として指定すると、IAM ユーザーに関連するすべてのセッションが拒否されます。
- AROAXROLE1 は、IAM ロール用の一意の識別子を表します。IAM ロールの一意の識別子をコンテキストキー `aws:userid` の値として指定すると、そのロールに関連するすべてのセッションが拒否されます。
- AROAXROLE2 は、assumed-role セッション用の一意の識別子を表します。assumed-role の一意の識別子の caller-specified-role-session-name の部分で、ロールのセッション名を指定するか、StringLike 条件演算子を使用する場合はワイルドカード文字を指定できます。ロールのセッション名を指定すると、認証情報を作成したロールのアクセス許可に影響を与えずに、指定されたロールのセッションが拒否されます。ロールのセッション名にワイルドカードを指定すると、そのロールに関連するすべてのセッションが拒否されます。
- `account-id:<federated-user-caller-specified-name>` は、フェデレーションユーザーのセッション用の一意の識別子を表します。フェデレーションユーザーは、GetFederationToken API を呼び出す IAM ユーザーによって作成されます。フェデレーションユーザーの一意の識別子

を指定すると、認証情報を作成したロールのアクセス許可に影響を与えずに、フェデレーションユーザーのセッションが拒否されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userId": [
            "AIDAXUSER1",
            "AROAXROLE1",
            "AROAXROLE2:<caller-specified-role-session-name>",
            "account-id:<federated-user-caller-specified-name>"
          ]
        }
      }
    }
  ]
}
```

プリンシパルキーの値の具体的な例については、「[プリンシパルキーの値](#)」を参照してください。IAM の一意の識別子についての詳細は、「[一意の識別子](#)」を参照してください。

### リソースベースのポリシーでセッションユーザーを拒否する

リソースベースのポリシーにプリンシパル ARN も含まれている場合、リソースベースのポリシーにある Principal 要素の特定のユーザーの principalId または sourceIdentity の値に基づきアクセス権を取り消す必要もあります。ロールのアクセス許可ポリシーのみを更新した場合でも、ユーザーはリソースベースのポリシーで許可されているアクションを実行できます。

1. サービスがリソースベースのポリシーをサポートしているかどうかについては、「[IAM と連携する AWS のサービス](#)」を確認してください。
2. AWS Management Console にサインインして、サービスのコンソールを開きます。ポリシーをアタッチするコンソール内の場所は、サービスごとに異なります。
3. ポリシーステートメントを編集して、認証情報の識別する情報を指定します。

- a. Principal に、拒否する認証情報の ARN を入力します。
- b. Effect に、「Deny」と入力します。
- c. Action に、サービスの名前空間と拒否するアクションの名前を入力します。すべてのアクションを拒否するには、ワイルドカード (\*) 文字を使用します。たとえば、「s3:\*」と入力します。
- d. Resource に、ターゲットリソースの ARN を入力します。例えば、「arn:aws:s3:::EXAMPLE-BUCKET」と入力します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": [
      "arn:aws:iam::222222222222:role/ROLENAME",
      "arn:aws:iam::222222222222:user/USERNAME",
      "arn:aws:sts::222222222222:federated-user/USERNAME"
    ],
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::EXAMPLE-BUCKET"
  }
}
```

4. 作業内容を保存します。

## 一時的なセキュリティ認証情報を作成するためのアクセス権限の付与

デフォルトで、IAM ユーザーには、フェデレーテッドユーザーおよびロールの一時的なセキュリティ認証情報を作成するアクセス許可がありません。ユーザーに上記の権限を提供するポリシーを使用する必要があります。ユーザーに直接アクセス権限を付与できますが、アクセス権限はグループに付与することを強くお勧めします。これによって、アクセス権限の管理が容易になります。ユーザーがアクセス権限に関連付けられているタスクを実行する必要がなくなった場合には、そのユーザーをグループから削除するだけです。他のユーザーがそのタスクを実行する必要がある場合には、そのユーザーをグループに追加して、アクセス権限を付与します。

フェデレーテッドユーザーまたはロールの一時的なセキュリティ認証情報を作成するアクセス許可を IAM グループに付与するには、次の権限の少なくとも 1 つを付与するポリシーをアタッチします。



- IAM ロールにアクセスするフェデレーテッドユーザーには、AWS STS AssumeRole に対するアクセス許可を付与します。
- ロールを必要としないフェデレーテッドユーザーには、AWS STS GetFederationToken に対するアクセス許可を付与します。

AssumeRole および GetFederationToken の API オペレーションの違いの詳細については、「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

また、IAM ユーザーは、一時的なセキュリティ認証情報を作成するために [GetSessionToken](#) を呼び出すこともできます。ユーザーが GetSessionToken を呼び出すためには、アクセス権限を必要としません。このオペレーションの目的は、MFA を使用してユーザーを認証することです。認証を制御するためにポリシーを使用することはできません。つまり、IAM ユーザーが GetSessionToken を呼び出して、一時的な認証情報を作成することを回避することはできません。

#### Example ロールを引き受けるアクセス許可を付与するポリシー

以下のポリシーの例では、AWS アカウント 123123123123 の UpdateApp ロールに対して AssumeRole を呼び出すアクセス許可が与えられます。AssumeRole を使用する場合、フェデレーテッドユーザーの代わりにセキュリティ認証情報を作成するユーザー (またはアプリケーション) は、ロールのアクセス許可ポリシーに指定されていないあらゆるアクセス許可を委任することができません。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"
  }]
}
```

#### Example フェデレーテッドユーザーの一時的なセキュリティ認証情報を作成するアクセス権限を付与するポリシーの例

次のポリシーの例では、GetFederationToken にアクセスできるアクセス許可が付与されます。

```
{
  "Version": "2012-10-17",
  "Statement": [{
```

```
"Effect": "Allow",
"Action": "sts:GetFederationToken",
"Resource": "*"
}]
}
```

### Important

GetFederationToken を使用して、IAM ユーザーにアクセス権限を与えてフェデレーテッドユーザーの一時的なセキュリティ認証情報を作成する場合は、この権限によってユーザーは自身の権限を委任できるようになることに注意してください。IAM ユーザーや AWS アカウント へのアクセス許可の委任については、「[アクセス権を委任するポリシーの例](#)」を参照してください。一時的なセキュリティ認証情報のアクセス許可を制御する方法の詳細については、「[一時的なセキュリティ認証情報のアクセス権限を制御する](#)」を参照してください。

Example フェデレーテッドユーザーの一時的セキュリティ認証情報を作成するユーザーを限定するアクセス許可を付与するポリシーの例

IAM ユーザーが GetFederationToken 呼び出しをできるようにする場合、IAM ユーザーに委任できる権限を制限することがベストプラクティスです。たとえば、次のポリシーは、名前が Manager で始まるフェデレーテッドユーザーの場合にのみ、IAM ユーザーに一時的なセキュリティ認証情報の作成を許可する方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]
  }]
}
```

## ID 対応コンソールセッションを使用するアクセス許可の付与

ID 対応コンソールセッションでは、AWS IAM Identity Center ユーザーおよびセッション ID を、サインイン時にユーザーの AWS コンソールセッションに含めることができます。例えば、Amazon Q Developer Pro では、ID 対応コンソールセッションを使用してサービスエクスペリエンスをパーソナライズします。ID 対応コンソールセッションについて詳しくは、「AWS IAM Identity Center

ユーザーガイド」の「[Enabling identity-aware console sessions](#)」を参照してください。Amazon Q Developer のセットアップについては、「Amazon Q Developer ユーザーガイド」の「[Setting up Amazon Q Developer](#)」を参照してください。

ID 対応コンソールセッションをユーザーが利用できるようにするには、ID ベースのポリシーを使用して、自分のコンソールセッションを表すリソースに対する `sts:SetContext` アクセス許可を IAM プリンシパルに付与する必要があります。

### ⚠ Important

デフォルトでは、ユーザーには ID 対応コンソールセッションのコンテキストを設定するためのアクセス許可はありません。これを許可するには、以下のポリシー例に示すように、ID ベースのポリシーで `sts:SetContext` アクセス許可を IAM プリンシパルに付与する必要があります。

以下の ID ベースのポリシー例では、`sts:SetContext` アクセス許可を IAM プリンシパルに付与することで、プリンシパルが自身の AWS コンソールセッションに対して ID 対応コンソールセッションのコンテキストを設定できるようにします。ポリシーリソース `arn:aws:sts::account-id:self` は、呼び出し元の AWS セッションを表します。IAM Identity Center のアクセス許可セットを使用してこのポリシーをデプロイする場合など、同じアクセス許可ポリシーが複数のアカウントにデプロイされる場合は、`account-id` ARN セグメントをワイルドカード文字 `*` に置き換えることができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:SetContext",
      "Resource": "arn:aws:sts::account-id:self"
    }
  ]
}
```

## AWS リージョンでの AWS STS の管理

デフォルトでは、AWS Security Token Service (AWS STS) は、グローバルなサービスと使用することができ、AWS STS リクエストはすべて、`https://sts.amazonaws.com` の単一エンドポイント

トに送信されます。AWS では、レイテンシーの低減、冗長性の構築、セッショントークンの有効性の強化のために、グローバルエンドポイントではなく、リージョンの AWS STS エンドポイントを使用することを推奨しています。

- レイテンシーの低減 – お客様のサービスやアプリケーションに地理的に近いエンドポイントに対して AWS STS の呼び出しを実行することにより、より低いレイテンシーとより高速な応答時間で AWS STS サービスにアクセスできます。
- 冗長性の構築 – 予測可能な範囲に影響を封じ込めることによって、ワークロード内の障害の影響を限られた数のコンポーネントに限定できます。リージョン AWS STS エンドポイントを使用すると、コンポーネントの範囲をセッショントークンの範囲に合わせることができます。この信頼性の柱の詳細については、「AWS Well-Architected Framework」の「[障害部分を切り離してワークロードを保護する](#)」を参照してください。
- セッショントークンの有効性を向上させる – リージョンの AWS STS エンドポイントからのセッショントークンはすべての AWS リージョン で有効です。グローバル STS エンドポイントからのセッショントークンは、デフォルトで有効になっている AWS リージョン でのみ有効です。アカウントで新しいリージョンを有効にする場合、リージョン別 AWS STS エンドポイントからのセッショントークンを使用できます。グローバルエンドポイントの使用を選択した場合、グローバルエンドポイントに対する AWS STS セッショントークンのリージョンの互換性を変更する必要があります。これにより、トークンはすべての AWS リージョン で有効になります。


## グローバルエンドポイントセッショントークンの管理

ほとんどの AWS リージョン はデフォルトですべての AWS のサービスのオペレーションに有効になっています。これらのリージョンは、AWS STS で使用できるように自動的にアクティブ化されます。アジアパシフィック (香港) など一部のリージョンは、手動で有効にする必要があります。AWS リージョン を有効および無効にする方法については、「AWS Account Management リファレンスガイド」の「[アカウントで使用できる AWS リージョン の指定](#)」を参照してください。これらの AWS リージョンを有効にすると、AWS STS で使用できるように、自動的にアクティブ化されます。無効になっているリージョンの AWS STS エンドポイントをアクティブ化することはできません。すべての AWS リージョン で有効なセッショントークンには、デフォルトで有効になっているリージョンで有効なトークンを超える文字が含まれています。この設定を変更すると、一時的にトークンを保存する既存のシステムに影響する可能性があります。

この設定は、AWS Management Console、AWS CLI、または AWS API を使用して変更できます。

グローバルエンドポイント (コンソール) に対するセッショントークンのリージョンの互換性を変更するには

1. IAM 管理タスクを実行するアクセス許可があるルートユーザーまたはユーザーとしてサインインします。セッショントークンの互換性を変更するには、`iam:SetSecurityTokenServicePreferences` アクションを許可するポリシーがある必要があります。
2. [\[IAM コンソール\]](#) を開きます。ナビゲーションペインで [\[アカウント設定\]](#) を選択します。
3. [\[Security Token Service \(STS\)\]](#) セクションの [\[Session Tokens from the STS endpoints\]](#) (STS エンドポイントからのセッショントークン)。[\[Global endpoint\]](#) (グローバルエンドポイント) は `Valid only in AWS ##### enabled by default` を示します。[\[Change\]](#) を選択します。
4. [\[Change region compatibility\]](#) (リージョンの互換性を変更) ダイアログボックスで、[\[All AWS リージョン\]](#) を選択します。次に、変更の保存を選択します。

 Note

すべての AWS リージョン で有効なセッショントークンには、デフォルトで有効になっているリージョンで有効なトークンを超える文字が含まれています。この設定を変更すると、一時的にトークンを保存する既存のシステムに影響する可能性があります。

グローバルエンドポイント (AWS CLI) に対するセッショントークンのリージョンの互換性を変更するには

セッショントークンのバージョンを設定します。バージョン 1 トークンは、デフォルトで利用できる AWS リージョン でのみ有効です。これらのトークンは、アジアパシフィック (香港) など、手動で有効になっているリージョンでは動作しません。バージョン 2 のトークンはすべてのリージョンで有効です。ただし、バージョン 2 トークンにはさらに多くの文字が含まれており、一時的にトークンを保存するシステムに影響する可能性があります。

- [aws iam set-security-token-service-preferences](#)

グローバルエンドポイント (AWS API) に対するセッショントークンのリージョンの互換性を変更するには

セッショントークンのバージョンを設定します。バージョン 1 トークンは、デフォルトで利用できる AWS リージョン でのみ有効です。これらのトークンは、アジアパシフィック (香港) など、手動

で有効になっているリージョンでは動作しません。バージョン 2 のトークンはすべてのリージョンで有効です。ただし、バージョン 2 トークンにはさらに多くの文字が含まれており、一時的にトークンを保存するシステムに影響する可能性があります。

- [SetSecurityTokenServicePreferences](#)

## AWS リージョン での AWS STS のアクティブ化と非アクティブ化

リージョンに対して STS を有効にすると、AWS STS は、AWS STS リクエストを行うアカウントのユーザーとロールに一時的な認証情報を発行できます。その後、これらの認証情報は、デフォルトで有効であるリージョン、または手動で有効にされているリージョンで使用できます。デフォルトで有効になっているリージョンでは、一時認証情報が生成されるアカウントでリージョン STS エンドポイントをアクティブ化する必要があります。リクエストを行うときに、ユーザーが同じアカウントにサインインしたかまたは別のアカウントにサインインしたかは関係ありません。リージョンが手動でアクティブ化される場合は、リージョンは、リクエストを行うアカウントと一時的な認証情報が生成されるアカウントの両方でアクティブ化する必要があります。

例えば、アカウント A 内のあるユーザーが AWS STS リージョンのエンドポイント `https://sts.ap-east-1.amazonaws.com` に `sts:AssumeRole` API リクエストを送信するとします。このリクエストは、アカウント B にある `Developer` という名前のロール用の一時的な認証情報を求めるものです。これはアカウント B 内のエンティティの認証情報を作成するリクエストであるため、アカウント B が `ap-east-1` リージョンをアクティブ化する必要があります。アカウント A (または他のアカウント) のユーザーは、`ap-east-1` AWS STS エンドポイントを呼び出して、自分のアカウントでこのリージョンがアクティブ化されているかどうかに関わらず、アカウント B の認証情報をリクエストできます。

### Note

アクティブなリージョンはそのアカウントで一時的な認証情報を使用するすべてのユーザーが利用できます。どの IAM ユーザーまたはロールがリージョンにアクセスできるかを制御するには、アクセス許可ポリシーで、[aws:RequestedRegion](#) 条件キーを使用します。

デフォルトで有効なリージョンで AWS STS をアクティブ化または非アクティブ化するには (コンソール)

1. IAM 管理タスクを実行するアクセス許可があるルートユーザーまたはユーザーとしてサインインします。

2. [IAM コンソール](#)を開き、ナビゲーションペインで [\[アカウント設定\]](#) を選択します。
3. [Security Token Service (STS)] セクションの [Endpoints] (エンドポイント) で、設定するリージョンを見つけ、[STS status] (STS ステータス) 列で [Active] (アクティブ) または [Inactive] (非アクティブ) を選択します。
4. 表示されたダイアログボックスで、[Activate] (有効化) または [Deactivate] (無効化) を選択します。

有効にする必要があるリージョンの場合、リージョンを有効にすると AWS STS が自動的にアクティブになります。リージョンを有効にすると、AWS STS はそのリージョンに対して常にアクティブになり、非アクティブ化することはできません。デフォルトで無効になっているリージョンを有効にする方法については、「AWS Account Management リファレンスガイド」の「[アカウントで利用できる AWS リージョンの指定](#)」を参照してください。

## AWS STS リージョンを使用するコードの記述

リージョンをアクティブ化すると、そのリージョンに AWS STS API 呼び出しを割り振ることができます。次の Java コードスニペットは、欧州 (ミラノ)(eu-south-1) リージョンにリクエストを送信するように AWSSecurityTokenService オブジェクトを設定する方法を示しています。

```
EndpointConfiguration regionEndpointConfig = new EndpointConfiguration("https://sts.eu-south-1.amazonaws.com", "eu-south-1");
AWSSecurityTokenService stsRegionalClient =
    AWSSecurityTokenServiceClientBuilder.standard()
        .withCredentials(credentials)
        .withEndpointConfiguration(regionEndpointConfig)
        .build();
```

AWS STS では、リージョンのエンドポイントへの呼び出しを推奨します。手動でリージョンを有効にする方法については、「AWS Account Management リファレンスガイド」の「[アカウントで利用できる AWS リージョンの指定](#)」を参照してください。

この例では、最初の行は regionEndpointConfig という EndpointConfiguration オブジェクトをインスタンス化し、エンドポイントの URL と AWS リージョン をパラメータとして渡します。

AWS SDK の環境変数を使用して AWS STS のリージョンエンドポイントを設定する方法については、「AWS SDK とツールリファレンスガイド」の「[AWS STS リージョンエンドポイント](#)」を参照してください。



他のすべての言語とプログラミング環境の組み合わせについては、「[関連する SDK のドキュメント](#)」を参照してください。

## リージョンとエンドポイント

次の表に、リージョンとそのエンドポイントを一覧表示します。ここでは、デフォルトでアクティブ化されるものや、ユーザーがアクティブ化または非アクティブ化できるものが示されています。

リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
--グローバル--	st.amazonaws.com	 はい	 いいえ
米国東部 (オハイオ)	sts.us-east-2.amazonaws.com	 はい	 はい
米国東部 (バージニア北部)	sts.us-east-1.amazonaws.com	 はい	 いいえ
米国西部 (北カリフォルニア)	sts.us-west-1.amazonaws.com	 はい	 はい
米国西部 (オレゴン)	sts.us-west-2.amazonaws.com	 はい	 はい









リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
アフリカ (ケープタウン)	sts.af-south-1.amazonaws.com	 No	 いいえ
アジアパシフィック (香港)	sts.ap-east-1.amazonaws.com	 No	 いいえ
アジアパシフィック (ハイデラバード)	sts.ap-south-2.amazonaws.com	 No	 いいえ
アジアパシフィック (ジャカルタ)	sts.ap-southeast-3.amazonaws.com	 No	 いいえ
アジアパシフィック (メルボルン)	sts.ap-southeast-4.amazonaws.com	 No	 いいえ
アジアパシフィック (ムンバイ)	sts.ap-south-1.amazonaws.com	 はい	 はい

リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
アジアパシフィック (大阪)	sts.ap-northeast-3.amazonaws.com	 はい	 はい
アジアパシフィック (ソウル)	sts.ap-northeast-2.amazonaws.com	 はい	 はい
アジアパシフィック (シンガポール)	sts.ap-southeast-1.amazonaws.com	 はい	 はい
アジアパシフィック (シドニー)	sts.ap-southeast-2.amazonaws.com	 はい	 はい
アジアパシフィック (東京)	sts.ap-northeast-1.amazonaws.com	 はい	 はい
カナダ (中部)	sts.ca-central-1.amazonaws.com	 はい	 はい

リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
カナダ西部 (カルガリー)	sts.ca-west-1.amazonaws.com	 はい	 はい
中国 (北京)	sts.cn-north-1.amazonaws.com.cn	 はい <sup>2</sup>	 いいえ
中国 (寧夏)	sts.cn-northwest-1.amazonaws.com.cn	 はい <sup>2</sup>	 はい
欧州 (フランクフルト)	sts.eu-central-1.amazonaws.com	 はい	 はい
欧州 (アイルランド)	sts.eu-west-1.amazonaws.com	 はい	 はい
欧州 (ロンドン)	sts.eu-west-2.amazonaws.com	 はい	 はい

リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
欧州 (ミラノ)	sts.eu-south-1.amazonaws.com	 No <sup>1</sup>	 いいえ
欧州 (パリ)	sts.eu-west-3.amazonaws.com	 はい	 はい
欧州 (スペイン)	sts.eu-south-2.amazonaws.com	 No <sup>1</sup>	 いいえ
欧州 (ストックホルム)	sts.eu-north-1.amazonaws.com	 はい	 はい
欧州 (チューリッヒ)	sts.eu-central-2.amazonaws.com	 No <sup>1</sup>	 いいえ
イスラエル (テルアビブ)	sts.il-central-1.amazonaws.com	 No <sup>1</sup>	 いいえ

リージョン名	エンドポイント	デフォルトでアクティブ	手動でアクティブ化/非アクティブ化
中東 (バーレーン)	sts.me-south-1.amazonaws.com	 No <sup>1</sup>	 いいえ
中東 (アラブ首長国連邦)	sts.me-central-1.amazonaws.com	 No <sup>1</sup>	 いいえ
南米 (サンパウロ)	sts.sa-east-1.amazonaws.com	 はい	 はい

<sup>1</sup>リージョンで使用するには、[リージョンを有効にする](#)必要があります。これにより、AWS STS が自動的にアクティブになります。これらのリージョンで AWS STS を手動でアクティブ化または非アクティブ化することはできません。

<sup>2</sup>中国で AWS を使用するには、中国内の AWS に特化されたアカウントと認証情報が必要です。

## AWS CloudTrail とリージョンのエンドポイント

リージョンのエンドポイントとグローバルエンドポイントへの呼び出しは、AWS CloudTrail の [tlsDetails] フィールドに記録されます。us-east-2.amazonaws.com などのリージョンのエンドポイントへの呼び出しは、CloudTrail で適切なリージョンに記録されます。グローバルエンドポイント sts.amazonaws.com への呼び出しは、グローバルサービスへの呼び出しとして記録されます。グローバル AWS STS エンドポイントのイベントは us-east-1 に記録されます。

**Note**

tlsDetails は、このフィールドをサポートするサービスに対してのみ表示できます。AWS CloudTrail ユーザーガイドの「[CloudTrail で TLS の詳細をサポートするサービス](#)」を参照してください。  
詳細については、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」を参照してください。

## ベアラートークンを使用する

一部の AWS のサービスでは、プログラムでリソースにアクセスする前に、AWS STS サービスベアラートークンを取得するアクセス許可が必要です。これらのサービスは、従来の[署名バージョン 4 の署名付きリクエスト](#)を使用する代わりに、ベアラートークンの使用が必要なプロトコルをサポートします。ベアラートークンをリクエストする AWS CLI または AWS API オペレーションを実行すると、AWS のサービスはお客様に代わってベアラートークンをリクエストします。サービスによってトークンが提供され、このトークンを使用して、サービスで後続のオペレーションを実行できます。

AWS STS サービスベアラートークンには、アクセス許可に影響する可能性がある元のプリンシパル認証の情報が含まれます。この情報には、プリンシパルタグ、セッションタグ、セッションポリシーが含まれます。トークンのアクセスキー ID は、ABIA プレフィックスで始まります。これは、CloudTrail ログ内のサービスベアラートークンを使用して実行されたオペレーションを識別するのに役立ちます。

**Important**

ベアラートークンは、それを生成するサービスへの呼び出しと、それが生成されたリージョンでのみ使用できます。ベアラートークンを使用して、他のサービスやリージョンでオペレーションを実行することはできません。

ベアラートークンをサポートするサービスの例は、AWS CodeArtifact です。NPM、Maven、または PIP などのパッケージマネージャを使用して AWS CodeArtifact を操作する前に、aws codeartifact get-authorization-token オペレーションを呼び出す必要があります。このオペレーションは、AWS CodeArtifact オペレーションを実行するために使用できるベアラートークンを返します。または、同じオペレーションを完了し、クライアントを自動的に設定する aws codeartifact login コマンドを使用できます。

ベアラートークンを生成する AWS のサービスでアクションを実行する場合は、IAM ポリシーに次のアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowServiceBearerToken",
      "Effect": "Allow",
      "Action": "sts:GetServiceBearerToken",
      "Resource": "*"
    }
  ]
}
```

サービスのベアラートークンの例については、「AWS CodeArtifact ユーザーガイド」の「[Using identity-based policies for AWS CodeArtifact](#)」(でのアイデンティティベースのポリシーの使用)を参照してください。

## 一時認証情報を使用するサンプルアプリケーション

AWS Security Token Service (AWS STS) を使用して、AWS リソースへのアクセスをコントロールできる一時的セキュリティ認証情報を持つ、信頼されたユーザーを作成および提供することができます。AWS STS の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。AWS STS を使用して一時的なセキュリティ認証情報を管理する方法を確認するために、完全なシナリオ例を実装する以下のサンプルアプリケーションをダウンロードできます。

- [Windows Active Directory、ADFS、SAML 2.0 を使用して AWS へのフェデレーションを有効化する](#) Windows アクティブディレクトリ (AD)、アクティブディレクトリフェデレーションサービス (ADFS) 2.0、SAML (セキュリティアサーションマークアップ言語) 2.0 で、エンタープライズフェデレーションを使用して AWS へのアクセスを委任する方法を示します。
- [カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可](#)。シングルサインオン (SSO) を有効にするカスタムフェデレーションプロキシを作成して、既存の Active Directory ユーザーが AWS Management Console にサインインできるようにする方法を示します。
- [Shibboleth を使用して AWS Management Console へのシングルサインオンを行う方法](#)。Shibboleth と SAML を使用して、AWS Management Console へのシングルサインオン (SSO) アクセスを可能にする方法を示します。

## OIDC フェデレーションのサンプル

以下のサンプルアプリケーションは、Login with Amazon、Amazon Cognito、Facebook、Google などのプロバイダーで OIDC フェデレーションを使用する方法を示しています。これらのプロバイダーからの認証情報を一時的な AWS セキュリティ認証情報に交換して、AWS のサービスにアクセスできます。

- [Amazon Cognitoチュートリアル](#) – モバイル開発用の AWS SDK で Amazon Cognitoを使用することをお勧めします。Amazon Cognito は、モバイルアプリの ID を管理するための最も簡単な方法であり、同期やクロスデバイス ID のような追加機能も利用できます。Amazon Cognito の詳細については、「Amplify ドキュメント」の「[Amplify による認証](#)」を参照してください。

## カスタム ID ブローカーに対する AWS コンソールへのアクセスの許可

組織のネットワークにサインインするユーザーに対して AWS Management Console への安全なアクセスを許可するには、そのための URL を生成するコードを記述して実行できます。この URL は、AWS から取得したサインイントークンを含み、それを使って AWS に対してユーザーを認証します。結果のコンソールセッションには、フェデレーションに起因する明確な AccessKeyId が含まれる場合があります。関連する CloudTrail イベントを介したフェデレーションサインインのアクセスキーの使用状況を追跡するには、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」と「AWS Management Console サインインイベント」を参照してください。

### Note

組織で、SAML と互換性のある ID プロバイダー (IdP) を使用している場合は、コードを記述せずにコンソールにアクセスできます。これは、Microsoft の Active Directory フェデレーションサービスやオープンソースの Shibboleth などのプロバイダーで機能します。詳細については、「[SAML 2.0 フェデレーテッドユーザーが AWS Management Consoleにアクセス可能にする](#)」を参照してください。


組織のユーザーによる AWS Management Console へのアクセスを許可するには、以下の手順を実行するカスタム ID ブローカーを作成できます。

1. ユーザーがローカル ID システムによって認証されていることを確認する。
2. AWS Security Token Service (AWS STS) の [AssumeRole](#) (推奨) または [GetFederationToken](#) API オペレーションを呼び出して、ユーザーの一時的セキュリティ認証情報を取得する。ロールを引き受ける別の方法については、「[IAM ロールを使用する](#)」を参照してください。セキュリティ認



認証情報を取得するときにオプションのセッションタグを渡す方法については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

- AssumeRole\* API オペレーションのいずれかを使用してロールの一時的なセキュリティ認証情報を取得した場合、この呼び出しに DurationSeconds パラメータを含めることができます。このパラメータは、ロールセッションの期間を 900 秒 (15 分) からそのロールの最大セッション期間設定まで指定します。AssumeRole\* オペレーションで DurationSeconds を使用する場合、長期的な認証情報を持つ IAM ユーザーとして呼び出す必要があります。それ以外の場合、ステップ 3 のフェデレーションエンドポイントへの呼び出しが失敗します。ロールの最大値を確認または変更する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。
  - GetFederationToken API オペレーションを使用して認証情報を取得するには、呼び出しに DurationSeconds パラメータを含めることができます。このパラメータは、ロールセッションの継続期間を指定します。値の範囲は 900 秒 (15 分) から 129,600 秒 (36 時間) です。この API コールは、IAM ユーザーの AWS 長期的セキュリティ認証情報を使用することでのみ行うことができます。AWS アカウントのルートユーザー 認証情報を使用してこれらの呼び出しを行うこともできますが、推奨されません。ルートユーザーとしてこの呼び出しを行うと、デフォルトのセッションの継続期間は 1 時間です。900 秒 (15 分) から最長 3,600 秒 (1 時間) までセッションを指定できます。
3. AWS フェデレーションエンドポイントを呼び出し、一時的なセキュリティ認証情報を指定して、サインイントークンをリクエストする。
  4. トークンを含むコンソールの URL を生成する:
    - URL で AssumeRole\* API オペレーションのいずれかを使用する場合、SessionDuration HTTP パラメータを含めることができます。このパラメータはコンソールセッションの継続期間を 900 秒 (15 分) から 43200 秒 (12 時間) までの間で指定します。
    - URL で GetFederationToken API オペレーションを使用する場合、DurationSeconds パラメータを含めることができます。このパラメータは、フェデレーテッドコンソールセッションの継続期間を指定します。値の範囲は 900 秒 (15 分) から 129,600 秒 (36 時間) です。

 Note

- SessionDuration を使用して一時的な認証情報を取得した場合は、GetFederationToken HTTP パラメータを使用しないでください。オペレーションが失敗します。
- 1 つのロールの認証情報を使用して別のロールを引き受けることは、[ロールの連鎖](#)と呼ばれます。ロールの連鎖を使用すると、新しい認証情報は最長期間である 1 時間に

制限されます。ロールを使用して [EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与する](#) 場合、これらのアプリケーションにはこの制限が適用されません。

5. URL をユーザーに渡すか、ユーザーに代わって URL を呼び出す。

フェデレーションエンドポイントによって渡される URL はその作成後から 15 分間、有効です。これは、URL に関連付けられた一時的セキュリティ認証情報の期間 (秒) とは異なります。これらの認証情報は、認証情報の作成時から、作成時に指定した期間だけ有効です。

#### Important

URL は、関連付けられた一時的セキュリティ認証情報でアクセス許可を有効にした場合、AWS Management Console を介した AWS リソースへのアクセスを許可することを忘れないでください。そのため、この URL は機密情報として扱う必要があります。例えば、SSL 接続による 302 HTTP レスポンスステータスコードを使用して、安全なリダイレクトによって URL を返すことをお勧めします。302 HTTP レスポンスステータスコードの詳細については、「[RFC 2616 セクション 10.3.3](#)」を参照してください。

これらのタスクを完了するために、[AWS Identity and Access Management \(IAM\) と AWS Security Token Service \(AWS STS\) の HTTPS クエリ API](#) を使用できます。または、Java、Ruby、C# などのプログラミング言語を該当する [AWS SDK](#) と共に使用できます。これらの方法のそれぞれについて、以下のトピックで説明します。

#### トピック

- [IAM クエリ API オペレーションを使用したコード例](#)
- [Python を使用したコード例](#)
- [Java を使用したコード例](#)
- [URL の作成方法を示す例 \(Ruby\)](#)

#### IAM クエリ API オペレーションを使用したコード例

フェデレーションユーザーに対して AWS Management Console への直接アクセスを許可する URL を作成できます。このタスクでは、IAM と AWS STS の HTTPS クエリ API を使用します。クエリリクエストの詳細については、「[クエリリクエストを行う](#)」を参照してください。

**Note**

以下の手順は、テキスト文字列の例を含んでいます。読みやすくするために、長い例の一部では改行が追加されています。これらの文字列をご自分で使用するときは、改行をすべて削除してください。

フェデレーテッドユーザーに AWS Management Console からリソースに対するアクセスを許可するには

1. ID および認証システムでユーザーを認証します。
2. ユーザーの一時的なセキュリティ認証情報を取得します。一時的な認証情報は、アクセスキー ID、シークレットアクセスキー、およびセッショントークンで構成されています。一時的な認証情報の作成方法の詳細については、「[IAM の一時的な認証情報](#)」を参照してください。

一時的な認証情報を取得するには、AWS STS の [AssumeRole](#) API (推奨) または [GetFederationToken](#) API を呼び出します。これらの API オペレーションの違いの詳細については、AWS セキュリティブログの「[AWS アカウントへのアクセスを安全に委任する API オペレーションの理解](#)」を参照してください。

**Important**

[GetFederationToken](#) API を使用して一時的セキュリティ認証情報を作成する場合、ロールを引き受けるユーザーに認証情報を提供するアクセス許可を指定する必要があります。AssumeRole\* で始まるいずれの API オペレーションでも、IAM ロールを使用してアクセス許可を割り当てます。その他の API オペレーションでは、この方法は API によって異なります。詳細については、「[一時的なセキュリティ認証情報のアクセス権限を制御する](#)」を参照してください。さらに、AssumeRole\* API オペレーションを使用する場合、長期的な認証情報を使用する IAM ユーザーとして呼び出す必要があります。それ以外の場合は、ステップ 3 のフェデレーションエンドポイントへの呼び出しが失敗します。


3. 一時的なセキュリティ認証情報を取得した後、この情報から JSON セッション文字列を生成して、サインイントークンに置き換えられるようにします。以下の例は、認証情報のエンコード方法を示しています。プレースホルダーテキストを、先ほどの手順で取得した認証情報の該当する値に置き換えます。

```
{"sessionId": "*** temporary access key ID ***",
```

```
"sessionKey": "*** temporary secret access key ***",  
"sessionToken": "*** session token ***"}
```

4. 前の手順からのセッション文字列を [URL エンコード](#) します。エンコードする情報は機密であるため、このエンコードにウェブサービスを利用しないことをお勧めします。代わりに、開発ツールキットのローカルにインストールされた関数または機能を使用して、この情報を安全にエンコードします。Python では `urllib.quote_plus` 関数、Java では `URLEncoder.encode` 関数、Ruby では `CGI.escape` 関数を使用できます。このトピックの後の例を参照してください。

- 5.

 Note

ここで AWS は POST リクエストをサポートします。

AWS フェデレーションエンドポイントにリクエストを送信します。


`https://region-code.signin.aws.amazon.com/federation`

実行可能な *region-code* 値のリストについては、「[AWS サインインエンドポイント](#)」の [Region] (リージョン) 列を参照します。任意で、以下のデフォルトの AWS サインイン フェデレーション エンドポイントが使用できます。

`https://signin.aws.amazon.com/federation`

以下の例のように、リクエストには、Action および Session パラメータを含める必要があります。[AssumeRole\\*](#) API オペレーションを (オプションで) 使用する場合は、SessionDuration HTTP パラメータを含める必要があります。

```
Action = getSigninToken  
SessionDuration = time in seconds  
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

 Note

このステップで以下の手順は、GET リクエストを使用する場合にのみ機能します。

SessionDuration HTTP パラメータは、コンソールセッションの継続期間を指定します。これは、DurationSeconds パラメータを使用して指定する一時的な認証情報の期間とは異なり

まず、`SessionDuration` の最大値を 43200 (12 時間) に指定できます。`SessionDuration` パラメータがない場合は、セッションはステップ 2 で AWS STS から取得した認証情報の期間 (デフォルトは 1 時間) をデフォルトに設定します。`DurationSeconds` パラメータを使用した期間の指定方法の詳細については、[AssumeRole API のドキュメント](#)を参照してください。1 時間より長いコンソールセッションを作成する機能は、フェデレーションエンドポイントの `getSignInToken` オペレーションに組み込まれます。

#### Note

- `SessionDuration` を使用して一時的な認証情報を取得した場合は、`GetFederationToken` HTTP パラメータを使用しないでください。オペレーションが失敗します。
- 1 つのロールの認証情報を使用して別のロールを引き受けることは、[ロールの連鎖](#)と呼ばれます。ロールの連鎖を使用すると、新しい認証情報は最長期間である 1 時間に制限されます。ロールを使用して [EC2 インスタンスで実行されるアプリケーションにアクセス許可を付与する](#) 場合、これらのアプリケーションにはこの制限が適用されません。

長時間にわたってコンソールセッションを有効にすると、認証情報が漏洩するリスクが高くなります。このリスクを軽減するには、IAM コンソールページの [ロールの概要] で、[セッションの無効化] を選択して、どのロールのアクティブなコンソールセッションもすぐに無効にできます。詳細については、「[IAM ロールの一時的なセキュリティ認証情報の取り消し](#)」を参照してください。

以下に示しているのは、リクエストの具体的な例です。ここでは読みやすいように改行していますが、リクエストは 1 行の文字列として送信する必要があります。

```
https://signin.aws.amazon.com/federation
?Action=getSignInToken
&SessionDuration=1800
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpc8s7HYjRsgcsrsm%22%2C+%22sessionToken%22%3A+%22FQoDYXdzEBQaDLbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFFzSL0Qd3vtYHw5A5dWAj0srkdPkghomIe3mJip5%2F0djDBbo7Sm0%2FENDEiCdpsQKodTpleKA8xQq0CwFg6a69xdEBQT8FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP0012obez9lijPSdF1k2b5PfGhiuyAR9aD5%2BubM0pY86fKex1qsytjvyTbZ9nXe6DvxVDcnC0h0GETJ7XfkSFdH0v%2FYR25CUAhJ3nXIkIbG7Ucv9c0EpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

フェデレーションエンドポイントからの応答は、SignInToken 値を含む JSON ドキュメントです。実際には次のようになります。

```
{"SignInToken": "*** the SignInToken string ***"}
```

6.

**Note**

ここで AWS は POST リクエストをサポートします。

最後に、フェデレーションユーザーが AWS Management Console へのアクセスに使用できる URL を作成します。URL は、「[Step 5](#)」で使用した同じフェデレーション URL エンドポイントに以下のパラメータを追加したものです。

```
?Action = login  
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***  
&Destination = *** the form-urlencoded URL to the desired AWS console page ***  
&SignInToken = *** the value of SignInToken received in the previous step ***
```

**Note**

このステップの以下の手順は、GET API を使用する場合にのみ機能します。

以下の例は、最終的な URL がどのようになるかを示します。URL は、作成時から 15 分間、有効です。URL 内に組み込まれた一時的なセキュリティ認証情報とコンソールセッションは、認証情報の初回リクエスト時に SessionDuration HTTP パラメータで指定した期間、有効です。

```
https://signin.aws.amazon.com/federation  
?Action=login  
&Issuer=https%3A%2F%2Fexample.com  
&Destination=https%3A%2F%2Fconsole.aws.amazon.com%2F  
&SignInToken=VCQgs5qZZt3Q6fn8Tr5EXAMPLEmLnwB7JjUc-SHwnUUwabcRdnWsi4DBn-dvC  
CZ85wrD0nmlDucZEXAMPLE-vXYH4Q__mleuF_W2BE5HYexbe9y40f-kje53SsjNNecATfjIzpw1  
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrn0JoK3dtP6I9a6hi6yPgm  
i0kPZMmNGmhsVvxetKzr8mx3pxhHbMEEXAMPLETv1pij0rok3IyCR2YVcIjqwfWv32HU2X1j471u  
3fU6u0fUComeKiqTGX974xzJ0ZbdmX_t_1LrhEXAMPLEDDIisSnyHGw2xaZZqudm4mo2uTDk9Pv
```

```
915K0ZCqIgEXAMPLEcA6tgLPykEWGUyH6BdSC6166n4M4JkXIQgac7_7821YqixsNxZ6rsrpzwf
nQoS1407R0eJCCJ684EXAMPLEZRdBNnuLbUYpz2Iw3vIN0tQg0ujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_0B12MZhuFxx55555EXAMPLEEhyETEd4Zu1KPdXHkg16T9Zk11Hz2Uy1RUTUhhUxNtSQ
nWc5xkbBoEcXqpoSIEk7yhje9Vzhd61AEXAMPLE1bWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
0LSG7RyYKeYN5VIZuk3YWQpyjP0RiT5KUrsUi-NEXAMPLExMOMdo0DBEgKQsk-iu2ozh6r8bxwC
RNhujg
```

## Python を使用したコード例

以下の例は、Python を使用して、プログラムでフェデレーションユーザーに AWS Management Console への直接アクセスを許可する URL を作成する方法を示します。ここでは、以下の 2 つの例を示します。

- GET リクエスト経由で AWS にフェデレートします
- POST リクエスト経由で AWS にフェデレートします

どちらの例でも、[AWS SDK for Python \(Boto3\)](#) や [AssumeRole](#) API を使用して、一時的なセキュリティ認証情報を取得します。

### GET リクエストの使用

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AWS #####,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,
# or in a configuration file, and will be discovered automatically by the
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
```



```
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
    assumed_role_object.get('Credentials').get('AccessKeyId')
url_credentials['sessionKey'] =
    assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
    assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
# parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
# temporary credentials
# as parameters.
request_parameters = "?Action=getSigninToken"
request_parameters += "&SessionDuration=43200"
if sys.version_info[0] < 3:
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)
request_parameters += "&Session=" +
    quote_plus_function(json_string_with_temp_credentials)
request_url = "https://signin.aws.amazon.com/federation" + request_parameters
r = requests.get(request_url)
# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create URL where users can use the sign-in token to sign in to
# the console. This URL must be used within 15 minutes after the
# sign-in token was issued.
request_parameters = "?Action=login"
request_parameters += "&Issuer=Example.org"
request_parameters += "&Destination=" + quote_plus_function("https://
console.aws.amazon.com/")
request_parameters += "&SigninToken=" + signin_token["SigninToken"]
request_url = "https://signin.aws.amazon.com/federation" + request_parameters
```



```
# Send final URL to stdout
print (request_url)
```

## POST リクエストの使用

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'
import os
from selenium import webdriver # 'pip install selenium', 'brew install chromedriver'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AAWS #####,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,

# or in a configuration file, and will be discovered automatically by the
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
if sys.version_info[0] < 3:
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)

sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleDemoSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
    assumed_role_object.get('Credentials').get('AccessKeyId')
```

```
url_credentials['sessionKey'] =
    assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
    assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
# parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
# temporary credentials
# as parameters.
request_parameters = {}
request_parameters['Action'] = 'getSigninToken'
request_parameters['SessionDuration'] = '43200'
request_parameters['Session'] = json_string_with_temp_credentials

request_url = "https://signin.aws.amazon.com/federation"
r = requests.post( request_url, data=request_parameters)

# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create a POST request where users can use the sign-in token to sign in to
# the console. The POST request must be made within 15 minutes after the
# sign-in token was issued.
request_parameters = {}
request_parameters['Action'] = 'login'
request_parameters['Issuer']='Example.org'
request_parameters['Destination'] = 'https://console.aws.amazon.com/'
request_parameters['SigninToken'] =signin_token['SigninToken']

jsrequest = ''
var form = document.createElement('form');
form.method = 'POST';
form.action = '{request_url}';
request_parameters = {request_parameters}
for (var param in request_parameters) {{
    if (request_parameters.hasOwnProperty(param)) {{
        const hiddenField = document.createElement('input');
        hiddenField.type = 'hidden';
        hiddenField.name = param;
        hiddenField.value = request_parameters[param];
        form.appendChild(hiddenField);
    }}
}}
```

```
}}  
document.body.appendChild(form);  
form.submit();  
''.format(request_url=request_url, request_parameters=request_parameters)  
  
driver = webdriver.Chrome()  
driver.execute_script(jsrequest);
```

## Java を使用したコード例

以下の例は、Java を使用して、プログラムでフェデレーションユーザーに AWS Management Console への直接アクセスを許可する URL を作成する方法を示します。以下のコード例では、[AWS SDK for Java](#) を使用しています。

```
import java.net.URLEncoder;  
import java.net.URL;  
import java.net.URLConnection;  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
// Available at http://www.json.org/java/index.html  
import org.json.JSONObject;  
import com.amazonaws.auth.AWSCredentials;  
import com.amazonaws.auth.BasicAWSCredentials;  
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;  
import com.amazonaws.services.securitytoken.model.Credentials;  
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;  
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;  
  
/* Calls to AWS STS API operations must be signed using the access key ID  
   and secret access key of an IAM user or using existing temporary  
   credentials. The credentials should not be embedded in code. For  
   this example, the code looks for the credentials in a  
   standard configuration file.  
*/  
AWSCredentials credentials =  
    new PropertiesCredentials(  
        AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));  
  
AWSSecurityTokenServiceClient stsClient =  
    new AWSSecurityTokenServiceClient(credentials);  
  
GetFederationTokenRequest getFederationTokenRequest =
```

```
new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(1800);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
// console.

String policy = "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Action\":\"sns:*\", \"
  \"Effect\":\"Allow\", \"Resource\":\"*\"}]}";

getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
    stsClient.getFederationToken(getFederationTokenRequest);

Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example https://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// AWS Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.

String issuerURL = "https://mysignin.internal.mycompany.com/";
String consoleURL = "https://console.aws.amazon.com/sns";
String signInURL = "https://signin.aws.amazon.com/federation";

// Create the sign-in token using temporary credentials,
// including the access key ID, secret access key, and session token.
String sessionJson = String.format(
    "{\"%1$s\":\"%2$s\", \"%3$s\":\"%4$s\", \"%5$s\":\"%6$s\"}",
    "sessionId", federatedCredentials.getAccessKeyId(),
    "sessionKey", federatedCredentials.getSecretAccessKey(),
    "sessionToken", federatedCredentials.getSessionToken());

// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSigninTokenURL = signInURL +
    "?Action=getSigninToken" +
    "&DurationSeconds=43200" +
    "&SessionType=json&Session=" +
    URLEncoder.encode(sessionJson, "UTF-8");
```

```
URL url = new URL(getSignInTokenURL);

// Send the request to the AWS federation endpoint to get the sign-in token
URLConnection conn = url.openConnection ();

BufferedReader bufferedReader = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
String returnContent = bufferedReader.readLine();

String signInToken = new JSONObject(returnContent).getString("SignInToken");

String signInTokenParameter = "&SignInToken=" + URLEncoder.encode(signInToken, "UTF-8");

// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the AWS console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL, "UTF-8");
String loginURL = signInURL + "?Action=login" +
    signInTokenParameter + issuerParameter + destinationParameter;
```

## URL の作成方法を示す例 (Ruby)

以下の例は、Ruby を使用して、プログラムでフェデレーションユーザーに AWS Management Console への直接アクセスを許可する URL を作成する方法を示します。このコード例では、[AWS SDK for Ruby](#) を使用しています。

```
require 'rubygems'
require 'json'
require 'open-uri'
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to AWS STS API operations must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the AWS Ruby SDK.
```

```
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work
# in the AWS SNS console.

session = sts.get_federation_token({
  duration_seconds: 1800,
  name: "UserName",
  policy: "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\"Action\":
\"sns:*\",\"Resource\":\"*\"}}",
})

# The issuer value is the URL where users are directed (such as
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the AWS STS federation endpoint.
issuer_url = "https://mysignin.internal.mycompany.com/"
console_url = "https://console.aws.amazon.com/sns"
signin_url = "https://signin.aws.amazon.com/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
  :sessionId => session.credentials[:access_key_id],
  :sessionKey => session.credentials[:secret_access_key],
  :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
# created earlier and the session information as a JSON block.
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
  "?Action=getSignInToken" +
  "&SessionType=json&Session=" +
  CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read
```

```
# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SignInToken']
signin_token_param = "&SignInToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
  issuer_param + destination_param
```

## 一時的なセキュリティ認証情報のための追加リソース

以下のシナリオやアプリケーションは、一時的なセキュリティ認証情報の使用時に役立ちます。

- [AWS STS SourceIdentity をアイデンティティプロバイダーと統合する方法](#)。この記事では、Okta、Ping、OneLogin を IdP として使用し、AWS STS SourceIdentity 属性を設定する方法について説明します。
- [OIDC フェデレーション](#)。このセクションでは、OIDC フェデレーションと AssumeRoleWithWebIdentity API を使用するとき IAM ロールを設定する方法について説明します。
- [MFA 保護 API アクセスの設定](#)。このトピックでは、アカウントで機密性の高い API アクションを保護するために、ロールを使用して多要素認証 (MFA) を要求する方法について説明します。

AWS におけるポリシーとアクセス権限の詳細については、以下のトピックを参照してください。

- [AWS リソースのアクセス管理](#)
- [ポリシーの評価論理](#)
- [Amazon Simple Storage Service ユーザーガイド](#)の「Amazon S3 リソースへのアクセス許可の管理」。
- 信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

## IAM リソースのタグ付け

タグは、ユーザーが AWS リソースに割り当てることのできるカスタム属性ラベルです。各 タグは 2 つの部分で構成されます:

- タグキー (例: CostCenter、Environment、Project、Purpose)。
- タグ値として知られるオプションのフィールド (例、111122223333、Production、チーム名など)。タグ値を省略すると、空の文字列を使用した場合と同じになります。

これらは共にキーバリューのペアと呼ばれます。IAM リソースで使用できるタグの数の制限については、「[IAM と AWS STS クォータ](#)」を参照してください。

### Note

タグキーとタグキー値での大文字と小文字の区別の詳細については、「[Case sensitivity](#)」を参照してください。

タグを使用すると、AWS リソースの特定と整理に役立ちます。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てて、リソースの関連を示すことができます。例えば、Amazon S3 バケットに割り当てたタグと同じタグを IAM ロールに割り当てることができます。タグ付け戦略の詳細については、ユーザーガイドの「[AWS リソースのタグ付け](#)」を参照してください。

IAM リソースの特定、整理、追跡に加え、IAM ポリシーのタグを使って、リソースを表示および操作できるユーザーを制御することもできます。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

## AWS タグ命名規則を選択する

IAM リソースにタグをアタッチする際は、タグの命名規則を慎重に選択します。すべての AWS タグに同じ規則を適用します。ポリシーでタグを使用して AWS リソースへのアクセスを制御する場合、



これは特に重要です。AWS のタグをすでに使用している場合は、命名規則を確認し、必要に応じて調整します。

#### Note

アカウントが AWS Organizations のメンバーである場合、「Organizations ユーザーガイド」の「[Tag policies](#)」(タグポリシー)を参照し、Organizations でタグを使用する方法についてご確認ください。

## タグの命名に関するベストプラクティス

ここでは、タグの命名規則に関するベストプラクティスについて説明します。

タグ名は一貫性を保って使用してください。たとえば、タグ CostCenter とタグ costcenter は別のものです。一方は財務分析とレポート用のコスト配分タグとして設定され、もう一方はそうではないかもしれません。同様に、Name タグは多くのリソース用の AWS コンソールで目にしますが、name タグはそうではありません。タグキーとタグキー値での大文字と小文字の区別の詳細については、「[Case sensitivity](#)」を参照してください。

いくつかのタグは AWS により事前に定義されています。また、さまざまな AWS のサービスによって自動的に作成されます。多くの場合、AWS で定義されるタグの名前はすべて小文字で、名前に含まれる単語はハイフンで区切られ、タグのソースサービスを識別するプレフィックスが付きます。

例:

- `aws:ec2spot:fleet-request-id` のタグは、インスタンスを起動した Amazon EC2 スポットインスタンスリクエストを識別します。
- `aws:cloudformation:stack-name` のタグは、リソースを作成した AWS CloudFormation スタックを識別します。
- `elasticbeanstalk:environment-name` のタグは、リソースを作成したアプリケーションを識別します。

タグの名前を付ける際は、すべて小文字を使用し、単語はハイフンで区切り、組織名や略称を識別するプレフィックスを付けることを検討してください。例えば、AnyCompany という名前の架空の会社の場合では、次のようにタグを定義できます。

- `anycompany:cost-center` のタグは、内部のコストセンターのコードを識別するのに使用

- `anycompany:environment-type` のタグは、開発、テスト、本番のいずれの環境であるかを識別するのに使用
- `anycompany:application-id` のタグは、リソースが作成されたアプリケーションを識別するのに使用

プレフィックスを付けることで、自分の組織が定義したタグだということが明確に識別でき、AWS または使用中のサードパーティーのツールにより定義されたタグではないことがわかります。すべて小文字を使用し、単語をハイフンで区切ることにより、タグ名に大文字を使用した場合の混乱を避けることができます。例えば、`anycompany:project-id` の方が、`ANYCOMPANY:ProjectID`、`anycompany:projectID`、`Anycompany:ProjectId` よりも覚えるのが簡単です。

## IAM および AWS STS でのタグ付けの規則

IAM および AWS STS でのタグの作成と適用を管理する多数の規則。

### 命名タグ

IAM リソース、AWS STS `assume-role` セッション、AWS STS フェデレーションユーザーセッションのタグ命名規則を作成するときは、次の規則に従います。

文字の要件 – タグキーバリューには、文字、数字、空白、記号 (`_ . : / = + - @`) の任意の組み合わせを使用できます。

大文字と小文字の区別 – タグキーの大文字と小文字の区別は、タグ付けされた IAM リソースの種類によって変わります。IAM ユーザーとロールのタグキーバリューのペアでは、大文字と小文字は区別されませんが、大文字と小文字は維持されます。つまり、**Department** と **department** のタグキーを別々に持つことはできません。**Department=finance** タグでユーザーをタグ付けし、**department=hr** タグを追加すると、最初のタグが置き換えられます。2 番目のタグは追加されません。

他の IAM リソースタイプでは、タグキーバリューでは大文字と小文字が区別されます。つまり、**Costcenter** と **costcenter** タグキーとを個別に使用できます。例えば、カスタマー管理ポリシーに **Costcenter = 1234** タグを付け、**costcenter = 5678** タグを追加すると、そのポリシーには **Costcenter** と **costcenter** タグキーの両方が表示されます。

ベストプラクティスとしては、大文字と小文字の扱いに一貫性がない場合は、同様のタグを使用しないことを推奨します。タグに大文字を使用する場合の戦略を決定し、その戦略をすべてのリソースタ

イプにわたって一貫して実装することを推奨します。タグ付けのベストプラクティスの詳細については、「AWS 全般のリファレンス」の「[AWS リソースのタグ付け](#)」を参照してください。

IAM リソースにアタッチされているタグキーの、大文字と小文字の区別の差異を次に示します。

タグキーバリューでは大文字と小文字が区別されません。

- IAM ロール
- IAM ユーザー

タグキーバリューでは大文字と小文字が区別されます。

- カスタマー管理ポリシー
- インスタンスプロファイル
- OpenID Connect ID プロバイダー
- SAML ID プロバイダー
- サーバー証明書
- 仮想 MFA デバイス

加えて、次のルールが適用されます。

- テキスト **aws:** から開始するタグキーや値を作成することはできません。このタグプレフィックスは AWS 社内使用のために予約されています。
- 空の値 (例: **phoneNumber =** ) を含むタグを作成することができます。空のタグキーを作成することはできません。
- 1 つのタグで複数の値を指定することはできませんが、カスタムの複数值構造を 1 つの値で作成することができます。たとえば、ユーザー Zhang がエンジニアリングチーム、QA チームで働いているとします。**team = Engineering** タグ、**team = QA** タグの順にアタッチする場合は、タグの値を **Engineering** から **QA** に変更します。代わりに、カスタム区切り文字を使用して 1 つのタグに複数の値を含めることができます。この例では、**team = Engineering:QA** タグを Zhang にアタッチします。

#### Note

**team** タグを使用して、この例のエンジニアへのアクセスを制御するには、**Engineering** を含む可能性のある各設定を許可するポリシー (例: **Engineering:QA**) を作成する必要があります。

あります。ポリシーのタグの使用の詳細については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

## タグの適用と編集

タグを IAM リソースにアタッチするときは、次の規則に従います。

- ほとんどの IAM リソースにはタグを付けられますが、グループ、引き受け済みのロール、アクセスレポート、ハードウェアベース MFA デバイスにはタグを付けることができません。
- タグエディタを使って IAM リソースにタグ付けすることはできません。タグエディタは IAM タグをサポートしていません。他のサービスでのタグエディタの使用の詳細については、「AWS Resource Groups ユーザーガイド」の「[タグエディタでの作業](#)」を参照してください。
- IAM リソースにタグ付けするには、特定のアクセス許可が必要です。リソースにタグを付ける、またはタグを解除するには、タグを一覧表示するアクセス許可も必要です。詳細については、このページの最後にある各 IAM リソースのトピックのリストを参照してください。
- AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。
- 同じタグを複数の IAM リソースに適用することができます。たとえば、AWS\_Development という名前の部門に 12 人のメンバーがあるとします。12 人のユーザーと、**department** のタグキー、**awsDevelopment** の値を持つロールを持つことができます (**department = awsDevelopment**)。また、[タグ付けをサポートする他のサービス](#)のリソースで同じタグを使用することもできます。
- IAM エンティティ (ユーザーまたはロール) は、タグキーが同じである複数のインスタンスを持つことはできません。たとえば、タグのキーバリューのペア **costCenter = 1234** を含むユーザーがいる場合は、タグキーバリューのペア **costCenter = 5678** をアタッチできます。IAM は、**costCenter** タグの値を **5678** に更新します。
- IAM エンティティ (ユーザーまたはロール) にアタッチされているタグを編集するには、新しい値のタグをアタッチして、既存のタグを上書きします。例えば、タグのキーバリューのペア **department = Engineering** を持つユーザーがいるとします。そのユーザーを QA 部門に移動させる必要がある場合、**department = QA** タグのキーバリューのペアをこのユーザーにアタッチします。その結果、**department** タグキーの **Engineering** 値は、**QA** 値に置き換わります。

## トピック

- [IAM ユーザーのタグ付け](#)
- [IAM ロールのタグ付け](#)
- [カスタマー管理ポリシーのタグ付け](#)
- [IAM ID プロバイダーのタグ付け](#)
- [Amazon EC2 ロール用のインスタンスプロファイルのタグ付け](#)
- [サーバー証明書のタグ付け](#)
- [仮想 MFA デバイスのタグ付け](#)
- [AWS STS でのセッションタグの受け渡し](#)

## IAM ユーザーのタグ付け

IAM タグのキーバリューのペアを使用することで、ユーザーにカスタム属性を追加できます。たとえば、位置情報をユーザーに追加するには、タグキー **location** とタグ値 **us\_wa\_seattle** を追加できます。または、3 つの位置のタグのキーバリューのペアを使用できます (**loc-country = us**、**loc-state = wa**、**loc-city = seattle**)。タグを使用することにより、リソースへのユーザーアクセスや、ユーザーにアタッチできるタグを、制御できます。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

### IAM ユーザーのタグ付けに必要なアクセス許可

IAM ユーザーに、他のユーザーへのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListUserTags
- iam:TagUser
- iam:UntagUser

IAM ユーザーに、特定のユーザーへのタグの追加、一覧表示、または削除を許可するには

タグを管理する必要がある IAM ユーザーのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、`<username>` を、タグの管理が必要とされているユーザーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM ユーザーにタグの自己管理を許可するには

独自のタグの管理をユーザーに許可するアクセス許可ポリシーに、以下のステートメントを追加します。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::user/${aws:username}"
}
```

IAM ユーザーに、特定のユーザーへのタグの追加を許可するには

特定のユーザーのタグを追加する必要があるが、削除やタグ付けは不要な IAM ユーザーのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

`iam:TagUser` アクションには、`iam:ListUserTags` アクションも含める必要があります。

このポリシーを使用するには、`<username>` を、タグの管理が必要とされているユーザーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## IAM ユーザーのタグの管理 (コンソール)

IAM ユーザーのタグを AWS Management Console から管理できます。

ユーザーのタグを管理するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで、[Users (ユーザー)] を選択し、編集するユーザーの名前を選択します。
3. [タグ] タブを選択し、以下のいずれかのアクションを完了します。
  - ユーザーがまだタグがない場合は、[新しいタグを追加] を選択します。
  - [Manage tags] (タグを管理) を選択して、既存のタグセットを管理します。
4. タグのセットを完了するには、タグを追加または削除します。次に、[Save changes] (変更の保存) を選択します。

## IAM ユーザーのタグの管理 (AWS CLI または AWS API)

IAM ユーザーのタグを一覧表示、アタッチ、または削除できます。IAM ユーザーのタグを管理するには、AWS CLI または AWS API を使用します。



IAM ユーザーに現在アタッチされているタグを一覧表示するには (AWS または AWS CLI API)

- AWS CLI: [aws iam list-user-tags](#)
- AWS API: [ListUserTags](#)

タグを IAM ユーザーにアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-user](#)
- AWS API: [TagUser](#)

IAM ユーザーからタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-user](#)
- AWS API: [UntagUser](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## IAM ロールのタグ付け

IAM タグのキーバリューのペアを使用することで、IAM ロールにカスタム属性を追加できます。例えば、位置情報をロールに追加するには、タグキー **location** とタグ値 **us\_wa\_seattle** を追加します。または、3 つの位置のタグのキーバリューのペアを使用できます (**loc-country = us**、**loc-state = wa**、**loc-city = seattle**)。タグを使用することにより、リソースへのロールのアクセスや、ロールにアタッチできるタグを、制御できます。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。



## IAM ロールのタグ付けに必要なアクセス許可

IAM ロールに、他のエンティティ (ユーザーまたはロール) へのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListRoleTags
- iam:TagRole
- iam:UntagRole
- iam:ListUserTags
- iam:TagUser
- iam:UntagUser

IAM ロールに、特定のユーザーへのタグの追加、一覧表示、または削除を許可するには

タグを管理する必要がある IAM ロールのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、`<username>` を、タグの管理が必要とされているユーザーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM ロールに、特定のユーザーへのタグの追加を許可するには

特定のユーザーのタグを追加する必要があるが、削除やタグ付けは不要な IAM ロールのアクセス許可ポリシーに、以下のステートメントを追加します。

このポリシーを使用するには、`<username>` を、タグの管理が必要とされているユーザーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM ロールに、特定のロールへのタグの追加、一覧表示、または削除を許可するには

タグを管理する必要がある IAM ロールのアクセス許可ポリシーに、以下のステートメントを追加します。`<rolename>` を、タグの管理が必要とされているロールの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListRoleTags",
    "iam:TagRole",
    "iam:UntagRole"
  ],
  "Resource": "arn:aws:iam::<account-number>:role/<rolename>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## IAM ロールのタグの管理 (コンソール)

IAM ロールのタグを AWS Management Console から管理できます。

ロールのタグを管理するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで、ロール を選択し、編集するロールの名前を選択します。
3. [タグ] タブを選択し、以下のいずれかのアクションを完了します。

- ロールにまだタグがない場合は、[Add new tag] (新しいタグを追加) を選択します。
  - [Manage tags] (タグを管理) を選択して、既存のタグセットを管理します。
4. タグのセットを完了するには、タグを追加または削除します。次に、[変更の保存] を選択します。

## IAM ロールのタグの管理 (AWS CLI または AWS API)

IAM ロールのタグを一覧表示、アタッチ、または削除できます。IAM ロールのタグ付けを管理するには、AWS CLI または AWS API を使用します。

IAM ロール (AWS CLI または AWS API) に現在アタッチされているタグを一覧表示するには

- AWS CLI: [aws iam list-role-tags](#)
- AWS API: [ListRoleTags](#)

タグを IAM ロール (AWS CLI または AWS API) にアタッチするには

- AWS CLI: [aws iam tag-role](#)
- AWS API: [TagRole](#)

IAM ロール (AWS CLI または AWS API) からタグを削除するには

- AWS CLI: [aws iam untag-role](#)
- AWS API: [UntagRole](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## カスタマー管理ポリシーのタグ付け

IAM タグのキーバリューのペアを使用することにより、カスタム属性をカスタマー管理ポリシーに追加できます。例えば、部門情報を使用してポリシーにタグを付けるには、タグキーと

**Department** とタグ値 **eng** を追加します。あるいは、ポリシーにタグを付けて、特定の環境 (**Environment = lab** など) 向けであることを示すこともできます。リソースへのアクセスを制御したり、どのタグをリソースにアタッチできるかを制御したりするために、タグを使用します。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

## カスタマー管理ポリシーのタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) に、カスタマー管理ポリシーへのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListPolicyTags
- iam:TagPolicy
- iam:UntagPolicy


IAM エンティティ (ユーザーまたはロール) にカスタマー管理ポリシーへのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、`<policyname>` を、タグの管理が必要とされているポリシーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListPolicyTags",
    "iam:TagPolicy",
    "iam:UntagPolicy"
  ],
  "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

IAM エンティティ (ユーザーまたはロール) に特定のカスタマー管理ポリシーへのタグの追加を許可するには

特定のポリシーのタグを追加する必要があるが、削除は不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

 Note

iam:TagPolicy アクションには、iam:ListPolicyTags アクションも含める必要があります。

このポリシーを使用するには、`<policyname>` を、タグの管理が必要とされているポリシーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListPolicyTags",
    "iam:TagPolicy"
  ],
  "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## IAM カスタマー管理ポリシーのタグの管理 (コンソール)

IAM カスタマー管理ポリシーのタグは、AWS Management Console から管理できます。

カスタマー管理ポリシーのタグを管理するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで、[Policies (ポリシー)] を選択し、続いて編集するカスタマー管理ポリシーの名前を選択します。
3. [タグ] タブを選択し、[タグを管理] を選択します。

4. タグのセットを完了するには、タグを追加または削除します。次に、[Save changes] (変更の保存) を選択します。

## IAM カスタマー管理ポリシーのタグの管理 (AWS CLI または AWS API)

IAM カスタマー管理ポリシーのタグを一覧表示、アタッチ、または削除することができます。IAM カスタマー管理ポリシーのタグを管理するには、AWS CLI または AWS API を使用します。

IAM カスタマー管理ポリシーに現在アタッチされているタグを一覧表示するには (AWS CLI または AWS API)

- AWS CLI: [aws iam list-policy-tags](#)
- AWS API: [ListPolicyTags](#)

IAM カスタマー管理ポリシーにタグをアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-policy](#)
- AWS API: [TagPolicy](#)

IAM カスタマー管理ポリシーからタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-policy](#)
- AWS API: [UntagPolicy](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## IAM ID プロバイダーのタグ付け

IAM タグのキーバリューのペアを使用することで、ID プロバイダー (IdP) にカスタム属性を追加できます。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

IAM での IdP のタグ付けについては、次のトピックを参照してください。

## トピック

- [OpenID Connect \(OIDC\) ID プロバイダーのタグ付け](#)
- [IAM SAML ID プロバイダーのタグ付け](#)

## OpenID Connect (OIDC) ID プロバイダーのタグ付け

IAM タグキーバリューを使用することで、IAM OpenID Connect (OIDC) ID プロバイダーにカスタム属性を追加できます。例えば、OIDC ID プロバイダーを特定するには、タグキー **google** とタグ値 **oidc** を追加します。リソースへのアクセスを制御したり、どのタグをオブジェクトにアタッチできるかを制御したりするために、タグを使用します。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

### IAM OIDC ID プロバイダーのタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) に IAM OIDC ID プロバイダーへのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListOpenIDConnectProviderTags
- iam:TagOpenIDConnectProvider
- iam:UntagOpenIDConnectProvider

IAM エンティティ (ユーザーまたはロール) に IAM OIDC ID プロバイダーへのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、**<OIDCProviderName>** を、タグの管理が必要とされている OIDC プロバイダーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
```

```
    "iam:ListOpenIDConnectProviderTags",
    "iam:TagOpenIDConnectProvider",
    "iam:UntagOpenIDConnectProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

IAM エンティティ (ユーザーまたはロール) に、特定の IAM OIDC ID プロバイダーへのタグの追加を許可するには

特定の ID プロバイダーのタグを追加する必要があるが、削除やタグ付けは不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

`iam:TagOpenIDConnectProvider` アクションには、`iam:ListOpenIDConnectProviderTags` アクションも含める必要があります。

このポリシーを使用するには、`<OIDCProviderName>` を、タグの管理が必要とされている OIDC プロバイダーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListOpenIDConnectProviderTags",
    "iam:TagOpenIDConnectProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## IAM OIDC ID プロバイダーのタグの管理 (コンソール)

IAM OIDC ID プロバイダーのタグは AWS Management Console から管理できます。



## OIDC ID プロバイダーのタグを管理するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. コンソールのナビゲーションペインで [Identity providers (ID プロバイダー)] を選択し、編集する ID プロバイダーの名前を選択します。
3. [Tags (タグ)] セクションで、[Manage tags (タグの管理)] を選択し、次のいずれかのアクションを実行します。
  - OIDC ID プロバイダーにまだタグがない場合、または新しいタグを追加する場合は、[Add tag (タグの追加)] を選択します。
  - 既存のタグキーバリューを編集します。
  - タグを削除するには、[Remove tag (タグの削除)] を選択します。
4. 次に、変更の保存を選択します。

## IAM OIDC ID プロバイダーのタグの管理 (AWS CLI または AWS API)

IAM OIDC ID プロバイダーのタグを一覧表示、アタッチ、または削除することができます。IAM OIDC ID プロバイダーのタグを管理するには、AWS CLI または AWS API を使用します。

IAM OIDC ID プロバイダーに現在アタッチされているタグを一覧表示するには (AWS CLI または AWS API)

- AWS CLI: [aws iam list-open-id-connect-provider-tags](#)
- AWS API: [ListOpenIDConnectProviderTags](#)

IAM OIDC ID プロバイダーにタグをアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-open-id-connect-provider](#)
- AWS API: [TagOpenIDConnectProvider](#)

IAM OIDC ID プロバイダーのタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-open-id-connect-provider](#)
- AWS API: [UntagOpenIDConnectProvider](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## IAM SAML ID プロバイダーのタグ付け

IAM タグのキーバリューのペアを使用することで、SAML プロバイダーにカスタム属性を追加できます。例えば、プロバイダーを特定するには、タグキー `okta` とタグ値 `saml` を追加します。リソースへのアクセスを制御したり、どのタグをオブジェクトにアタッチできるかを制御したりするために、タグを使用します。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

### SAML ID プロバイダーのタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) に SAML 2.0 ベースの ID プロバイダー (IdP) へのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- `iam:ListSAMLProviderTags`
- `iam:TagSAMLProvider`
- `iam:UntagSAMLProvider`

IAM エンティティ (ユーザーまたはロール) に SAML ID プロバイダーへのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、`<SAMLProviderName>` を、タグの管理が必要とされている SAML プロバイダーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called "JSON エディターを使用したポリシーの作成"](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListSAMLProviderTags",
    "iam:TagSAMLProvider",
    "iam:UntagSAMLProvider"
  ]
}
```

```
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
  }
```

IAM エンティティ (ユーザーまたはロール) に、特定の SAML ID プロバイダーへのタグの追加を許可するには

特定の SAML プロバイダーのタグを追加する必要はあるが、削除やタグ付けは不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

iam:TagSAMLProvider アクションには、iam:ListSAMLProviderTags アクションも含める必要があります。

このポリシーを使用するには、<SAMLProviderName> を、タグの管理が必要とされている SAML プロバイダーの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListSAMLProviderTags",
    "iam:TagSAMLProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

### IAM SAML ID プロバイダーのタグの管理 (コンソール)

IAM SAML ID プロバイダーのタグは AWS Management Console から管理できます。

### SAML ID プロバイダーのタグを管理するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. コンソールのナビゲーションペインで [Identity providers (ID プロバイダー)] を選択し、編集する SAML ID プロバイダーの名前を選択します。
3. [Tags (タグ)] セクションで、[Manage tags (タグの管理)] を選択し、次のいずれかのアクションを実行します。
  - SAML ID プロバイダーにまだタグがない場合、または新しいタグを追加する場合は、[Add tag (タグの追加)] を選択します。
  - 既存のタグキーバリューを編集します。
  - タグを削除するには、[Remove tag (タグの削除)] を選択します。
4. タグのセットを完了するには、タグを追加または削除します。次に、変更の保存を選択します。

#### IAM SAML ID プロバイダーのタグの管理 (AWS CLI または AWS API)

IAM SAML ID プロバイダーのタグを一覧表示、アタッチ、または削除することができます。IAM SAML ID プロバイダーのタグを管理するには、AWS CLI または AWS API を使用します。

SAML ID プロバイダーに現在アタッチされているタグを一覧表示するには (AWS CLI または AWS API)

- AWS CLI: [aws iam list-saml-provider-tags](#)
- AWS API: [ListSAMLProviderTags](#)

SAML ID プロバイダーにタグをアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-saml-provider](#)
- AWS API: [TagSAMLProvider](#)

SAML ID プロバイダーのタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-saml-provider](#)
- AWS API: [UntagSAMLProvider](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## Amazon EC2 ロール用のインスタンスプロファイルのタグ付け

Amazon EC2 インスタンスを起動するときに、そのインスタンスに関連付ける IAM ロールを指定します。インスタンスプロファイルは IAM ロールのコンテナであり、インスタンスの起動時に Amazon EC2 インスタンスにロール情報を渡すために使用できます。インスタンスプロファイルにタグを付けることができるのは、AWS CLI または AWS API を使用しているときです。

IAM タグのキーバリューのペアを使用することで、インスタンスプロファイルにカスタム属性を追加できます。例えば、インスタンスプロファイルに部門情報を追加するには、タグキー **access-team** とタグ値 **eng** を追加します。これにより、一致するタグを持つプリンシパルは、同じタグを持つインスタンスプロファイルにアクセスできるようになります。複数のタグのキーバリューのペアを使用して、チームとプロジェクト、**access-team = eng** と **project = peg** を指定できます。タグを使用することにより、リソースへのユーザーアクセスや、ユーザーにアタッチできるタグを、制御できます。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

### インスタンスプロファイルのタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) にインスタンスプロファイルへのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListInstanceProfileTags
- iam:TagInstanceProfile
- iam:UntagInstanceProfile

IAM エンティティ (ユーザーまたはロール) にインスタンスプロファイルへのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、**<InstanceProfileName>** を、タグの管理が必要とされているインスタンスプロファイルの名前に置き換えます。この例の JSON ポリシードキュメントを使

用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListInstanceProfileTags",
    "iam:TagInstanceProfile",
    "iam:UntagInstanceProfile"
  ],
  "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

IAM エンティティ (ユーザーまたはロール) にインスタンスプロファイルへのタグの追加を許可するには

特定のインスタンスプロファイルのタグを追加する必要があるが、削除は不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

iam:TagInstanceProfile アクションには、iam:ListInstanceProfileTags アクションも含める必要があります。

このポリシーを使用するには、*<InstanceProfileName>* を、タグの管理が必要とされているインスタンスプロファイルの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListInstanceProfileTags",
    "iam:TagInstanceProfile"
  ],
  "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## インスタンスプロファイルのタグの管理 (AWS CLI または AWS API)

インスタンスプロファイルのタグを一覧表示、アタッチ、または削除することができます。AWS CLI または AWS API を使用して、インスタンスプロファイルのタグ付けを管理できます。

インスタンスプロファイルに現在アタッチされているタグを一覧表示するには (AWS CLI または AWS API)

- AWS CLI: [aws iam list-instance-profile-tags](#)
- AWS API: [ListInstanceProfileTags](#)

インスタンスプロファイルにタグをアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-instance-profile](#)
- AWS API: [TagInstanceProfile](#)

インスタンスプロファイルからタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-instance-profile](#)
- AWS API: [UntagInstanceProfile](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## サーバー証明書のタグ付け

AWS CLI を使用して SSL/TLS 証明書を管理する場合は、または AWS API を使用して IAM あるサーバー証明書にタグ付けします。AWS Certificate Manager (ACM) でサポートされているリージョンの証明書については、IAM ではなく ACM を使用して、サーバー証明書をプロビジョン、管理、デプロイすることをお勧めします。サポートされていないリージョンでは、IAM を Certificate Manager として使用する必要があります。ACM がサポートするリージョンについては、「AWS 全般のリファレンス」の「[AWS Certificate Manager エンドポイントとクォータ](#)」を参照してください。

IAM タグのキーバリューのペアを使用することで、サーバー証明書にカスタム属性を追加できます。例えば、サーバー証明書の所有者または管理者に関する情報を追加するには、タグキー **owner** とタグ値 **net-eng** を追加します。または、タグキー **CostCenter** とタグ値 **1234** を追加して、コストセンターを指定することもできます。リソースへのアクセスを制御したり、どのタグをリソースにアタッチできるかを制御したりするために、タグを使用します。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

## サーバー証明書のタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) にサーバー証明書へのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListServerCertificateTags
- iam:TagServerCertificate
- iam:UntagServerCertificate

IAM エンティティ (ユーザーまたはロール) にサーバー証明書へのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、**<CertificateName>** を、タグの管理が必要とされているサーバー証明書の名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListServerCertificateTags",
    "iam:TagServerCertificate",
    "iam:UntagServerCertificate"
  ],
  "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```



```
}
```

IAM エンティティ (ユーザーまたはロール) に特定のサーバー証明書へのタグの追加を許可するには

特定の サーバー証明書のタグを追加する必要はあるが、削除やタグ付けは不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

iam:TagServerCertificate アクションには、iam:ListServerCertificateTags アクションも含める必要があります。

このポリシーを使用するには、*<CertificateName>* を、タグの管理が必要とされているサーバー証明書の名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListServerCertificateTags",
    "iam:TagServerCertificate"
  ],
  "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## サーバー証明書のタグの管理 (AWS CLI または AWS API)

サーバー証明書のタグを一覧表示、アタッチ、または削除できます。サーバー証明書のタグを管理するには、AWS CLI または AWS API を使用します。

サーバー証明書に現在アタッチされているタグを一覧表示するには (AWS CLI または AWS API)

- AWS CLI: [aws iam list-server-certificate-tags](#)
- AWS API: [ListServerCertificateTags](#)

サーバー証明書にタグをアタッチするには (AWS CLI または AWS API)

- AWS CLI: [aws iam tag-server-certificate](#)
- AWS API: [TagServerCertificate](#)

サーバー証明書からタグを削除するには (AWS CLI または AWS API)

- AWS CLI: [aws iam untag-server-certificate](#)
- AWS API: [UntagServerCertificate](#)

他の AWS サービスのリソースにタグをアタッチする方法については、これらのサービスのドキュメントを参照してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## 仮想 MFA デバイスのタグ付け

IAM タグのキーバリューのペアを使用することで、仮想 MFA デバイスにカスタム属性を追加できます。例えば、ユーザーの仮想 MFA デバイスのコストセンター情報を追加するには、タグキー **CostCenter** とタグ値 **1234** を追加します。リソースへのアクセスを制御したり、どのタグをオブジェクトにアタッチできるかを制御したりするために、タグを使用します。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

AWS STS のタグを使用して、ロールを引き受けるとき、またはユーザーをフェデレートするときにカスタム属性を追加することもできます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

## 仮想 MFA デバイスのタグ付けに必要なアクセス許可

IAM エンティティ (ユーザーまたはロール) に仮想 MFA デバイスへのタグ付けを許可するには、アクセス許可を設定する必要があります。IAM ポリシーの次の IAM タグアクションのいずれかまたはすべてを指定することができます。

- iam:ListMFADeviceTags
- iam:TagMFADevice
- iam:UntagMFADevice

IAM エンティティ (ユーザーまたはロール) に仮想 MFA デバイスへのタグの追加、一覧表示、削除を許可するには

タグを管理する必要がある IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。アカウント番号を使用し、`<MFATokenID>` を、タグの管理が必要とされている MFA デバイスの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListMFADeviceTags",
    "iam:TagMFADevice",
    "iam:UntagMFADevice"
  ],
  "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

IAM エンティティ (ユーザーまたはロール) に特定の仮想 MFA デバイスへのタグの追加を許可するには

特定のポリシーの MFA デバイスを追加する必要はあるが、削除は不要である IAM エンティティのアクセス許可ポリシーに、以下のステートメントを追加します。

#### Note

`iam:TagMFADevice` アクションには、`iam:ListMFADeviceTags` アクションも含める必要があります。

このポリシーを使用するには、`<MFATokenID>` を、タグの管理が必要とされている MFA デバイスの名前に置き換えます。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListMFADeviceTags",
```

```
    "iam:TagMFADevice"  
  ],  
  "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"  
}
```

または、[IAMFullAccess](#) などの AWS 管理ポリシーを使用して、IAM へのフルアクセスを付与することもできます。

## 仮想 MFA デバイスのタグの管理 (AWS CLI または AWS API)

仮想 MFA デバイスのタグを一覧表示、アタッチ、または削除できます。仮想 MFA デバイスのタグを管理するには、AWS CLI または AWS API を使用します。

仮想 MFA デバイ스에 현재 아태치되어 있는 태그를 일러 표시するには (AWS CLI 또는 AWS API)

- AWS CLI: [aws iam list-mfa-device-tags](#)
- AWS API: [ListMFADeviceTags](#)

仮想 MFA デ바이스에 태그를 아태치するには (AWS CLI 또는 AWS API)

- AWS CLI: [aws iam tag-mfa-device](#)
- AWS API: [TagMFADevice](#)

仮想 MFA 데바이스에서 태그를 제거するには (AWS CLI 또는 AWS API)

- AWS CLI: [aws iam untag-mfa-device](#)
- AWS API: [UntagMFADevice](#)

他の AWS 서비스의 리소스에 태그를 아태치하는 방법については、これらの 서비스의 문서를 참조してください。

IAM を使用して、タグで詳細なアクセスを許可する設定については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## AWS STS でのセッションタグの受け渡し

セッションタグは、AWS STS で IAM ロールを引き受けるとき、またはユーザーをフェデレートするときに渡すキーバリューのペアの属性です。これを行うには、AWS CLI または ID プロバイダー (IdP) を介して AWS STS または AWS API リクエストを実行します。AWS STS を使用して一時的な

セキュリティ認証情報をリクエストすると、セッションが生成されます。セッションは有効期限があり、アクセスキーペアやセッショントークンなどの [認証情報](#) があります。セッション認証情報を使用して後続のリクエストを行う場合、[リクエストコンテキスト](#)には、[aws:PrincipalTag](#) コンテキストキーが含まれます。ポリシーの Condition 要素で [aws:PrincipalTag](#) キーを使用して、それらのタグに基づいてアクセスを許可または拒否できます。

一時的な認証情報を使用してリクエストを行う場合、プリンシパルに一連のタグが含まれる場合があります。これらのタグは、次のソースから取得されます。

1. セッションタグ – これらのタグは、ロールを引き受けるか、AWS CLI または AWS API を使用してユーザーをフェデレーションしたときに渡されました。これらのオペレーションの詳細については、[セッションのタグ付けオペレーション](#) を参照してください。
2. 受信推移的セッションタグ – これらのタグは、ロール連鎖内の前のセッションから継承されました。詳細については、このトピックで後述する「[セッションタグを使用したロールの連鎖](#)」を参照してください。
3. IAM タグ – IAM が引き受けたロールにアタッチされたタグ。

## トピック

- [セッションのタグ付けオペレーション](#)
- [セッションタグについて知っておくべきこと](#)
- [セッションタグの追加に必要なアクセス許可](#)
- [AssumeRole を使用したセッションタグの受け渡し](#)
- [AssumeRoleWithSAML を使用したセッションタグの受け渡し](#)
- [AssumeRoleWithWebIdentity を使用したセッションタグの受け渡し](#)
- [GetFederationToken を使用したセッションタグの受け渡し](#)
- [セッションタグを使用したロールの連鎖](#)
- [ABAC でのセッションタグの使用](#)
- [CloudTrail でのセッションタグの表示](#)

## セッションのタグ付けオペレーション

セッションタグを渡すには、AWS STS の次の AWS CLI または AWS API オペレーションを使用します。AWS Management Console [[Switch Role](#)] 機能では、セッションタグを渡すことはできません。

セッションタグを推移的に設定することもできます。推移的なタグは、ロールの連鎖中も保持されま  
す。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

### セッションタグを渡すための方法の比較

操作	だれがロールを引き受けるか	タグを渡す方法	推移的なタグを設定する方法
<a href="#">assume-role</a> CLI または <a href="#">AssumeRole</a> API オペレーション	IAM ユーザーまたはセッション	Tags API パラメータまたは <code>--tags</code> CLI オプション	TransitiveTagKeys API パラメータまたは <code>--transitive-tag-keys</code> CLI オプション
<a href="#">assume-role-with-saml</a> CLI または <a href="#">AssumeRoleWithSAML</a> API オペレーション	SAML ID プロバイダーを使用して認証されたユーザー	PrincipalTag SAML 属性	TransitiveTagKeys SAML 属性
<a href="#">assume-role-with-web-identity</a> CLI または <a href="#">AssumeRoleWithWebIdentity</a> API オペレーション	OIDC プロバイダーを使用して認証されたユーザー	PrincipalTag OIDC トークン	TransitiveTagKeys OIDC トークン
<a href="#">get-federation-token</a> CLI または <a href="#">GetFederationToken</a> API オペレーション	IAM ユーザーまたはルートユーザー	Tags API パラメータまたは <code>--tags</code> CLI オプション	サポートされていません

セッションのタグ付けをサポートするオペレーションは、次の条件で失敗する可能性があります。

- 50 を超えるセッションタグを渡している。
- セッションタグキーのプレーンテキストが 128 文字を超えている。
- セッションタグ値のプレーンテキストが 256 文字を超えている。
- セッションポリシーのプレーンテキストの合計サイズが 2048 文字を超えている。
- セッションポリシーとセッションタグを組み合わせた合計パックサイズが大きすぎる。オペレーションが失敗した場合、エラーメッセージは、ポリシーとタグの組み合わせが上限サイズにどれだけ近いかをパーセントで示します。

## セッションタグについて知っておくべきこと

セッションタグを使用する前に、セッションとタグに関する次の詳細を確認してください。

- セッションタグを使用する場合、タグを渡す ID プロバイダー (IdP) に接続されているすべてのロールの信頼ポリシーに [sts:TagSession](#) アクセス許可が必要です。信頼ポリシーでこのアクセス許可を持たないロールの場合、AssumeRole オペレーションは失敗します。
- セッションを要求するときに、プリンシパルタグをセッションタグとして指定できます。タグは、セッションの認証情報を使用して行うリクエストに適用されます。
- セッションタグはキーバリューのペアです。たとえば、セッションに連絡先情報を追加するには、セッションタグキー email とタグ値 johndoe@example.com を追加します。
- セッションタグは、[IAM および AWS STS のタグ命名規則](#)に従う必要があります。このトピックでは、セッションタグに適用される大文字と小文字の区別と制限付きプレフィックスについて説明します。
- 新しいセッションタグは、引き受けた既存のロールまたはフェデレーションユーザーのタグのうち、同じタグキー (大文字/小文字は問わない) を持つタグをオーバーライドします。
- AWS Management Console を使用してセッションタグを渡すことはできません。
- セッションタグは、現在のセッションに対してのみ有効です。
- セッションタグは [ロールの連鎖](#)をサポートします。デフォルトでは、AWS STS は後続のロールセッションに渡されません。ただし、セッションタグを推移的に設定することはできます。推移的なタグは、ロールの連鎖中は保持され、ロールの信頼ポリシーの評価後に一致する ResourceTag の値を置き換えます。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。
- セッションタグを使用して、リソースへのアクセスを制御したり、後続のセッションに渡すことができるタグを制御できます。詳細については、「[IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)」を参照してください。



- セッションのプリンシパルタグ (セッションタグを含む) を AWS CloudTrail ログに表示できます。詳細については、「[CloudTrail でのセッションタグの表示](#)」を参照してください。
- セッションタグごとに 1 つの値を渡す必要があります。AWS STS は、複数值を持つセッションタグをサポートしていません。
- 最大 50 個のセッションタグを渡すことができます。AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。
- AWS 変換は、渡されたセッションポリシーとセッションタグを結合して、個別の制限を持つひとまとめのバイナリ形式に圧縮します。この制限を超えた場合、AWS CLI または AWS API エラーメッセージは、ポリシーとタグの組み合わせが上限サイズにどれだけ近いかをパーセントで示します。

## セッションタグの追加に必要なアクセス許可

API オペレーションに一致するアクションに加えて、ポリシーには次のアクセス許可のみのアクションが必要です。

```
sts:TagSession
```

### Important

セッションタグを使用する場合、ID プロバイダー (IdP) に接続されているすべてのロールの信頼ポリシーに `sts:TagSession` アクセス許可が必要です。このアクセス許可なしでセッションタグを渡している IdP に接続されているロールでは、`AssumeRole` オペレーションは失敗します。各ロールのロール信頼ポリシーを更新しない場合は、セッションタグを渡すために個別の IdP インスタンスを使用できます。その後、個別の IdP に接続されているロールにのみ `sts:TagSession` アクセス許可を追加します。

`sts:TagSession` アクションは、次の条件キーで使用できます。

- [aws:PrincipalTag](#) – このキーを使用して、リクエストを行うプリンシパルにアタッチされたタグと、ポリシーで指定したタグを比較します。たとえば、リクエストを行うプリンシパルに指定されたタグがある場合にのみ、プリンシパルがセッションタグを渡すことを許可できます。
- [aws:RequestTag](#) – このキーを使用して、リクエストで渡されたタグキーバリューのペアと、ポリシーで指定したタグペアを比較します。たとえば、プリンシパルが指定したセッションタグを渡すことを許可し、指定した値のみを渡すことができます。



- [aws:ResourceTag](#) – このキーを使用して、ポリシーで指定したタグキーバリューのペアと、リソースにアタッチされているキーバリューのペアを比較します。たとえば、プリンシパルが引き受けるロールに指定されたタグが含まれている場合にのみ、セッションタグを渡すことを許可できます。
- [aws:TagKeys](#) – このキーを使用して、リクエスト内のタグキーとポリシーで指定したキーを比較します。たとえば、プリンシパルが指定したタグキーを持つセッションタグのみを渡すことを許可できます。この条件キーは、渡すことができるセッションタグの最大セットを制限します。
- [sts:TransitiveTagKeys](#) – このキーを使用して、リクエスト内の推移的なセッションタグキーとポリシーで指定されたセッションタグキーを比較します。たとえば、プリンシパルが特定のタグのみを推移的に設定することを許可するポリシーを記述できます。推移的なタグは、ロールの連鎖中も保持されます。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

たとえば、次の[ロール信頼ポリシー](#)では、test-session-tags ユーザーはポリシーがアタッチされているロールを引き受けることができます。そのユーザーがロールを引き受けるときは、AWS CLI または AWS API を使用して、3 つの必須セッションタグと必要な[外部 ID](#) を渡す必要があります。さらに、ユーザーは Project および Department タグを推移的に設定することもできます。

#### Example セッションタグのロール信頼ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowIamUserAssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Project": "*",
          "aws:RequestTag/CostCenter": "*",
          "aws:RequestTag/Department": "*"
        },
        "StringEquals": {"sts:ExternalId": "Example987"}
      }
    },
    {
      "Sid": "AllowPassSessionTagsAndTransitive",
      "Effect": "Allow",
```

```
"Action": "sts:TagSession",
"Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
"Condition": {
  "StringLike": {
    "aws:RequestTag/Project": "*",
    "aws:RequestTag/CostCenter": "*"
  },
  "StringEquals": {
    "aws:RequestTag/Department": [
      "Engineering",
      "Marketing"
    ]
  },
  "ForAllValues:StringEquals": {
    "sts:TransitiveTagKeys": [
      "Project",
      "Department"
    ]
  }
}
}
```

## このポリシーで行うこと

- AllowIamUserAssumeRole ステートメントは、ポリシーがアタッチされているロールを引き受けることを test-session-tags ユーザーに許可します。そのユーザーがロールを引き受けるときは、必要なセッションタグと [外部 ID](#) を渡す必要があります。
- このステートメントの最初の条件ブロックでは、ユーザーは、Project、CostCenter、および Department セッションタグを渡す必要があります。このステートメントではタグの値は重要ではないため、タグ値にはワイルドカード (\*) を使用しました。このブロックでは、ユーザーは少なくともこれら 3 つのセッションタグを渡します。それ以外の場合は、このオペレーションは失敗します。ユーザーは追加のタグを渡すことができます。
- 2 番目の条件ブロックでは、ユーザーは値 Example987 とともに [外部 ID](#) を渡す必要があります。
- AllowPassSessionTagsAndTransitive ステートメントは、sts:TagSession アクセス許可のみのアクションを許可します。このアクションは、ユーザーがセッションタグを渡す前に許可する必要があります。ポリシーに 2 番目のステートメントを含まない最初のステートメントが含まれている場合、ユーザーはロールを引き受けることができません。

- このステートメントの最初の条件ブロックでは、ユーザーは CostCenter および Project セッションタグに任意の値を渡すことができます。これを行うには、ポリシーのタグ値にワイルドカード (\*) を使用します。この場合、[StringLike](#) 条件演算子を使用する必要があります。
- 2 番目の条件ブロックでは、ユーザーは、Department セッションタグの Engineering または Marketing 値のみを渡すことができます。
- 3 番目の条件ブロックは、推移的として設定できるタグの最大セットを一覧表示します。ユーザーは、サブセットを設定するか、タグを推移的として設定しないかを選択できます。追加のタグを推移的として設定することはできません。"Null": {"sts:TransitiveTagKeys":"false"} を含む別の条件ブロックを追加することで、タグの少なくとも 1 つを推移的として設定するよう要求できます。

## AssumeRole を使用したセッションタグの受け渡し

AssumeRole オペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。IAM ユーザーまたはロールの認証情報を使用して AssumeRole を呼び出すことができます。ロールを引き受けるときにセッションタグを渡すには、`--tags` AWS CLI オプションまたは `Tags` AWS API パラメータを使用します。

タグを推移的として設定するには、`--transitive-tag-keys` AWS CLI オプションまたは `TransitiveTagKeys` AWS API パラメータを使用します。推移的なタグは、ロールの連鎖中も保持されます。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

次の例は、AssumeRole を使用するリクエストの例を示しています。この例では、`my-role-example` ロールを引き受けるときに `my-session` という名前のセッションを作成します。セッションタグのキーバリューのペア `Project = Automation`、`CostCenter = 12345`、および `Department = Engineering` を追加します。また、Project タグと Department のタグのキーを指定して、推移的タグとして設定します。

### Example AssumeRole CLI リクエストの例

```
aws sts assume-role \  
--role-arn arn:aws:iam::123456789012:role/my-role-example \  
--role-session-name my-session \  
--tags Key=Project,Value=Automation Key=CostCenter,Value=12345 \  
Key=Department,Value=Engineering \  
--transitive-tag-keys Project Department \  
--external-id Example987
```

## AssumeRoleWithSAML を使用したセッションタグの受け渡し

AssumeRoleWithSAML オペレーションは、SAML ベースのフェデレーションを使用して認証されます。このオペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。SAML ベースのフェデレーションを使用して AWS Management Console にアクセスする方法の詳細については、「[SAML 2.0 フェデレーテッドユーザーが AWS Management Console にアクセス可能にする](#)」を参照してください。AWS CLI または AWS API アクセスの詳細については、「[SAML 2.0 フェデレーション](#)」を参照してください。Active Directory ユーザーの SAML フェデレーションを設定するチュートリアルについては、AWS セキュリティブログの「[Active Directory フェデレーションサービスを使用した AWS フェデレーション認証 \(ADFS\)](#)」を参照してください。

管理者は、企業ディレクトリのメンバーに AWS STS AssumeRoleWithSAML オペレーションを使用する AWS フェデレーションを許可できます。これを行うには、以下のタスクを完了する必要があります。

1. [ネットワークを AWS の SAML プロバイダーとして設定する](#)
2. [IAM で SAML プロバイダーを作成するには](#)
3. [フェデレーテッドユーザーのロールとそのアクセス許可を AWS で設定する](#)
4. [SAML IdP の設定を終了し、SAML 認証レスポンスのアサーションを作成する](#)

AWS には、ID ソリューションを使用したセッションタグのエンドツーエンドのエクスペリエンスを認定したパートナーが含まれます。これらの ID プロバイダーを使用してセッションタグを設定する方法については、「[サードパーティーの SAML ソリューションプロバイダーを AWS に統合する](#)」を参照してください。

SAML 属性をセッションタグとして渡すには、Attribute 属性を Name に設定した `https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}` 要素を含めます。AttributeValue 要素を使用して、タグの値を指定します。セッションタグごとに個別の Attribute 要素を含めます。

たとえば、次の ID 属性をセッションタグとして渡すとします。

- Project:Automation
- CostCenter:12345
- Department:Engineering

これらの属性を渡すには、SAML アサーションに以下の要素を含めます。

## Example SAML アサーションのスニペットの例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Automation</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Department">
  <AttributeValue>Engineering</AttributeValue>
</Attribute>
```

上記のタグを推移的として設定するには、Name 属性を `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys` に設定した別の Attribute 要素を含めます。推移的なタグは、ロールの連鎖中も保持されます。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

Project タグと Department タグを推移的として設定するには、次の多値属性を使用します。

## Example SAML アサーションのスニペットの例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>Department</AttributeValue>
</Attribute>
```

## AssumeRoleWithWebIdentity を使用したセッションタグの受け渡し

AssumeRoleWithWebIdentity オペレーションは、OpenID Connect (OIDC) 準拠のフェデレーションを使用して認証されます。このオペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。AWS Management Console アクセスにウェブ ID フェデレーションを使用する方法の詳細については、「[OIDC フェデレーション](#)」を参照してください。

OpenID Connect (OIDC) からセッションタグを渡すには、JSON ウェブトークン (JWT) にセッションタグを含める必要があります。AssumeRoleWithWebIdentity リクエストを送信するときに、トークンの `https://aws.amazon.com/` tags 名前空間にセッションタグを含めます。OIDC トークンとクレームの詳細については、「Amazon Cognito 開発者ガイド」の「[ユーザープールでのトークンの使用](#)」を参照してください。

たとえば、次のデコードされた JWT は、Project、CostCenter、および Department セッションタグを使用して AssumeRoleWithWebIdentity を呼び出すために使用されるトークンです。

トークンはまた、CostCenter タグと Project タグを推移的に設定します。推移的なタグは、ロールの連鎖中も保持されます。詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

#### Example デコードされた JSON ウェブトークンの例

```
{
  "sub": "johndoe",
  "aud": "ac_oic_client",
  "jti": "ZYUCeRMQVtqHypVPWAN3VB",
  "iss": "https://xyz.com",
  "iat": 1566583294,
  "exp": 1566583354,
  "auth_time": 1566583292,
  "https://aws.amazon.com/tags": {
    "principal_tags": {
      "Project": ["Automation"],
      "CostCenter": ["987654"],
      "Department": ["Engineering"]
    },
    "transitive_tag_keys": [
      "Project",
      "CostCenter"
    ]
  }
}
```

#### GetFederationToken を使用したセッションタグの受け渡し

GetFederationToken では、ユーザーをフェデレートできます。このオペレーションは、AWS リソースへのアクセスに使用できる一時的な認証情報のセットを返します。フェデレーテッドユーザーセッションにタグを追加するには、--tags AWS CLI オプションまたは Tags AWS API パラメータを使用します。GetFederationToken を使用する場合、一時的な資格情報を使用してロールを引き受けることができないため、セッションタグを推移的なものとして設定することはできません。この場合、ロールチェーンを使用することはできません。

次の例は、GetFederationToken を使用したレスポンスのサンプルです。この例では、トークンをリクエストするときに、my-fed-user という名前のセッションを作成します。セッションタグのキーバリューのペア Project = Automation と Department = Engineering を追加します。

## Example GetFederationToken の CLI リクエストの例

```
aws sts get-federation-token \  
--name my-fed-user \  
--tags key=Project,value=Automation key=Department,value=Engineering
```

GetFederationToken オペレーションによって返される一時的な認証情報を使用すると、セッションのプリンシパルタグには、ユーザーのタグと渡されたセッションタグが含まれます。

## セッションタグを使用したロールの連鎖

1つのロールを引き受け、一時的な認証情報を使用して別のロールを引き受けることができます。セッション間で続行できます。これを[ロールの連鎖](#)と呼びます。ロールを引き受けるときにセッションタグを渡すと、キーを推移的に設定できます。これにより、これらのセッションタグがロールチェーン内の後続のセッションに渡されるようになります。ロールタグを推移的として設定することはできません。これらのタグを後続のセッションに渡すには、セッションタグとして指定します。

### Note

推移的なタグは、ロールの連鎖中は保持され、ロールの信頼ポリシーの評価後に一致する ResourceTag の値を置き換えます。

次の例は、AWS STS がセッションタグ、推移タグ、およびロールタグをロールチェーン内の後続のセッションに渡す方法を示しています。

次のロールの連鎖シナリオ例では、AWS CLI で IAM ユーザーのアクセスキーを使用して Role1 という名前のロールを引き受けます。次に、結果のセッション認証情報を使用して、Role2 という名前の 2 番目のロールを引き受けます。次に、2 番目のセッション認証情報を使用して、Role3 という 3 番目のロールを引き受けることができます。これらのリクエストは、3 つの個別のオペレーションとして発生します。各ロールはすでに IAM でタグ付けされています。また、リクエストごとに、追加のセッションタグを渡します。

ロールをチェーン化すると、以前のセッションのタグが後のセッションにも確実に保持されるようになります。assume-role CLI コマンドを使用してこれを行うには、タグをセッションタグとして渡し、タグを推移的に設定する必要があります。タグ Star = 1 をセッションタグとして渡します。タグ Heart = 1 はロールにアタッチされ、セッションの使用時にプリンシパルタグとして適用されます。ただし、Heart = 1 タグを自動的に 2 番目または 3 番目のセッションに渡すこともできます。



これを行うには、セッションタグとして手動で含めます。作成されるセッションプリンシパルタグには、これらのタグの両方が含まれ、推移的として設定されます。

このリクエストは、次の AWS CLI コマンドを使用して実行します。

#### Example AssumeRole CLI リクエストの例

```
aws sts assume-role \  
--role-arn arn:aws:iam::123456789012:role/Role1 \  
--role-session-name Session1 \  
--tags Key=Star,Value=1 Key=Heart,Value=1 \  
--transitive-tag-keys Star Heart
```

次に、そのセッションの認証情報を使用して、Role2 を引き受けます。タグ Sun = 2 は 2 番目のロールにアタッチされ、2 番目のセッションを使用するときにプリンシパルタグとして適用されます。Star タグと Heart タグは、最初のセッションで推移的なセッションタグから継承されます。2 番目のセッションの結果となるプリンシパルタグは、Heart = 1、Star = 1、および Sun = 2 です。Heart および Star は引き続き推移的です。Role2 にアタッチされた Sun タグは、セッションタグではないため、推移的としてマークされていません。今後のセッションはこのタグを継承しません。

この 2 番目のリクエストは、次の AWS CLI コマンドを使用して実行します。

#### Example AssumeRole CLI リクエストの例

```
aws sts assume-role \  
--role-arn arn:aws:iam::123456789012:role/Role2 \  
--role-session-name Session2
```

次に、2 番目のセッション認証情報を使用して、Role3 を引き受けます。3 番目のセッションのプリンシパルタグは、新しいセッションタグ、継承された推移的なセッションタグ、およびロールタグから取得されます。2 番目のセッションの Heart = 1 タグと Star = 1 タグは最初のセッションでの推移的なタグから継承されました。Sun = 2 セッションタグを渡そうとすると、オペレーションは失敗します。継承された Star = 1 セッションタグは、ロールの Star = 3 タグを上書きします。ロールの連鎖では、ロールの信頼ポリシーの評価後に、推移的なタグの値によって ResourceTag の値に一致するロールがオーバーライドされます。この例では、この例では、Role3 がロール信頼ポリシーで Star を ResourceTag として使用し、呼び出し元のロールセッションからの推移的なタグ値に ResourceTag 値を設定する場合。ロールの Lightning タグは 3 番目のセッションにも適用され、推移的として設定されません。



3 番目のリクエストを実行するには、次の AWS CLI コマンドを使用します。

#### Example AssumeRole CLI リクエストの例

```
aws sts assume-role \  
--role-arn arn:aws:iam::123456789012:role/Role3 \  
--role-session-name Session3
```

## ABAC でのセッションタグの使用

属性ベースのアクセスコントロール (ABAC) は、タグ属性に基づいてアクセス許可を定義する認可戦略です。

企業ユーザー ID に SAML ベースの ID プロバイダー (IdP) を使用している場合は、セッションタグを AWS に渡すように アサーションを設定できます。たとえば、企業のユーザー ID の場合、従業員が AWS にフェデレーションすると、AWS は結果のプリンシパルに属性を適用します。その後、ABAC を使用して、これらの属性に基づいてアクセス許可を許可または拒否できます。詳細については、「[IAM チュートリアル: ABAC で SAML セッションタグを使用する](#)」を参照してください。

IAM Identity Center で ABAC を使用方法の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[アクセスコントロールの属性](#)」を参照してください。

## CloudTrail でのセッションタグの表示

AWS CloudTrail を使用して、ロールを引き受けるか、ユーザーをフェデレートするために行われたリクエストを表示できます。CloudTrail ログファイルには、引き受けたロールまたはフェデレートされたユーザーセッションのプリンシパルタグに関する情報が含まれます。詳細については、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」を参照してください。

たとえば、AWS STS AssumeRoleWithSAML リクエストを行い、セッションタグを渡し、これらのタグを推移的として設定するとします。CloudTrail ログには、次の情報があります。

#### Example AssumeRoleWithSAML CloudTrail ログの例

```
"requestParameters": {  
  "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",  
  "roleSessionName": "MyRoleSessionName",  
  "principalTags": {  
    "CostCenter": "987654",
```

```
    "Project": "Unicorn"
  },
  "transitiveTagKeys": [
    "CostCenter",
    "Project"
  ],
  "durationSeconds": 3600,
  "roleArn": "arn:aws:iam::123456789012:role/SAMLTestRoleShibboleth",
  "principalArn": "arn:aws:iam::123456789012:saml-provider/Shibboleth"
},
```

次の CloudTrail ログ例を参照して、セッションタグを使用するイベントを表示できます。

- [CloudTrail ログファイル内の AWS STS ロール連鎖 API イベントの例](#)
- [CloudTrail ログファイル内の SAML AWS STS API イベントの例](#)
- [CloudTrail ログファイル内の OIDC AWS STS API イベントの例](#)

## AWS CloudTrail による IAM および AWS STS の API コールのログ記録

IAM および AWS STS は AWS CloudTrail と統合されています。このサービスは、IAM ユーザーやロールのサービスによって実行されたアクションを記録するサービスです。CloudTrail は、コンソールからの呼び出しや API 呼び出しを含む、IAM と AWS STS のすべての API 呼び出しをイベントとしてキャプチャします。証跡を作成する場合は、 のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができません 証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail を使用して、IAM または AWS STS に対して行われたリクエストに関する情報を取得できます。たとえば、リクエストの実行元 IP アドレス、実行者、実行日時、およびその他の詳細を表示できます。

CloudTrail の詳細については、[AWS CloudTrail ユーザーガイド](#)を参照してください。

### トピック

- [CloudTrail での IAM および AWS STS 情報](#)
- [IAM および AWS STS API リクエストのログ記録](#)
- [他の AWS のサービスへの API リクエストのログ記録](#)
- [ユーザーサインインイベントのログ記録](#)
- [一時的な認証情報のサインインイベントのログ記録](#)

- [CloudTrail ログの IAM API イベントの例](#)
- [例 AWS STS CloudTrail ログの API イベント](#)
- [CloudTrail ログのサインインイベントの例](#)
- [IAM ロールの信頼ポリシーの動作](#)

## CloudTrail での IAM および AWS STS 情報

CloudTrail は、AWS アカウントを作成すると、その中で有効になります。IAM または AWS STS でアクティビティが発生すると、そのアクティビティは [Event history] (イベント履歴) で AWS のその他のサービスのイベントと共に CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、「[CloudTrail イベント履歴でイベントを表示する](#)」を参照してください。

IAM や AWS STS のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべてのリージョンに証跡が適用されます。証跡は、AWSパーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- [証跡を作成するための概要](#)
- [CloudTrail がサポートするサービスと統合](#)
- [CloudTrail 用 Amazon SNS 通知の構成](#)
- 「[複数のリージョンから CloudTrail ログファイルを受け取る](#)」および「[複数のアカウントから CloudTrail ログファイルを受け取る](#)」

すべての IAM および AWS STS アクションは CloudTrail によってログに記録されます。これらのアクションについては、[IAM API リファレンス](#)と[AWS Security Token Service API リファレンス](#)。

## IAM および AWS STS API リクエストのログ記録

CloudTrail は、認証されたすべての API リクエストを IAM および AWS STS API オペレーションにログ記録します。CloudTrail は、認証されていないリクエストを AWS STS アクション、AssumeRoleWithSAML および AssumeRoleWithWebIdentity に記録し、ID プロバイダーから提供された情報も記録します。ただし、認証されていない AWS STS リクエストの中には、正当

なリクエストとして信頼されるほど十分に有効であるという最低限の要求を満たしていないため、ログに記録されないものもあります。

ログに記録された情報を使用して、引き受けたロールを持つフェデレーションユーザーによって行われた呼び出しを元の外部フェデレーション呼び出し元にマッピングできます。AssumeRole の場合、呼び出しをその元の AWS サービスまたはユーザーのアカウントに再度マッピングすることができます。CloudTrail ログエントリの JSON データの `userIdentity` セクションには、AssumeRole\* リクエストを特定のフェデレーションユーザーにマッピングするのに必要な情報が含まれます。詳細については、「AWS CloudTrail ユーザーガイド」の「[CloudTrail userIdentity Element](#)」を参照してください。

たとえば、IAM `CreateUser`、`DeleteRole`、`ListGroups`、およびその他の API オペレーションに対する呼び出しはすべて、CloudTrail によってログに記録されます。

このタイプのログエントリの例については、このトピックの後半で説明します。

## 他の AWS のサービスへの API リクエストのログ記録

他の AWS のサービス API オペレーションに対する認証されたリクエストは CloudTrail によってログに記録され、そのログエントリにはリクエストの生成元に関する情報が含まれます。

たとえば、Amazon EC2 インスタンスの一覧表示または AWS CodeDeploy のデプロイグループの作成をリクエストしたとします。リクエストを行ったユーザーまたはサービスに関する詳細は、そのリクエストのログエントリに含まれています。この情報は、リクエストが AWS アカウントのルートユーザー、IAM ユーザー、ロール、または別の AWS サービスのいずれによって行われたかを判断するのに役立ちます。

CloudTrail ログエントリのユーザー ID 情報の詳細については、[AWS CloudTrail ユーザーガイド](#)の「`userIdentity` 要素」を参照してください。

## ユーザーサインインイベントのログ記録

CloudTrail は、サインインイベントを AWS Management Console、AWS ディスカッションフォーラム、および AWS Marketplace に記録します。CloudTrail は、IAM ユーザーとフェデレートッドユーザーのサインインの成功と失敗をログに記録します。

rootユーザーのサインインが成功した場合と失敗した場合のサンプルCloudTrailイベントを表示するには、[AWS CloudTrail ユーザーガイド](#)の「ルートユーザーのイベントレコードの例」を参照してください。

セキュリティのベストプラクティスとして、AWS は、サインインの失敗した原因が間違っただユーザー名である場合、入力された IAM ユーザー名テキストをログに記録しません。ユーザー名テキストは、HIDDEN\_DUE\_TO\_SECURITY\_REASONS という値によってマスクされます。この例として、このトピックの「[間違っただユーザー名が原因で発生したサインイン失敗イベントの例](#)」を参照してください。失敗の原因がユーザーエラーであるため、ユーザー名のテキストが隠れています。これらのエラーを記録すると、機密性の高い情報が公開される可能性があります。例:

- 誤ってユーザー名ボックスにパスワードを入力した。
- ある AWS アカウント のサインインページのリンクを選択したが、別の AWS アカウント のアカウント番号を入力した。
- どのアカウントにサインインしているのかを忘れて、誤って個人メールのアカウント名や銀行のサインイン ID などのプライベート ID を入力した。

## 一時的な認証情報のサインインイベントのログ記録

プリンシパルが一時的な認証情報をリクエストすると、プリンシパルタイプによって CloudTrail がイベントをログに記録する方法が決まります。これは、プリンシパルが別のアカウントのロールを引き受ける場合に複雑になる可能性があります。ロールのクロスアカウントオペレーションに関連するオペレーションを実行するために、複数の API コールがあります。まず、プリンシパルは AWS STS API を呼び出して一時的な認証情報を取得します。このオペレーションは、呼び出し元のアカウントと AWS STS オペレーションが実行されたアカウントに記録されます。次に、プリンシパルはロールを使用して、引き受けたロールのアカウントで他の API コールを実行します。

ロール信頼ポリシーで `sts:SourceIdentity` 条件キーを使用すると、ユーザーがロールを引き受けるときに ID を指定するように要求できます。例えば、IAM ユーザーがセッション名として自分のユーザー名を指定するように要求できます。これにより、AWS の特定のアクションを実行したユーザーを特定できます。詳細については、「[sts:SourceIdentity](#)」を参照してください。[sts:RoleSessionName](#) を使用して、ユーザーが役割を引き受けるときにセッション名を指定するようにユーザーに要求することもできます。これは、AWS CloudTrail ログを確認するときに異なるプリンシパルによって使用されるロールのロールセッションを区別するのに役立ちます。

次の表は、一時認証情報を生成する各 AWS STS API に対するさまざまなユーザー ID 情報を CloudTrail がどのようにログに記録しているかを示しています。

プリンシパルタイプ	STS API	呼び出し元アカウントの CloudTrail ログ内のユーザー ID	割り当てられたロールのアカウントの CloudTrail ログのユーザー ID	ロールの後続の API コールの CloudTrail ログ内のユーザー ID
AWS アカウントのルートユーザー 認証情報	GetSessionToken	ルートユーザー ID	ロール所有者アカウントと呼び出し元アカウントが同じ	ルートユーザー ID
IAM ユーザー	GetSessionToken	IAM ユーザーアイデンティティ	ロール所有者アカウントと呼び出し元アカウントが同じ	IAM ユーザーアイデンティティ
IAM ユーザー	GetFederationToken	IAM ユーザーアイデンティティ	ロール所有者アカウントと呼び出し元アカウントが同じ	IAM ユーザーアイデンティティ
IAM ユーザー	AssumeRole	IAM ユーザーアイデンティティ	アカウント番号およびプリンシパル ID (ユーザーの場合)、または AWS サービスプリンシパル	ロール ID のみ (ユーザーの記録はしない)
外部で認証されたユーザー	AssumeRoleWithSAML	該当なし	SAML ユーザー ID	ロール ID のみ (ユーザーの記録はしない)
外部で認証されたユーザー	AssumeRoleWithWebIdentity	該当なし	OIDC/Web の ユーザー ID	ロール ID のみ (ユーザーの記録はしない)

アクションがリソースに変更を加える効果がない場合、そのアクションは CloudTrail で読み取り専用と見なされます。読み取り専用イベントをログに記録すると、CloudTrail はログに含まれる responseElements 情報を編集します。読み取り専用ではないイベントが CloudTrail によってログに記録されると、完全な responseElements がログエントリに表示されます。ただし、AWS STS API の AssumeRole、AssumeRoleWithSAML、AssumeRoleWithWebIdentity の場合、読み取り専用としてログに記録されますが、CloudTrail は、これらの API のログに完全な responseElements を含めます。

次の表は、一時認証情報を生成する各 AWS STS API に対する responseElements と readOnly の情報を CloudTrail がどのようにログに記録しているかを示しています。

STS API	レスポンス要素情報	[Read-only]
AssumeRole	含まれる	true
AssumeRoleWithSAML	含まれる	true
AssumeRoleWithWebIdentity	含まれる	true
GetFederationToken	含まれる	false
GetSessionToken	含まれる	false

## CloudTrail ログの IAM API イベントの例

CloudTrail ログファイルには、イベントが JSON 形式で書き込まれます。API イベントは 1 つの API リクエストを表すものであり、プリンシパル、リクエストされたアクション、すべてのパラメータ、およびアクションの日時に関する情報が含まれます。

### CloudTrail ログファイルの IAM API イベントの例

次の例は、IAM の GetUserPolicy アクションに対するリクエストの CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
```



```
{
  "arn": "arn:aws:iam::444455556666:user/JaneDoe",
  "accountId": "444455556666",
  "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
  "userName": "JaneDoe",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2014-07-15T21:39:40Z"
    }
  },
  "invokedBy": "signin.amazonaws.com"
},
{
  "eventTime": "2014-07-15T21:40:14Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "GetUserPolicy",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "signin.amazonaws.com",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "userName": "JaneDoe",
    "policyName": "ReadOnlyAccess-JaneDoe-201407151307"
  },
  "responseElements": null,
  "requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
  "eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE"
}
}
```

このイベント情報から、このリクエストは、ReadOnlyAccess-JaneDoe-201407151307 要素で指定された、ユーザー JaneDoe の requestParameters というユーザーポリシーを取得するために送信されたことを確認できます。また、このリクエストは、JaneDoe という IAM ユーザーによって 2014 年 7 月 15 日の午後 9 時 40 分 (UTC) に送信されたことも確認できます。この例では、userAgent 要素から、このリクエストの実行元は AWS Management Console であることがわかります。

## 例 AWS STS CloudTrail ログの API イベント

CloudTrail ログファイルには、イベントが JSON 形式で書き込まれます。API イベントは 1 つの API リクエストを表すものであり、プリンシパル、リクエストされたアクション、すべてのパラメータ、およびアクションの日時に関する情報が含まれます。



## CloudTrail ログファイル内のクロスアカウント AWS STS API イベントの例

アカウント 777788889999 の JohnDoe という名前の IAM ユーザーは、AWS STS AssumeRole アクションを呼び出して、アカウント 111122223333 のロール EC2-dev を引き受けます。アカウント管理者は、ロールを引き受けるときに、ユーザ名と同じソース ID を設定することをユーザーに要求します。ユーザーは JohnDoe のソースID値を渡します。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:user/JohnDoe",
    "accountId": "777788889999",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 boto-core/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2023, 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
    }
  },
  "sourceIdentity": "JohnDoe"
},
"resources": [
```

```
{
  "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
  "accountId": "111122223333",
  "type": "AWS::IAM::Role"
},
],
"requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
"sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
"eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}
```

2 番目の例は、同じリクエストを行ったロールアカウント (111122223333) の CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2014, 4:07:39 PM"
    },
    "assumedRoleUser": {
```

```
    "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
    "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
  },
  "sourceIdentity": "JohnDoe"
},
"requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
"sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
"eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}
```

## CloudTrail ログファイル内の AWS STS ロール連鎖 API イベントの例

次の例は、アカウント 111111111111 で John Doe によって行われたリクエストの CloudTrail ログエントリを示しています。John は以前、JohnRole1 ロールを引き受けるために JohnDoe ユーザーを使用していました。このリクエストでは、その JohnRole2 ロールの認証情報を使用してロールを引き受けます。これは、[ロールの連鎖](#)と呼ばれます。彼が JohnDoe1 ロールを引き受けたときに設定したソースIDは、JohnRole2 を引き受ける要求に存続します。ロールを引き受けるときに John が別のソース ID を設定しようとする、要求は拒否されます。John は 2 つの [セッションタグ](#) をリクエストに渡します。彼はこれらの 2 つのタグを推移的に設定します。ジョンが JohnRole1 を引き受けたときに推移的として設定したため、リクエストは Department タグを推移的として継承します。ソース ID の詳細については、[引き受けたロールで実行されるアクションのモニタリングと制御](#)を参照してください。ロールチェーン内の推移的キーの詳細については、「[セッションタグを使用したロールの連鎖](#)」を参照してください。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIN5ATK5U7KEXAMPLE:JohnRole1",
    "arn": "arn:aws:sts::111111111111:assumed-role/JohnDoe/JohnRole1",
    "accountId": "111111111111",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-10-02T21:50:54Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIN5ATK5U7KEXAMPLE",
        "arn": "arn:aws:iam::111111111111:role/JohnRole1",
```



```
+C+WKFZb701eiv9J5La2EXAMPLETOKEN/c7S5Iro1WUJ0q3Cxuo/8HUoSxVhQHM7zF7mWWLhXLEQ52ivL
+F6q5dpXu4aTFedpMfnJa8JtkWwG9x1Axj0Ypy2ok8v5unpQGWyv1vwdvj6ez1Dm8Xg1+qIzXILiEXAMPLETOKEN/
vQGqu8H+nxp3kabcrt0vTFTvxX6vsc80GwUfHhzAfYGEEXAMPLETOKEN/
L6v1yMM3B10wF0rQBno1HEjfl0NI8RnQiMNFdU0twYj7HUZI0CZmjfn8PPHq77N7GJl9lvIZKQA00wcjg
+mc78zHCj8y0siY8C96paEXAMPLETOKEN/
E3cpksxWdgs91HRzJWScjN2+r2LTGjYhyPqcmFzso2mCE7mBNEXAMPLETOKEN/oJy
+2o83YNW5t0iDmczgDzJZ4UKR84yGYOMfSnF4XcEJrDgAJ30JFwmTcTQICAlSwLEXAMPLETOKEN"
  },
  "assumedRoleUser": {
    "assumedRoleId": "AROAIFR7WHDTSOYQYHFUE:Role2WithTags",
    "arn": "arn:aws:sts::111111111111:assumed-role/test-role/Role2WithTags"
  },
  "sourceIdentity": "JohnDoe"
},
"requestID": "b96b0e4e-e561-11e9-8b3f-7b396EXAMPLE",
"eventID": "1917948f-3042-46ec-98e2-62865EXAMPLE",
"resources": [
  {
    "ARN": "arn:aws:iam::111111111111:role/JohnRole2",
    "accountId": "111111111111",
    "type": "AWS::IAM::Role"
  }
],
"eventType": "AwsApiCall",
"recipientAccountId": "111111111111"
}
```

## CloudTrail ログファイルの AWS のサービス AWS STS API イベントの例

次の例は、サービスロールのアクセス許可を使用して別のサービス API を呼び出す AWS のサービスによって行われたリクエストの CloudTrail ログエントリを示しています。アカウント 777788889999 で行われたリクエストの CloudTrail ログエントリが表示されます。

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
```

```
    "mfaAuthenticated": "false",
    "creationDate": "2016-11-14T17:25:26Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AROARSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
    "accountId": "777788889999",
    "userName": "AssumeNothing"
  }
},
"eventTime": "2016-11-14T17:25:45Z",
"eventSource": "s3.amazonaws.com",
"eventName": "DeleteBucket",
"awsRegion": "us-east-2",
"sourceIPAddress": "192.0.2.1",
"userAgent": "[aws-cli/1.11.10 Python/2.7.8
Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 boto/1.4.67]",
"requestParameters": {
  "bucketName": "my-test-bucket-cross-account"
},
"responseElements": null,
"requestID": "EXAMPLE463D56D4C",
"eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "777788889999"
}
```

## CloudTrail ログファイル内の SAML AWS STS API イベントの例

次の例は、AWS STS の AssumeRoleWithSAML アクションに対するリクエストの CloudTrail ログエントリを示しています。リクエストには SAML アサーションを介して[セッションタグ](#)として渡される SAML 属性 CostCenter と Project が含まれています。これらのタグは推移的として設定され、[ロールの連鎖シナリオでも保持されます](#)。リクエストにはオプションの API パラメータ DurationSeconds が含まれます。これは CloudTrail ログでは durationSeconds として表され、1800 秒に設定されます。リクエストには、SAML アサーションで渡される SAML 属性 sourceIdentity も含まれています。作成されたロールセッションの資格情報を使用して別のロールを引き受ける場合、このソース ID は保持されます。

```
{
  "eventVersion": "1.08",
```

```

"userIdentity": {
  "type": "SAMLUser",
  "principalId": "SampleUkh1i4+ExampLexL/jEvs=:SamlExample",
  "userName": "SamlExample",
  "identityProvider": "bdG0nTesti4+ExampLexL/jEvs="
},
"eventTime": "2023-08-28T18:30:58Z",
"eventSource": "sts.amazonaws.com",
"eventName": "AssumeRoleWithSAML",
"awsRegion": "us-east-2",
"sourceIPAddress": "AWS Internal",
"userAgent": "aws-internal/3 aws-sdk-java/1.12.479
Linux/5.10.186-157.751.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.7+11 java/17.0.7
kotlin/1.3.72 vendor/Amazon.com_Inc. cfg/retry-mode/standard",
"requestParameters": {
  "sAMLEntityID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
  "roleSessionName": "MyAssignedRoleSessionName",
  "sourceIdentity": "MySAMLUser",
  "principalTags": {
    "CostCenter": "987654",
    "Project": "Unicorn",
    "Department": "Engineering"
  },
  "transitiveTagKeys": [
    "CostCenter",
    "Project"
  ],
  "roleArn": "arn:aws:iam::444455556666:role/SAMLEntityRoleShibboleth",
  "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth",
  "durationSeconds": 1800
},
"responseElements": {
  "credentials": {
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionToken": "<encoded session token blob>",
    "expiration": "Aug 28, 2023, 7:00:58 PM"
  },
  "assumedRoleUser": {
    "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
    "arn": "arn:aws:sts::444455556666:assumed-role/SAMLEntityRoleShibboleth/MyAssignedRoleSessionName"
  },
  "packedPolicySize": 1,
  "subject": "SamlExample",

```

```
    "subjectType": "transient",
    "issuer": "https://server.example.com/idp/shibboleth",
    "audience": "https://signin.aws.amazon.com/saml",
    "nameQualifier": "bdG0nTesti4+ExampLexL/jEvs=",
    "sourceIdentity": "MySAMLUser"
  },
  "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "accountId": "444455556666",
      "type": "AWS::IAM::Role",
      "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth"
    },
    {
      "accountId": "444455556666",
      "type": "AWS::IAM::SAMLProvider",
      "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "444455556666",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
  }
}
```

## CloudTrail ログファイル内の OIDC AWS STS API イベントの例

次の例は、AWS STS の AssumeRoleWithWebIdentity アクションに対するリクエストの CloudTrail ログエントリを示しています。リクエストには ID プロバイダトークンを介して [セッションタグ](#)として渡される属性 CostCenter と Project が含まれています。これらのタグは推移的として設定され、[ロールの連鎖でも保持されます](#)。リクエストには、ID プロバイダトークンからの sourceIdentity 属性が含まれています。作成されたロールセッションの資格情報を使用して別のロールを引き受ける場合、このソース ID は保持されます。

```
{
```



```
"eventVersion": "1.05",
"userIdentity": {
  "type": "WebIdentityUser",
  "principalId": "accounts.google.com:<id-of-
application>.apps.googleusercontent.com:<id-of-user>",
  "userName": "<id of user>",
  "identityProvider": "accounts.google.com"
},
"eventTime": "2016-03-23T01:39:51Z",
"eventSource": "sts.amazonaws.com",
"eventName": "AssumeRoleWithWebIdentity",
"awsRegion": "us-east-2",
"sourceIPAddress": "192.0.2.101",
"userAgent": "aws-cli/1.3.23 Python/2.7.6 Linux/2.6.18-164.el5",
"requestParameters": {
  "sourceIdentity": "MyWebIdentityUser",
  "durationSeconds": 3600,
  "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
  "roleSessionName": "MyAssignedRoleSessionName"
  "principalTags": {
    "CostCenter": "24680",
    "Project": "Pegasus"
  },
  "transitiveTagKeys": [
    "CostCenter",
    "Project"
  ],
},
"responseElements": {
  "provider": "accounts.google.com",
  "subjectFromWebIdentityToken": "<id of user>",
  "sourceIdentity": "MyWebIdentityUser",
  "audience": "<id of application>.apps.googleusercontent.com",
  "credentials": {
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "expiration": "Mar 23, 2016, 2:39:51 AM",
    "sessionToken": "<encoded session token blob>"
  },
  "assumedRoleUser": {
    "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:MyAssignedRoleSessionName",
    "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/
MyAssignedRoleSessionName"
  }
},
```

```
"resources": [
  {
    "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "accountId": "444455556666",
    "type": "AWS::IAM::Role"
  }
],
"requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
"eventID": "bEXAMPLE-0b30-4246-b28c-e3da3EXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "444455556666"
}
```

## CloudTrail ログのサインインイベントの例

CloudTrail ログファイルには、イベントが JSON 形式で書き込まれます。サインインイベントは、シングルサインインリクエストを表し、サインインプリンシパル、リージョン、およびアクションの日時に関する情報を含みます。

### CloudTrail ログファイル内のサインイン成功イベントの例

成功したサインインイベントの CloudTrail のログエントリの例を以下に示します。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/JohnDoe",
    "accountId": "111122223333",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-16T15:49:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.110",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
Firefox/24.0",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Success"
  },
}
```

```
"additionalEventData": {
  "MobileVersion": "No",
  "LoginTo": "https://console.aws.amazon.com/s3/ ",
  "MFAUsed": "No"
},
"eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

CloudTrail ログファイルに含まれる情報の詳細については、[ユーザーガイド](#)の「AWS CloudTrail CloudTrail イベントリファレンス」を参照してください。

## CloudTrail ログファイル内のサインイン失敗イベントの例

失敗したサインインイベントの CloudTrail のログエントリの例を以下に示します。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/JaneDoe",
    "accountId": "111122223333",
    "userName": "JaneDoe"
  },
  "eventTime": "2014-07-08T17:35:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.100",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
  "errorMessage": "Failed authentication",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/sns",
    "MFAUsed": "No"
  },
  "eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
}
```

この情報から、JaneDoe 要素に示されているように、このサインインは `userIdentity` という IAM ユーザーによって試行されたことを確認できます。また、`responseElements` 要素に示されているように、サインインの試行が失敗したこともわかります。さらに、JaneDoe が 2014 年 7 月 8 日、午後 5 時 35 分 (UTC) に Amazon SNS コンソールにサインインしようとしたことも確認できません。

## 間違ったユーザー名が原因で発生したサインイン失敗イベントの例

以下の例は、ユーザーが間違ったユーザー名を入力したことが原因で発生したサインイン失敗イベントの CloudTrail ログ項目を示しています。AWS は、`userName` テキストを `HIDDEN_DUE_TO_SECURITY_REASONS` でマスクして、機密となっている可能性のある情報が漏れないようにします。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "accountId": "123456789012",
    "accessKeyId": "",
    "userName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "eventTime": "2015-03-31T22:20:42Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
Firefox/24.0",
  "errorMessage": "No username found in supplied account",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs
%23&isauthcode=true",
    "MobileVersion": "No",
    "MFAUsed": "No"
  },
  "eventID": "a7654656-0417-45c6-9386-ea8231385051",
  "eventType": "AwsConsoleSignin",
  "recipientAccountId": "123456789012"
```

}

## IAM ロールの信頼ポリシーの動作

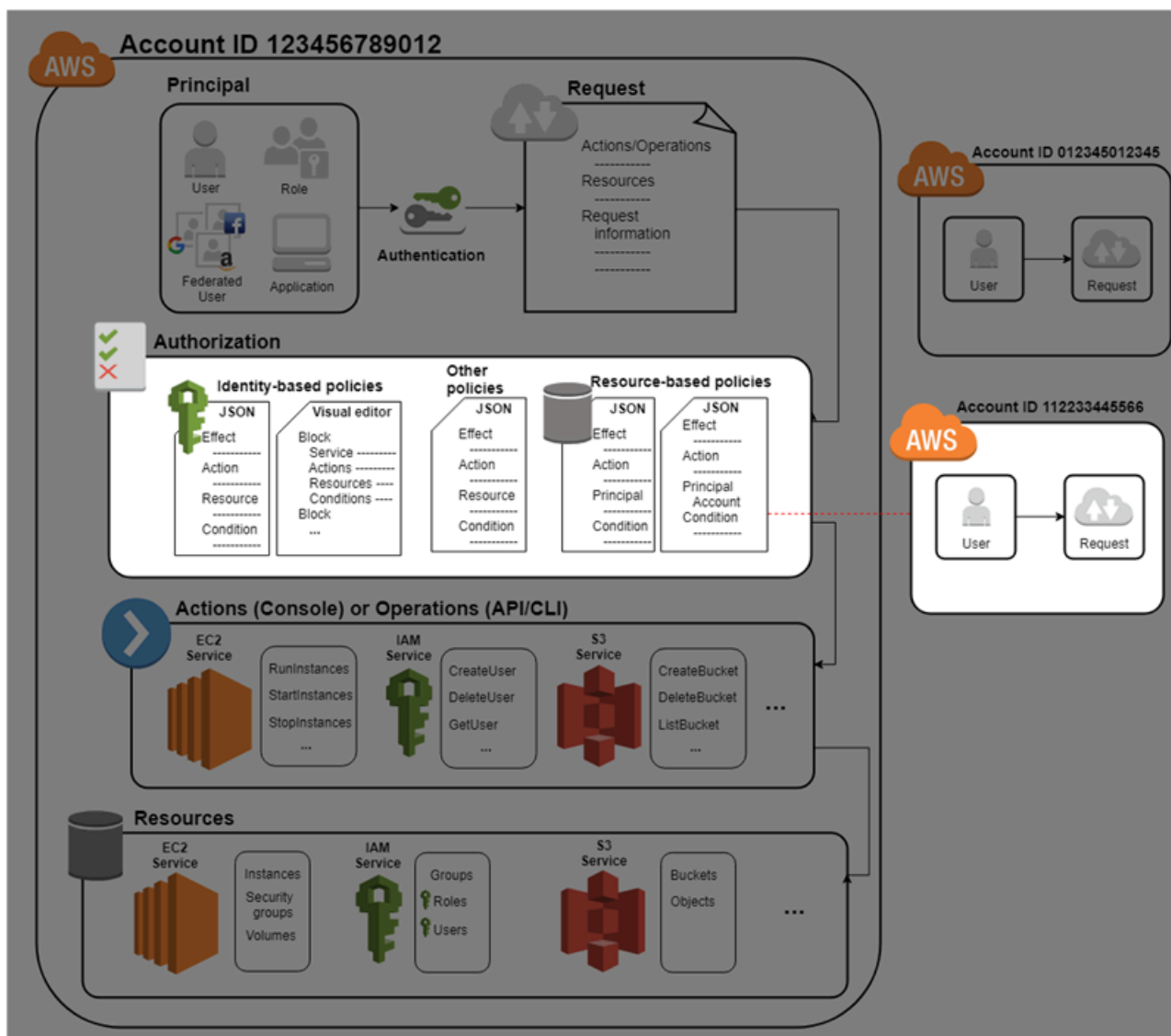
2022 年 9 月 21 日、AWS は IAM ロールの信頼ポリシーの動作を変更し、ロールがそれ自体を引き受けるときにロールの信頼ポリシーで明示的な許可を必要とするようにしました。レガシー動作の許可リストに含まれている IAM ロールには、AssumeRole イベントのための explicitTrustGrant に関連する additionalEventData フィールドがあります。レガシー許可リストのロールがレガシー動作を使用してそれ自体を引き受けた場合、explicitTrustGrant の値は false になります。レガシー許可リストのロールがそれ自体を引き受けたとしても、ロールの信頼ポリシーの動作が更新され、ロールがそれ自体を引き受けることが明示的に許可されている場合、explicitTrustGrant の値は true になります。

レガシー動作の許可リストに含まれている IAM ロールの数は非常に少ないです。このフィールドがこれらのロールの CloudTrail ログに存在するのは、これらのロールがそれら自体を引き受けた場合にのみです。ほとんどの場合、IAM ロールがそれ自体を引き受ける必要はありません。AWS では、プロセス、コード、または設定を更新してこの動作を削除するか、ロールの信頼ポリシーを更新してこの動作を明示的に許可することをお勧めしています。詳細については、「[Announcing an update to IAM role trust policy behavior](#)」を参照してください。

# AWS リソースの アクセス管理

AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスを安全に管理するためのウェブサービスです。AWS で [プリンシパル](#) がリクエストを行うと、AWS 強制コードは、プリンシパルが認証 (サインイン) され、許可されている (アクセス許可を持っている) かどうかをチェックします。AWS でアクセスを管理するには、ポリシーを作成して IAM ID や AWS リソースにアタッチします。ポリシーは、アイデンティティやリソースにアタッチして、そのアクセス許可を定義する、AWS の JSON ポリシードキュメントです。ポリシーのタイプと用途の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

残りの認証と許可の詳細については、「[IAM の仕組み](#)」を参照してください。



認可の間、AWS エンフォースメントコードでは、[リクエストコンテキスト](#)からの値に基づいて、一致するポリシーをチェックし、リクエストの許可または拒否を決定します。

AWS は、リクエストのコンテキストに該当する各ポリシーをチェックします。1つのポリシーでリクエストが拒否されると、そのリクエストは AWS で全面的に拒否され、ポリシーの評価が停止します。このプロセスは明示的な拒否と呼ばれています。リクエストはデフォルトで拒否されるため、IAM では、リクエストのすべての部分が該当するポリシーによって許可された場合に限り、リクエストを承認します。単一アカウント内のリクエストの[評価ロジック](#)は、以下のルールに基づきます。

- デフォルトでは、すべてのリクエストが明示的に拒否されます。(または、AWS アカウントのルートユーザーには、デフォルトでフルアクセス権が付与されています)。
- アイデンティティベースのポリシーまたはリソースベースのポリシーに明示的な許可が含まれている場合、このデフォルト設定は上書きされます。
- アクセス許可の境界、Organizations SCP、またはセッションポリシーがある場合、この許可は明示的な拒否で上書きされる場合があります。
- ポリシー内の明示的な拒否は、すべての許可に優先します。

リクエストが認証され承認された後で、AWS はリクエストを承認します。別のアカウントでリクエストする必要がある場合は、別のアカウントポリシーで、リソースへのアクセスを許可する必要があります。さらに、リクエストに使用する IAM エンティティに、そのリクエストを許可するアイデンティティベースのポリシーが必要です。

## アクセス管理リソース

権限に関する詳細およびポリシーの作成に関する詳細については、以下のリソースを参照してください。

AWS セキュリティブログの以下のエントリは、Amazon S3 バケットおよびオブジェクトへのアクセスに関するポリシーを記述するための一般的な方法について説明しています。

- [IAM ポリシーの記述: Amazon S3 バケットへのアクセス権を付与する方法](#)
- [IAM ポリシーの記述: Amazon S3 バケット内のユーザー固有のフォルダへのアクセスを許可する方法](#)
- [IAM ポリシー、バケットポリシー、および ACL! Oh, My! \(S3 リソースへのアクセス制御\)](#)
- [RDS のリソース レベルのアクセス許可入門](#)

- [EC2 リソースレベルアクセス許可とは](#)

## IAM でのポリシーとアクセス許可

AWS でのアクセスを管理するには、ポリシーを作成し、IAM アイデンティティ (ユーザー、ユーザーのグループ、ロール) または AWS リソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、IAM プリンシパル (ユーザーまたはロール) によってリクエストが行われると、それらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。通常、ポリシーは JSON ドキュメントとして AWS に保存されます。AWS は、6 つのポリシータイプ (アイデンティティベースのポリシー、リソースベースのポリシー、アクセス許可の境界、SCP、ACL、セッションポリシー) をサポートします。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、[GetUser](#) アクションを許可するポリシーを適用されたユーザーは、AWS Management Console、AWS CLI、または AWS API からユーザー情報を取得できます。IAM ユーザーを作成したら、コンソールまたはプログラムによるアクセスを許可するように選択できます。コンソールへのアクセスが許可されている場合、IAM ユーザーは、サインイン認証情報を使用してコンソールにサインインできます。プログラムによるアクセスが許可されている場合、ユーザーは、アクセスキーで CLI または API を使用します。

### ポリシータイプ

AWS では、使用頻度の高いものから低いものの順にリストされた次のポリシータイプを使用できます。詳細については、以下のポリシータイプ別のセクションを参照してください。

- [ID ベースのポリシー](#) – [管理](#)ポリシーと [インライン](#)ポリシーを IAM ID (ユーザー、ユーザーが所属するグループ、またはロール) に添付することができます。アイデンティティベースのポリシーでは、アクセス許可はアイデンティティに付与されます。
- [リソースベースのポリシー](#) – インラインポリシーをリソースにアタッチします。リソースベースのポリシーとして最も一般的な例は、Amazon S3 バケットポリシーと IAM ロールの信頼ポリシーです。リソースベースのポリシーでは、アクセス許可は、ポリシーで指定されているプリンシパルに付与されます。プリンシパルは、リソースと同じアカウントか、別のアカウントになります。
- [アクセス許可の境界](#) – IAM エンティティ (ユーザーまたはロール) のアクセス許可の境界として、管理ポリシーを使用します。このポリシーでは、アイデンティティベースのポリシーでエンティティに付与できるアクセス許可の上限を定義しますが、アクセス許可は付与されません。アクセス



許可の境界では、リソースベースのポリシーでエンティティに付与できるアクセス許可の上限は定義されません。

- [Organizations SCP](#) – AWS Organizations サービスコントロールポリシー (SCP) を使用して、組織または組織単位 (OU) のメンバーアカウントのアクセス許可の上限を定義します。SCP では、アイデンティティベースのポリシーまたはリソースベースのポリシーで、アカウント内のエンティティ (ユーザーまたはロール) に付与するアクセス許可が制限されますが、アクセス許可は付与されません。
- [アクセスコントロールリスト \(ACL\)](#) – ACL を使用して、ACL がアタッチされているリソースにアクセスすることができる他のアカウントのプリンシパルを制御します。ACL は、リソースベースのポリシーと似ていますが、JSON ポリシードキュメント構造を使用しない唯一のポリシータイプです。ACL は、指定されたプリンシパルにアクセス許可を付与するクロスアカウントのアクセス許可ポリシーです。ACL では、同じアカウント内のエンティティにアクセス許可を付与することはできません。
- [セッションポリシー](#) – AWS CLI または AWS API を使用して、ロールまたはフェデレーティッドユーザーを引き受ける場合に高度なセッションポリシーを渡します。セッションポリシーでは、ロールまたはユーザーのアイデンティティベースのポリシーでセッションに付与するアクセス許可を制限します。セッションポリシーでは、作成したセッションのアクセス許可が制限されますが、アクセス許可は付与されません。詳細については、「[セッションポリシー](#)」を参照してください。

## アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、ユーザーのグループ、ロール) が実行できるアクション、リソース、および条件を制御する JSON アクセス許可ポリシードキュメントです。アイデンティティベースのポリシーはさらに次のように分類できます。

- 管理ポリシー - AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンのアイデンティティベースのポリシーです。管理ポリシーには 2 種類あります。
  - AWS 管理ポリシー – AWS が作成および管理する管理ポリシー。
  - カスタマー管理ポリシー - AWS アカウント で作成および管理する管理ポリシー。カスタマー管理ポリシーでは、AWS 管理ポリシーに比べて、より正確にポリシーを管理できます。
- インラインポリシー - 単一のユーザー、グループ、ロールに直接追加するポリシー。インラインポリシーは、ポリシーとIDの間の厳密な1対1の関係を維持します。これらは、ID を削除すると削除されます。

管理ポリシーとインラインポリシーを使い分ける方法については、「[管理ポリシーとインラインポリシーの比較](#)」を参照してください。

## リソースベースのポリシー

リソースベースのポリシーは、Amazon S3 バケットなどのリソースにアタッチする JSON ポリシードキュメントです。これらのポリシーでは、そのリソースに対して特定のアクションを実行するために指定されたプリンシパルのアクセス許可を付与するとともに、このアクセス許可が適用される条件を定義します。リソースベースのポリシーはインラインポリシーです。マネージド型のリソースベースのポリシーはありません。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルおよびリソースが別の AWS アカウント である場合は、アイデンティティベースのポリシーを使用して、リソースへのアクセス権をプリンシパルに付与する必要があります。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。クロスサービスアクセスを許可する手順については、[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)。

IAM サービスは、ロールの信頼ポリシーと呼ばれるリソースベースのポリシーのタイプを 1 つのみサポートします。これが、IAM ロールにアタッチされています。IAM ロールは、リソースベースのポリシーをサポートするアイデンティティかつリソースです。そのため、信頼ポリシーとアイデンティティベースのポリシーのいずれも IAM ロールにアタッチする必要があります。信頼ポリシーでは、ロールを引き受けることができるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーテッドユーザー) を定義します。IAM ロールと、他のリソースベースのポリシーとの違いについては、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

リソースベースのポリシーをサポートするその他のサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。リソースベースのポリシーの詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

## IAM アクセス許可の境界

アクセス許可の境界は、アイデンティティベースのポリシーが IAM エンティティに付与できるアクセス許可の上限を設定する高度な機能です。エンティティのアクセス許可の境界を設定した場合、工

ンティティは、アイデンティティベースのポリシーとそのアクセス許可の境界の両方で許可されているアクセス許可のみ実行できます。プリンシパルとしてユーザーまたはロールを指定するリソースベースのポリシーは、アクセス許可の境界では制限されません。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。アクセス許可の境界の詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください

## サービスコントロールポリシー (SCP)

AWS Organizations は、ビジネスが所有する複数の AWS アカウント をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、組織または組織単位 (OU) のアクセス許可の上限を指定する JSON ポリシーです。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。

Organizations と SCP の詳細については、[AWS Organizations ユーザーガイド](#)の「SCP の仕組み」を参照してください。

## アクセスコントロールリスト (ACL)

アクセスコントロールポリシー (ACL) は、リソースにアクセスできる別のアカウントのプリンシパルを制御できるサービスポリシーです。ACL を使用して、同じアカウント内のプリンシパルのアクセス権を制御することはできません。ACL は、リソースベースのポリシーと似ていますが、JSON ポリシードキュメント形式を使用しない唯一のポリシータイプです。Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、[Amazon Simple Storage Service デベロッパーガイド](#)の「アクセスコントロールリスト (ACL) の概要」を参照してください。

## セッションポリシー

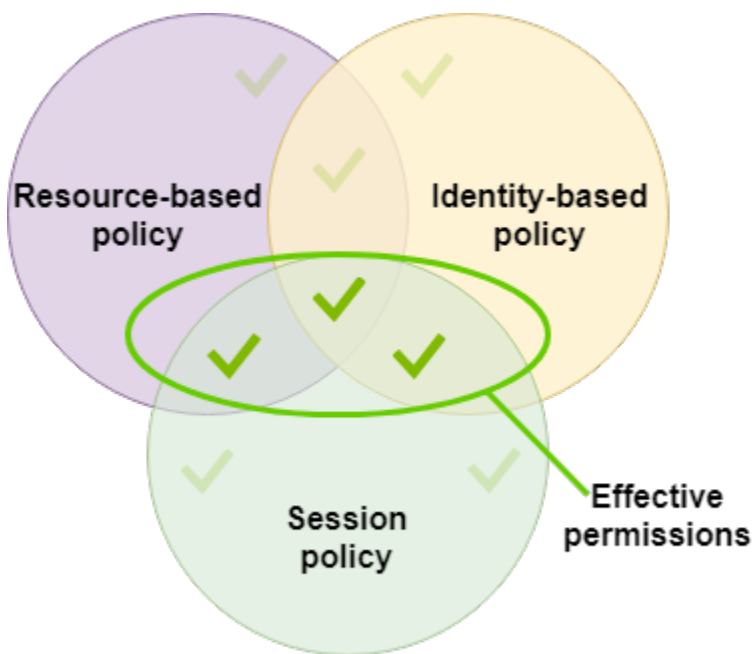
セッションポリシーは、ロールまたはフェデレーテッドユーザーの一時セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。セッションのアクセス許可は、セッションの作成に使用する IAM エンティティ (ユーザーまたはロール) のアイデンティティベースのポリシーとセッションポリシーの共通部分です。また、リソースベースのポリシーから権限が派生する場合があります。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。

API の AssumeRole、AssumeRoleWithSAML、または AssumeRoleWithWebIdentity オペレーションを使用して、ロールセッションを作成し、プログラムでセッションポリシーを渡すことができます。Policy パラメータを使用して単一の JSON インラインセッションポリシードキュメントを

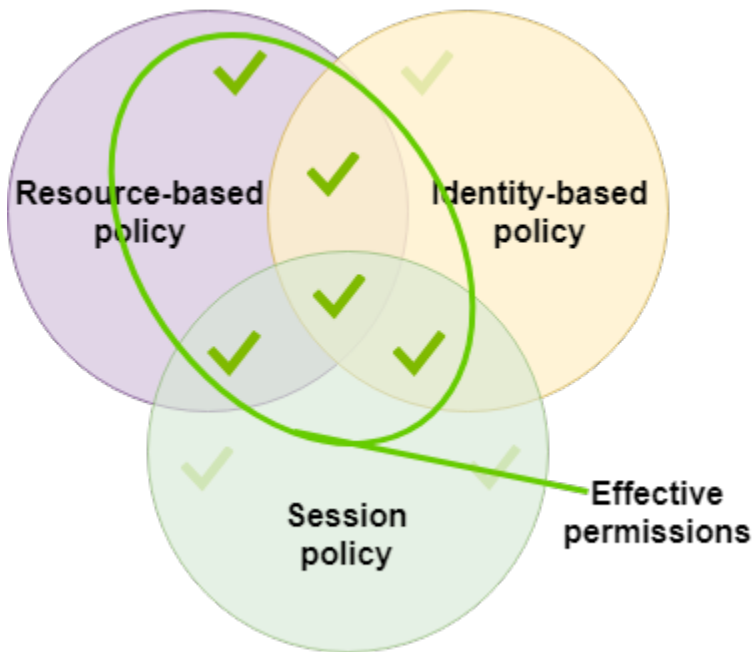
渡すことができます。PolicyArns パラメータを使用して、最大 10 個の管理セッションポリシーを指定できます。ロールセッションに関する詳細については、「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

フェデレーションユーザーのセッションを作成する場合は、IAM ユーザーのアクセスキーを使用して、API の `GetFederationToken` オペレーションをプログラムで呼び出します。また、セッションポリシーも渡す必要があります。結果として得られるセッションのアクセス許可は、ID ベースのポリシーとセッションポリシーの共通部分です。フェデレートッドユーザーセッションの作成に関する詳細については、「[GetFederationToken — カスタム ID ブローカーを介したフェデレーション](#)」を参照してください。

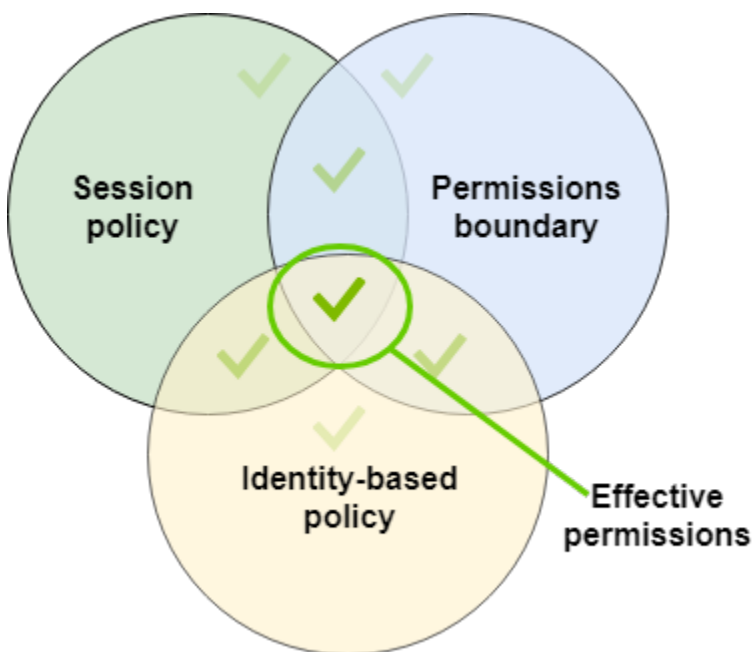
リソースベースのポリシーでは、プリンシパルとしてユーザーまたはロールの ARN を指定できます。その場合、セッションが作成される前に、リソースベースのポリシーのアクセス許可がロールまたはユーザーのアイデンティティベースのポリシーに追加されます。このセッションポリシーでは、リソースベースのポリシーおよびアイデンティティベースのポリシーによって付与されるアクセス許可の合計を制限します。結果として得られるセッションのアクセス許可は、セッションポリシーとリソースベースのポリシーの共通部分に加えて、セッションポリシーと ID ベースのポリシーの共通部分です。



リソースベースのポリシーでは、プリンシパルとしてセッションの ARN を指定できます。その場合、リソースベースのポリシーのアクセス権限がセッションの作成後に追加されます。リソースベースのポリシーのアクセス許可は、セッションポリシーで制限されません。結果として得られるセッションには、リソースベースのポリシーのすべてのアクセス許可に加えて、アイデンティティベースのポリシーとセッションポリシーの共通部分があります。



アクセス許可の境界で、セッションの作成に使用するユーザーまたはロールのアクセス許可の上限が設定されます。その場合、結果として得られるセッションのアクセス許可は、セッションポリシー、アクセス許可の境界、およびアイデンティティベースのポリシーの共通部分です。ただし、アクセス許可の境界では、結果として得られるセッションの ARN を指定するリソースベースのポリシーによって付与されたアクセス許可は制限されません。



## ポリシーとルートユーザー

AWS アカウントのルートユーザーに影響するポリシータイプは限定されます。ルートユーザーにはアイデンティティベースのポリシーをアタッチできません。また、ルートユーザーにはアクセス許可の境界を設定できません。ただし、ルートユーザーはリソースベースのポリシーまたは ACL でプリンシパルとして指定できます。ルートユーザーは引き続きアカウントのメンバーです。そのアカウントが AWS Organizations の組織のメンバーの場合、ルートユーザーからアカウントの SCP の影響を受けます。

## JSON ポリシー概要

大半のポリシーは JSON ドキュメントとして AWS に保存されます。アイデンティティベースのポリシーおよび境界のアクセス許可設定に使用されるポリシーは、ユーザーまたはロールにアタッチする JSON ポリシードキュメントです。リソースベースのポリシーは、リソースにアタッチする JSON ポリシードキュメントです。SCP は、AWS Organizations 組織単位 (OU) にアタッチする (構文が制限された) JSON ポリシードキュメントです。ACL もリソースにアタッチしますが、別の構文を使用する必要があります。セッションポリシーは、ロールまたはフェデレーティッドユーザーセッションを引き受けるときに指定した JSON ポリシーです。

JSON 構文を理解する必要はありません。AWS Management Console でビジュアルエディタを使用すると、JSON を一切使用することなく、カスタマー管理ポリシーを作成および編集できます。ただし、グループあるいは複雑なポリシーに対してインラインポリシーを使用する場合は、コンソールで JSON エディタを使用してポリシーを作成および編集する必要があります。ビジュアルエディタの詳細については、「[IAM ポリシーの作成](#)」および「[IAM ポリシーの編集](#)」を参照してください。

JSON ポリシーを作成または編集するときに、IAM はポリシー検証を実行し、効果的なポリシーを作成するのに役立ちます。IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックを提供します。ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。IAM Access Analyzer のポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」IAM Access Analyzer ポリシーの検証を参照してください。

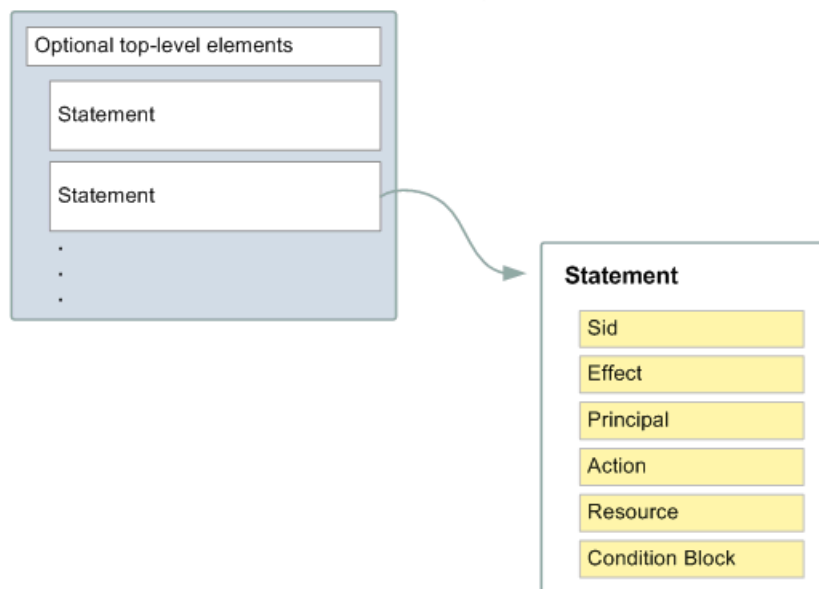
## JSON ポリシードキュメント構造

次の図に示すように、JSON ポリシードキュメントは以下の要素で構成されます。

- ドキュメントの最上部に記載されるポリシー全体の情報 (任意)
- 1 つ以上の個別のステートメント



各ステートメントには、1つのアクセス許可に関する情報が含まれています。ポリシーに複数のステートメントが含まれている場合、AWS は評価時にステートメント全体に論理 OR を適用します。複数のポリシーが1つのリクエストに適用される場合、AWS は評価時にすべてのポリシーに論理 OR を適用します。



ステートメントの情報は、一連の要素の中に含まれています。

- Version – 使用するポリシー言語のバージョンを指定します。最新 2012-10-17 バージョンの使用をお勧めします。詳細については、「[IAM JSON ポリシー要素Version](#)」を参照してください。
- Statement – このポリシーのメイン要素であり、以下の要素のコンテナになります。ポリシーには、複数のステートメントを含めることができます。
- Sid (オプション) – 複数のステートメントを区別するための任意のステートメント ID が含まれます。
- Effect – Allow または Deny を使用してポリシーで付与または拒否するアクセス許可を指定します。
- Principal (一部の状況でのみ必須) リソースベースのポリシーを作成する場合は、アクセスを許可または拒否するアカウント、ユーザー、ロール、またはフェデレーテッドユーザーを指定する必要があります。ユーザーまたはロールにアタッチする IAM アクセス許可ポリシーを作成する場合は、この要素を含めることはできません。プリンシパルは、そのユーザーまたはロールを暗黙に示しています。
- Action – ポリシーで許可または拒否するアクションのリストが含まれます。

- Resource (一部の状況でのみ必須) IAM アクセス許可ポリシーを作成する場合は、アクションが適用されるリソースのリストを指定する必要があります。リソースベースのポリシーを作成する場合は、この要素はオプションです。この要素を含めない場合、アクションが適用されるリソースは、ポリシーがアタッチされているリソースです。
- Condition (オプション) ポリシーでアクセス許可を付与する状況を指定します。

上記およびさらに詳細なポリシー要素の詳細については、「[IAM JSON ポリシー要素のリファレンス](#)」を参照してください。

## 複数のステートメントと複数のポリシー

エンティティ (ユーザーまたはロール) に対して複数のアクセス許可を定義する場合は、1つのポリシーで複数のステートメントを使用することができます。また、複数のポリシーをアタッチすることもできます。1つのステートメントで複数のアクセス許可を定義すると、ポリシーから期待するアクセス許可が付与されない場合があります。ポリシーをリソースタイプ別に分割することをお勧めします。

[ポリシーのサイズが制限されている](#) ために、複雑なアクセス許可には複数のポリシーが必要になる場合があります。機能別にアクセス許可をグループ化して別個のカスタマー管理ポリシーを作成することも有効です。たとえば、IAM ユーザー管理用、自己管理用、S3 バケット管理用に3つのポリシーを作成するとします。AWS では、複数のステートメントや複数のポリシーの組み合わせにかかわらず、ポリシーを同じ方法で[評価](#)します。

たとえば、次のポリシーには3つのステートメントがあります。ステートメント別に異なるアクセス許可セットが1つのアカウント内に定義されます。各ステートメントでは以下が定義されます。

- 最初のステートメントは、Sid (ステートメント ID) が FirstStatement であり、ポリシーがアタッチされたユーザーに対して各自のパスワードを変更することを許可します。このステートメントの Resource 要素は「すべてのリソース」を意味する「\*」です。しかし、実際に API オペレーション ChangePassword (CLI コマンド change-password に対応) が影響するのはリクエストを行うユーザーのパスワードのみです。
- 2つ目のステートメントでは、ユーザーは AWS アカウント のすべての Amazon S3 バケットを一覧表示できます。このステートメントの Resource 要素は「すべてのリソース」を意味する "\*" です。ただし、ポリシーでは他のアカウントのリソースへのアクセスが許可されていないため、ユーザーは自分の AWS アカウント のバケットのみ一覧表示できます。
- 3番目のステートメントでは、confidential-data というバケット内の任意のオブジェクトを表示および取得することをユーザーに許可しますが、ユーザーが多要素認証 (MFA) で認証することが条件です。このポリシーの Condition 要素で MFA 認証が強制されます。



ポリシーステートメントに Condition 要素が含まれている場合、このステートメントは Condition 要素が true に評価されたときにのみ有効になります。この例では、ユーザーが MFA 認証された場合に限り、Condition 要素が true に評価されます。ユーザーが MFA 認証されていない場合、この Condition は false に評価されます。この場合、このポリシーの 3 番目のステートメントは適用されず、ユーザーは confidential-data バケットにアクセスできません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::confidential-data",
        "arn:aws:s3:::confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

## JSON ポリシー構文の例

次のIDベースのポリシーにより、暗黙のプリンシパルは `example_bucket` という名前の単一の Amazon S3 バケットをリストできます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

次のリソースベースのポリシーは Amazon S3 バケットにアタッチできます。このポリシーでは、バケット `mybucket` に対して Amazon S3 のすべてのアクションを実行することを、特定の AWS アカウントのメンバーに許可します。バケットやバケット内のオブジェクトに対して任意のアクションを実行できます。(このポリシーではアカウントに対してのみ信頼が付与されているため、指定された Amazon S3 アクションを実行するには、アクションへのアクセス許可をアカウント内の個々のユーザーに付与する必要があります。)

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::account-id:root"]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }]
}
```

一般的なシナリオのポリシー例については、[「IAM アイデンティティベースのポリシーの例」](#)を参照してください。

## 最小限の特権を認める。

IAM ポリシーを作成する場合、最小限のアクセス権を付与するという標準的なセキュリティアドバイスに従うか、タスクの実行に必要なアクセス許可のみ付与します。ユーザー (およびロール) が何をする必要があるのかを決定してから、それらのタスクのみの実行を許可するポリシーを作成します。

最小限の許可からスタートし、必要に応じて追加の許可を付与します。この方法は、寛容過ぎる許可から始めて、後から厳しくしようとするよりも安全です。

最小限のアクセス許可の代わりに、[AWS 管理対象ポリシー](#)またはワイルドカード \* アクセス許可を持つポリシーを使用して、ポリシーの使用を開始できます。プリンシパルにジョブに必要な以上のアクセス許可を付与すると、セキュリティ上のリスクを考慮してください。これらのプリンシパルをモニタリングして、使用しているアクセス許可を確認します。次に、最小限の特権ポリシーを記述します。

IAM には、付与するアクセス許可を絞り込むのに役立つオプションがいくつか用意されています。

- アクセスレベルのグループ化 – ポリシーが付与するアクセスレベルを把握できます。[ポリシーアクション](#)は、List、Read、Write、Permissions management、または Tagging として分類されています。たとえば、List や Read アクセスレベルからアクションを選択し、ユーザーに読み取り専用のアクセスレベルを付与することができます。アクセスレベルの権限を理解するためにポリシーの概要を使用するには「[ポリシー概要内のアクセスレベルの概要](#)」をご覧ください。
- ポリシーの検証 – JSON ポリシーを作成および編集するときに、IAM Access Analyzer を使用してポリシーの検証を実行できます。既存のすべてのポリシーを確認および検証することをお勧めします。IAM Access Analyzer では、ポリシーを検証するために 100 を超えるポリシーチェックが提供されます。ポリシー内のステートメントで、過度に許容されるアクセスを許可すると、セキュリティ警告が生成されます。最小限の特権の付与に向けて作業するときに、セキュリティ警告によって提供される実用的な推奨事項を使用できます。IAM Access Analyzer で提供されるポリシーチェックの詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。
- アクセスアクティビティに基づくポリシーの生成 – 付与するアクセス権限を調整するために、IAM エンティティ (ユーザーまたはロール) のアクセスアクティビティに基づく IAM ポリシーを生成できます。IAM Access Analyzer は AWS CloudTrail ログを確認し、指定した日付範囲内のロールが使用したアクセス許可を含むポリシーテンプレートを生成します。テンプレートを使用して、きめ細かなアクセス権限で管理ポリシーを作成し、それを IAM エンティティにアタッチできます。これにより、特定のユースケースでロールが AWS リソースとインタラクションするために必要なアクセス権限のみを付与します。詳細については、「[アクセスアクティビティに基づいてポリシーを生成する](#)」を参照してください。

- 最終アクセス情報を使用 — 最低限のアクセス許可で役立つもう 1 つの機能は、最終アクセス情報です。IAM ユーザー、グループ、ロール、ポリシーのこの情報は、IAM コンソールの詳細ページの [アクセスアドバイザー] に表示されます。最終アクセス情報には、Amazon EC2、IAM、Lambda、Amazon S3 など、一部のサービスで最後にアクセスされたアクションに関する情報も含まれます。AWS Organizations 管理アカウントの認証情報を使用してサインインすると、IAM コンソールの [AWS Organizations] セクションでサービスの最終アクセス情報を表示できます。AWS CLI または AWS API を使用して、IAM または Organizations でエンティティまたはポリシーの最終アクセス情報のレポートを取得するために使用することもできます。この情報を使用して不要なアクセス許可を識別し、最小権限の原則により良く準拠するよう IAM または Organizations ポリシーを改善することができます。詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。
- AWS CloudTrail のアカウントイベントを確認 - アクセス許可をさらに削減するには、AWS CloudTrail のイベント履歴でアカウントのイベントを表示できます。CloudTrail のイベントログには、ポリシーのアクセス許可を減らすために使用できる詳細なイベント情報が含まれます。ログには、IAM エンティティが必要とするアクションとリソースのみが含まれます。詳細については、AWS CloudTrail ユーザーガイドの [CloudTrail イベント履歴でのイベントの表示](#) を参照してください。

詳細については、以下の各サービスのポリシートピックを参照してください。サービス固有のリソースに対するポリシーの書き方の例を挙げています。

- Amazon DynamoDB 開発者ガイドの [Amazon DynamoDB に対する認証とアクセスコントロール](#)
- Amazon Simple Storage Service ユーザーガイドの [バケットポリシーとユーザーポリシーの使用](#)
- [Amazon Simple Storage Service ユーザーガイド](#) の「アクセスコントロールリスト (ACL) の概要」

## 管理ポリシーとインラインポリシー

IAM ID のアクセス許可を設定する必要がある場合は、AWS 管理ポリシー、カスタマー管理ポリシー、またはインラインポリシーを使用するかどうかを決定します。以下のトピックでは、各種のアイデンティティベースのポリシーとそれぞれの用途について詳しく説明します。

### トピック

- [AWS マネージドポリシー](#)
- [カスタマー管理ポリシー](#)
- [インラインポリシー](#)

- [管理ポリシーとインラインポリシーの比較](#)
- [管理ポリシーの開始方法](#)
- [インラインポリシーを管理ポリシーに変換する](#)
- [非推奨の AWS 管理ポリシー](#)

## AWS マネージドポリシー

AWS 管理ポリシーは、AWS が作成および管理するスタンドアロンポリシーです。スタンドアロンポリシーとは、ポリシー名を含む独自の Amazon リソースネーム (ARN) の付いたポリシーです。たとえば、arn:aws:iam::aws:policy/IAMReadOnlyAccess は AWS 管理ポリシーです。ARN の詳細については、[IAM ARN](#)を参照してください。AWS のサービスに対する AWS マネージドポリシーのリストについては、「[AWS マネージドポリシー](#)」を参照してください。

AWS 管理ポリシーは、ユーザー、グループおよびロールに適切な権限を割り当てるのに便利です。ポリシーを自分で作成するよりも簡単で、多くの一般的なユースケースに対応したアクセス許可が含まれています。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。AWS は、AWS 管理ポリシーで定義されているアクセス許可を不定期に更新します。AWS によるこの更新は、ポリシーがアタッチされているすべてのプリンシパルエンティティ (ユーザー、グループ、およびロール) に影響を与えます。AWS は、新しい AWS サービスが開始されたときや既存のサービスで新しい API コールが利用可能になったときに、AWS 管理ポリシーを更新する可能性が最も高くなります。たとえば、ReadOnlyAccess という AWS 管理ポリシーでは、すべての AWS サービスおよびリソースへの読み取り専用アクセスが許可されます。AWS は新しいサービスを開始すると、AWS により ReadOnlyAccess ポリシーを更新して、新しいサービスに対する読み取り専用アクセス許可を追加します。更新されたアクセス権限は、ポリシーがアタッチされているすべてのプリンシパルエンティティに適用されます。

フルアクセスの AWS 管理ポリシーでは、フルアクセスをサービスに付与して、サービス管理者のアクセス許可を定義します。

- [AmazonDynamoDBFullAccess](#)
- [IAMFullAccess](#)

AWS パワーユーザー管理ポリシーは、AWS サービスとリソースへのフルアクセスを提供しますが、ユーザーとグループの管理は許可しません。

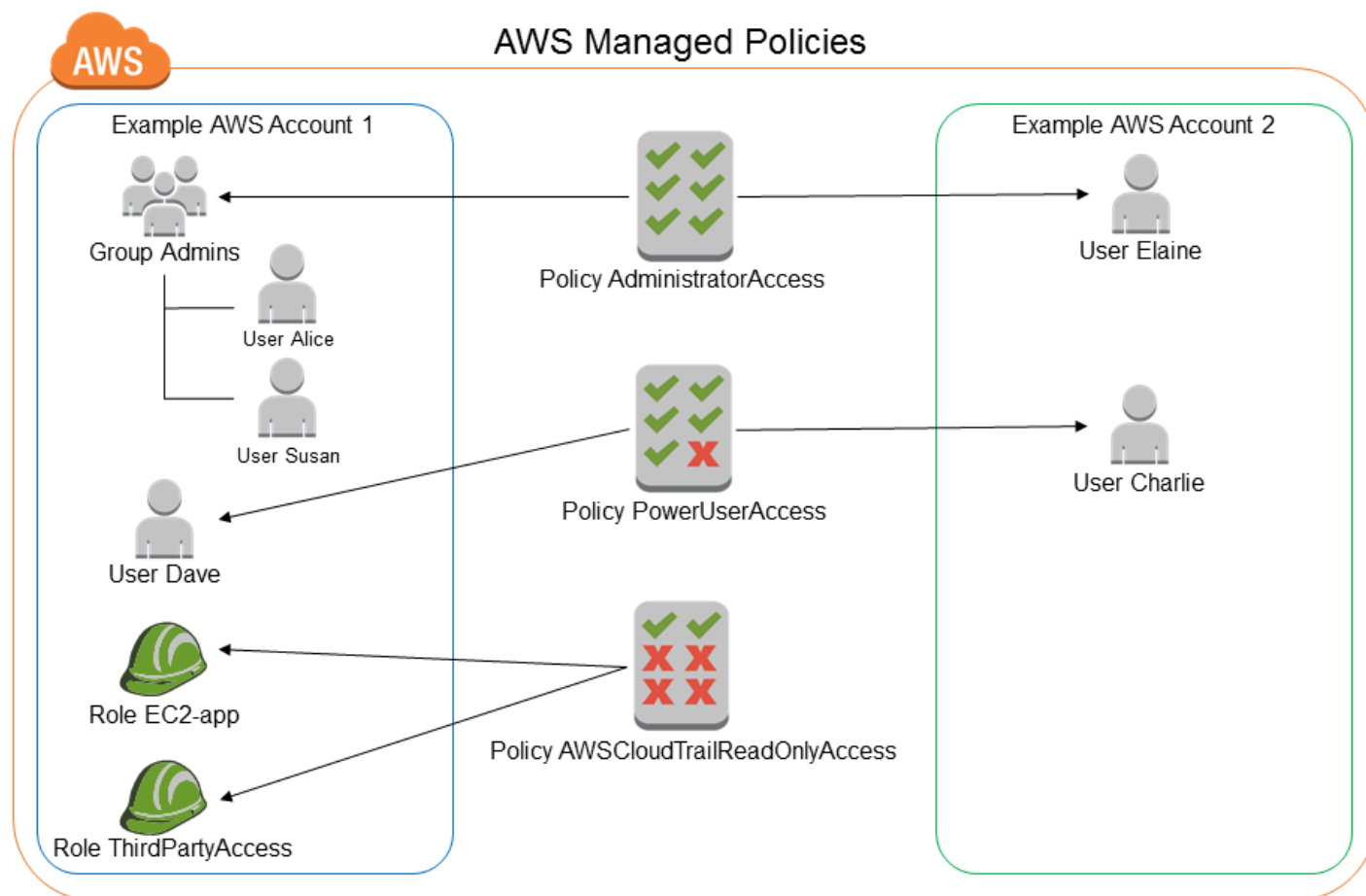
- [AWSCodeCommitPowerUser](#)
- [AWSKeyManagementServicePowerUser](#)

部分的なアクセスの AWS 管理ポリシーは、[アクセス許可管理](#) アクセスレベルの許可を与えずに、特定のレベルの AWS サービスに対するアクセスを提供します。

- [AmazonMobileAnalyticsWriteOnlyAccess](#)
- [AmazonEC2ReadOnlyAccess](#)

AWS 管理ポリシーの中でも特に有用なカテゴリーは、ジョブ機能用に設計されているものです。これらのポリシーは、IT 業界で一般的に使用されているジョブ機能と密接に連携し、これらのジョブ機能に対するアクセス許可の付与を容易に与えることができます。ジョブ機能ポリシーを使用する重要な利点としては、新しいサービスとして AWS によってこれらの保守や更新が行われ、API オペレーションが導入されることが 1 つ挙げられます。たとえば、[AdministratorAccess](#) ジョブ関数は、AWS の各サービスおよびリソースへのフルアクセスを許可し、アクセス許可の委任が可能です。このポリシーは、アカウント管理者のみに使用することをお勧めします。IAM と Organizations への制限されたアクセスを除き、すべてのサービスへのフルアクセスを必要とするパワーユーザーの場合は、[PowerUserAccess](#) ジョブ関数を使用します。ジョブ機能ポリシーのリストと説明については、[AWSジョブ機能の 管理ポリシー](#) を参照してください。

次の図は AWS 管理ポリシーを示しています。図には、3 つの AWS 管理ポリシー (AdministratorAccess、PowerUserAccess、AWS CloudTrailReadOnlyAccess) が示されています。1 つの AWS 管理ポリシーを複数の AWS アカウント のプリンシパルエンティティに、また、1 つの AWS アカウント の複数のプリンシパルエンティティにアタッチできることに注意してください。



## カスタマー管理ポリシー

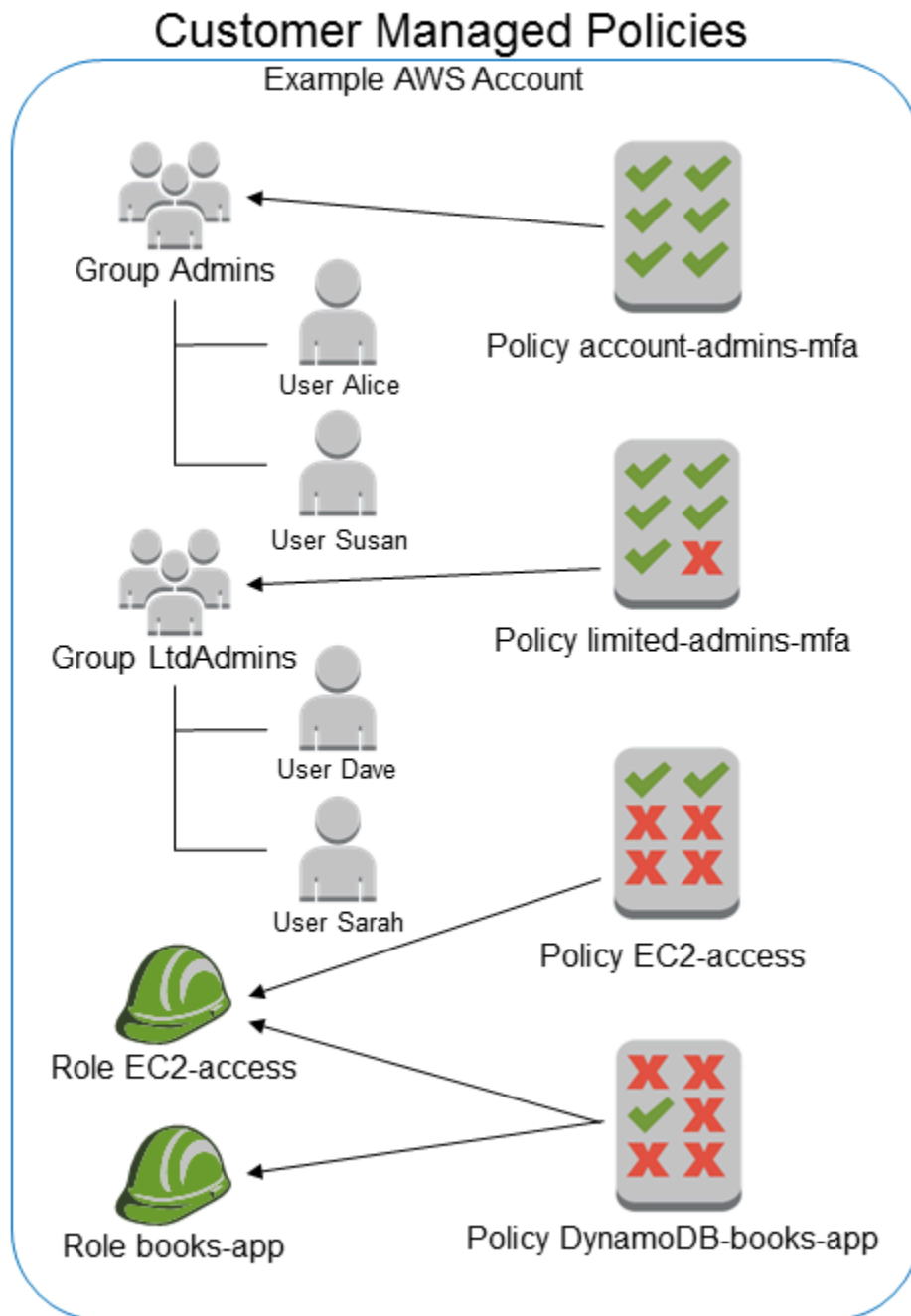
プリンシパルエンティティ (ユーザー、グループ、ロール) にアタッチできる単独のポリシーを独自の AWS アカウント に作成することができます。これらのカスタマー管理ポリシーは、特定のユースケースに合わせて作成し、何度でも変更および更新が可能です。AWS 管理ポリシーのように、プリンシパルエンティティにアタッチすると、ポリシーで定義されたアクセス権限がエンティティに付与されます。ポリシーに含まれるアクセス許可が更新されると、その変更はポリシーがアタッチされているすべてのプリンシパルエンティティに適用されます。

お客様が管理するポリシーを作成する優れた方法は、既存の AWS 管理ポリシーをコピーして開始することです。この方法で、最初の段階でポリシーが正しいことがわかっている場合、ご使用の環境に合わせて必要なカスタマイズを行うことができます。

次の図は、カスタマー管理ポリシーを示しています。各ポリシーは、IAM 内の独自の [Amazon リソースネーム \(ARN\)](#) (ポリシー名を含む) を持つエンティティです。同じポリシーを複数のプリンシパルエンティティにアタッチできること (たとえば、同じ DynamoDB-books-app ポリシーが異なる 2 つの IAM ロールにアタッチされていること) に注目してください。



詳細については、「[IAM ポリシーの作成](#)」を参照してください。

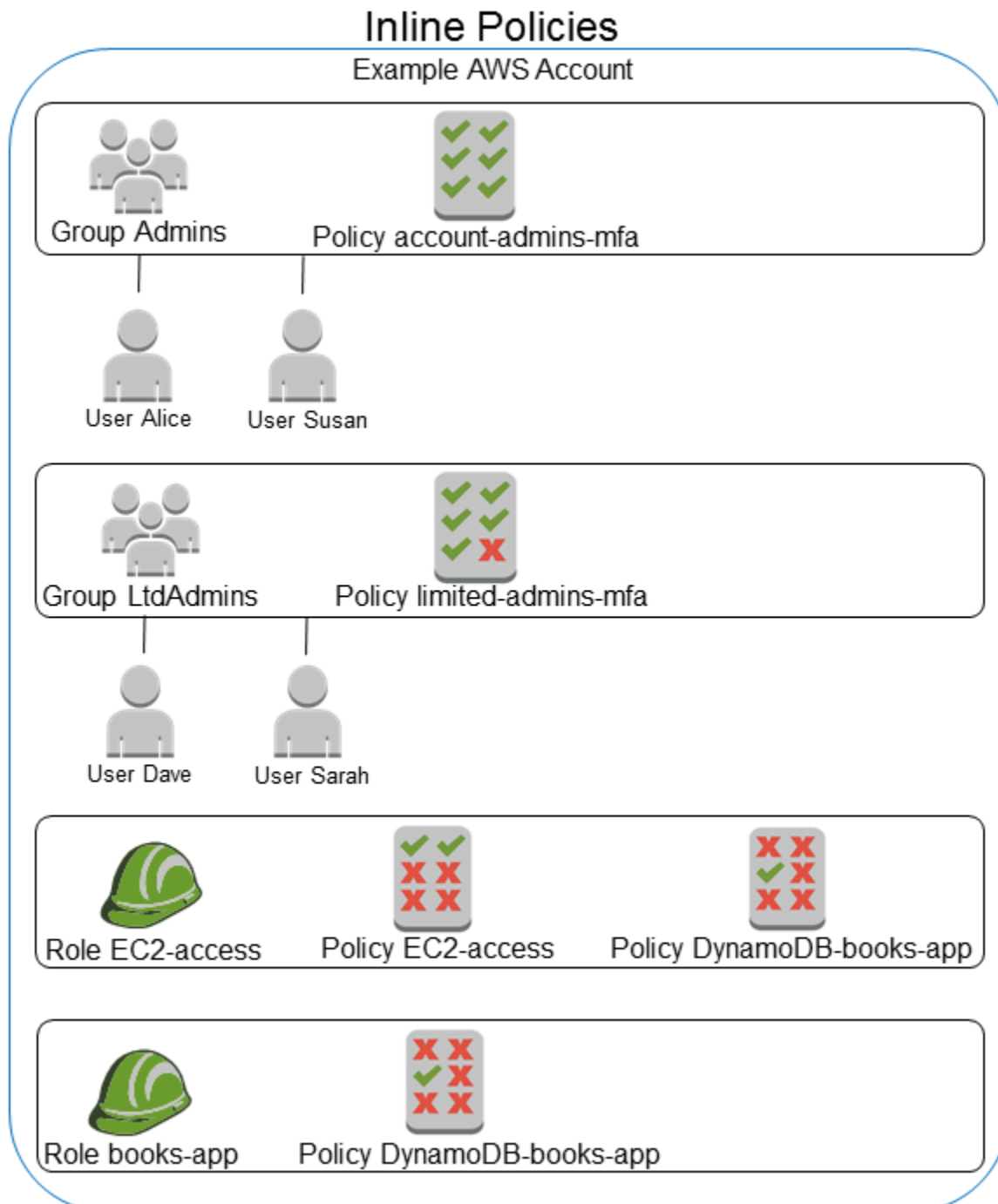


## インラインポリシー

インラインポリシーは、1つの IAM アイデンティティ (ユーザー、グループ、またはロール) に埋め込まれたポリシーです。インラインポリシーは、ポリシーとIDの間の厳密な1対1の関係を維持します。これらは、ID を削除すると削除されます。ID の作成時、またはそれ以降で、ポリシーを作成して ID に埋め込むことができます。ポリシーが複数のエンティティに適用される可能性がある場合は、管理ポリシーを使用することをお勧めします。



次の図はインラインポリシーを示しています。各ポリシーは本質的にユーザー、グループ、またはロールの一部です。2つのロールが同じポリシー (DynamoDB-books-app ポリシー) を含んでいますが、1つのポリシーを共有していないことに注意してください。各ロールには、独自のポリシーのコピーがあります。



## 管理ポリシーとインラインポリシーの比較

管理ポリシーとインラインポリシーのどちらを選ぶかは、ユースケースを考慮して決定します。通常、インラインポリシーではなく、管理ポリシーを使用することをお勧めします。

### Note

管理ポリシーとインラインポリシーの両方を併用して、プリンシパルエンティティに共通および固有のアクセス許可を定義することができます。

管理ポリシーは次の機能を備えています。

### 再利用可能性

1つの管理ポリシーを複数のプリンシパルエンティティ (ユーザー、グループ、ロール) にアタッチすることができます。AWS アカウント にとって有用なアクセス許可を定義するポリシーのライブラリを作成し、これらのポリシーを必要に応じてプリンシパルエンティティにアタッチできます。

### 一元化された変更管理

管理ポリシーを変更すると、変更はポリシーがアタッチされているすべてのプリンシパルエンティティに適用されます。例えば、新しい AWS API のアクセス許可を追加する場合、カスタマー管理ポリシーを更新するか、AWS 管理ポリシーを関連付けてアクセス許可を追加します。AWS 管理ポリシーを使用している場合は、AWS がポリシーを更新します。管理ポリシーが更新されると、変更はその管理ポリシーがアタッチされているすべてのプリンシパルエンティティに適用されます。一方、インラインポリシーを変更するには、そのインラインポリシーを含む各アイデンティティを個別に編集する必要があります。たとえば、グループとロールの両方に同じインラインポリシーがある場合、両方のプリンシパルエンティティを個別に編集して、ポリシーを変更する必要があります。

### バージョニングとロールバック

カスタマー管理ポリシーを変更しても、変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。IAM は、最大 5 つのバージョンのカスタマー管理ポリシーを保存します。ポリシーバージョンを使用し、必要に応じてポリシーを以前のバージョンに戻すことができます。

**Note**

ポリシーのバージョンは、Version ポリシーの要素とは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。ポリシーのバージョンの詳細については、「[the section called “IAM ポリシーのバージョンング”](#)」を参照してください。Version ポリシー要素の詳細については、「[IAM JSON ポリシー要素Version](#)」を参照してください。

## アクセス許可管理の委任

ポリシーで定義されたアクセス許可を制御しながら、AWS アカウント のユーザーにポリシーのアタッチとデタッチを許可できます。これを行うには、一部のユーザーを完全な権限を持つ管理者 (ポリシーの作成、更新、削除が可能な管理者) として指定します。次に、権限が制限された管理者として他のユーザーを指定できます。他のプリンシパルエンティティにポリシーをアタッチできる限定管理者ですが、アタッチを許可したポリシーに限ります。

アクセス許可の委任の詳細については、「[ポリシーへのアクセスの制御](#)」を参照してください。

## より大きいポリシー文字制限

管理ポリシーの最大文字サイズ制限は、インラインポリシーの文字制限よりも大きいです。インラインポリシーの文字サイズの上限に達した場合は、さらに IAM グループを作成し、管理ポリシーをそのグループにアタッチできます。

クォータと制限の詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。

## AWS 管理ポリシーの自動更新

AWS は、AWS 管理ポリシーを維持し、新しい AWS サービスのアクセス許可を追加するなど、必要に応じて更新することで、お客様が変更することなく、サービスを提供することができます。更新は、AWS 管理ポリシーをアタッチしているプリンシパルエンティティに自動的に適用されます。

## インラインポリシーの使用

インラインポリシーは、ポリシーとそれが適用されている ID との厳密な 1 対 1 の関係を維持する必要がある場合に便利です。たとえば、ポリシー内のアクセス許可が意図したアイデンティティ以外のアイデンティティに間違って割り当てられないようにする必要がある場合などです。インラインポリシーを使用すると、ポリシーのアクセス許可が間違っただアイデンティティにアタッチされることはあ

りません。また、AWS Management Console を使用してアイデンティティを削除すると、そのアイデンティティに組み込まれたポリシーもプリンシパルエンティティの一部であるため、同様に削除されます。

## 管理ポリシーの開始方法

[最小権限を付与する](#) ポリシーを使用するか、タスクの実行に必要なアクセス許可のみを付与することをお勧めします。最小特権を付与する最も安全な方法は、チームに必要な権限のみを使用してカスタマー管理ポリシーを作成することです。必要に応じて、チームがより多くの権限を要求できるようにプロセスを作成する必要があります。チームに必要な許可のみを提供する [IAM カスタマー管理ポリシー](#) を作成するには、時間と専門知識が必要です。

IAM ID (ユーザー、ユーザーのグループ、およびロール) にアクセス許可を追加するために、[AWS マネージドポリシー](#) を使用できます。AWS 管理ポリシーは、最小特権のアクセス許可を付与しません。プリンシパルにジョブに必要な以上のアクセス許可を付与すると、セキュリティ上のリスクを考慮する必要があります。

ジョブ機能を含む AWS 管理ポリシーを任意の IAM ID にアタッチできます。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

最小特権のアクセス許可に切り替えるには、AWS Identity and Access Management Access Analyzer を実行して、AWS 管理ポリシーでプリンシパルをモニタリングします。どのアクセス許可を使用しているかを学習したら、チームに必要なアクセス許可のみを持つカスタマー管理ポリシーを作成または生成することができます。これは安全性が低くなりますが、チームが AWS をどのように使用しているかを学習するにつれて柔軟性が高まります。詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

AWS 管理ポリシーは、多くの一般的なユースケースでアクセス許可を提供できるように設計されています。特定のジョブ機能用に設計された AWS 管理ポリシーの詳細については、[AWS ジョブ機能の管理ポリシー](#) を参照してください。

AWS 管理ポリシーの一覧については、「[AWS 管理ポリシーリファレンスガイド](#)」を参照してください。

## インラインポリシーを管理ポリシーに変換する

アカウントにインラインポリシーがある場合は、管理ポリシーに変換することができます。これを行うには、ポリシーを新しい管理ポリシーにコピーします。次に、インラインポリシーを持つ ID に新しいポリシーをアタッチします。次に、インラインポリシーを削除します。

## インラインポリシーを管理ポリシーに変換するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ]、[ユーザー]、または [ロール] を選択します。
3. リストで、削除するポリシーを持つユーザーグループ、ユーザー、またはロールの名前を選択します。
4. [アクセス許可] タブを選択します。
5. ユーザーグループで、削除するインラインポリシーの名前を選択します。ユーザーおよびロールで、[Show *n* more] を選択し、必要であれば、削除するインラインポリシーを展開します。
6. [コピー] を選択し、ポリシーの JSON ポリシードキュメントをコピーします。
7. ナビゲーションペインで、[ポリシー] を選択します。
8. [ポリシーの作成] を選択し、[JSON] オプションを選択します。
9. 既存のテキストを JSON ポリシーテキストに置き換え、[次へ] を選択します。
10. ポリシーの名前と説明 (省略可能) を入力し、[ポリシーを作成] を選択します。
11. ナビゲーションペインで、[ユーザーグループ]、[ユーザー]、または [ロール] を選択し、削除するポリシーを含むグループ、ユーザー、またはロールの名前を再度選択します。
12. [アクセス許可] タブを選択してから、[アクセス許可を追加] を選択します。
13. ユーザーグループで、新しいポリシーの名前の横にあるチェックボックスをオンにし、[アクセス許可の追加] に続いて [ポリシーのアタッチ] を選択します。ユーザーまたはロールで、[アクセス許可の追加] を選択します。次のページで、[A既存のポリシーを直接アタッチ] を選択し、新しいポリシーの名前の横にあるチェックボックスをオンにしたら、[次へ]、[アクセス許可を追加] の順に選択します。

ユーザーグループ、ユーザー、またはロールの [概要] ページに戻ります。

14. 削除するインラインポリシーの横にあるチェックボックスをオンにして、[削除] を選択します。

## 非推奨の AWS 管理ポリシー

アクセス許可の割り当てを簡素化するために、AWS は [管理ポリシー](#) を提供します。このポリシーは、IAM ユーザー、グループ、およびロールへのアタッチが可能な状態の事前定義されたポリシーです。

新しいサービスの導入時など、AWS で既存のポリシーに新しいアクセス許可を追加する必要が生じる場合があります。既存のポリシーに新しいアクセス権限を追加しても、機能や働きを中断または削除されることはありません。

ただし、既存のポリシーに必要な変更を適用した場合にそれらが顧客に影響するときに、AWS で新しいポリシーを作成してもかまいません。たとえば、既存のポリシーからアクセス許可を削除すると、そのアクセス許可に依存していたすべての IAM エンティティまたはアプリケーションのアクセス許可が破棄されて、重大なオペレーションを中断する可能性があります。

したがって、このような変更が必要な場合、AWS では完全に新しいポリシーを作成して必要な変更を適用し、顧客が使用できるようにします。古いポリシーは、非推奨としてマークされます。非推奨のマネージドポリシーが、IAM コンソールの[ポリシー]リストの横に警告アイコンとともに表示されます。

非推奨のポリシーには以下のような特徴があります。

- ポリシーは、現在アタッチされているすべてのユーザー、グループ、およびロールの処理を続けます。中断される処理はありません。
- ポリシーを新しいユーザー、グループ、またはロールにアタッチすることはできません。現在のエンティティからポリシーをデタッチした場合、再アタッチすることはできません。
- 現在のすべてのエンティティからポリシーをデタッチしたら、そのポリシーは表示されなくなり、いかなる方法でも使用不能となります。

ユーザー、グループ、またはロールにポリシーが必要な場合は、代わりに新しいポリシーをアタッチする必要があります。ポリシーが非推奨であるという注意を受信した場合は、すべてのユーザー、グループ、およびロールの代替ポリシーへのアタッチと、非推奨のポリシーからのデタッチをただちに計画することをお勧めします。非推奨のポリシーを使用し続けことはリスクを伴い、代替ポリシーへの切り替え以外にはリスク低減の方法はありません。

## データ境界を使用してアクセス許可のガードレールを確立する

データ境界ガードレールは、さまざまな AWS アカウントやリソースでデータを保護するための、常時オンの境界として機能します。データ境界は、IAM セキュリティのベストプラクティスに従い、[複数のアカウントにまたがるアクセス許可ガードレールを確立します](#)。このような組織全体のアクセス許可ガードレールは、既存のきめ細かいアクセスコントロールに取って代わるものではありません。その代わりに、ユーザー、ロール、リソースが定義された一連のセキュリティ標準に準拠するようにすることで、セキュリティ戦略の改善に役立つ、粗粒度のアクセスコントロールの役割を果たします。

データ境界は、信頼できる ID だけが想定されるネットワークから信頼できるリソースにアクセスできるようにするための、AWS 環境内のアクセス許可ガードレールのセットです。

- 信頼できる ID: AWS アカウントのプリンシパル (IAM ロールまたはユーザー)、およびユーザーに代わって動作する AWS サービス。
- 信頼できるリソース: AWS アカウントまたはユーザーに代わって動作する AWS サービスが所有するリソース。
- 想定されるネットワーク: オンプレミスデータセンターおよび仮想プライベートクラウド (VPC)、またはユーザーに代わって動作する AWS サービスのネットワーク。

### Note

場合によっては、信頼できるビジネスパートナーによるアクセスも含めるために、データ境界の拡張が必要になることがあります。会社および AWS のサービスの使用に固有の信頼できる ID、信頼できるリソース、想定されるネットワークの定義を作成する際には、想定されるすべてのデータアクセスパターンを考慮する必要があります。

データ境界コントロールは、情報セキュリティおよびリスク管理プログラムにおける他のセキュリティコントロールと同様に扱う必要があります。つまり、脅威分析を行ってクラウド環境内の潜在的なリスクを特定し、独自のリスク許容基準に基づいて、適切なデータ境界コントロールを選択して実装する必要があります。データ境界を実装するための反復的なリスクベースのアプローチをより効果的に理解するには、データ境界コントロールによって対処されるセキュリティリスクおよび脅威ベクトルと、セキュリティの優先事項を理解する必要があります。

## データ境界コントロール

データ境界の粗粒度のコントロールでは、[ポリシータイプ](#)および[条件キー](#)のさまざまな組み合わせを実装することで、3 つのデータ境界で 6 つの異なるセキュリティ目標を達成できます。

境界	コントロールの目的	使用	適用対象	グローバル条件コンテキストキー
アイデンティティ	信頼できる ID のみ がリソースにアクセスできる	リソースベースのポリシー	リソース	aws:PrincipalOrgID



境界	コントロールの目的	使用	適用対象	グローバル条件 コンテキスト キー
	信頼できる ID のみ がネットワークから許可される	VPC エンドポイントポリシー	ネットワーク	aws:PrincipalOrgPaths  aws:PrincipalAccount  aws:PrincipalAwsService
リソース	ID は信頼できるリソースにのみ アクセスできる	SCP	ID	aws:ResourceOrgID  aws:ResourceOrgPaths  aws:ResourceAccount
	信頼できるリソースのみに ネットワークからアクセスできる	VPC エンドポイントポリシー	ネットワーク	
ネットワーク	ID は想定されるネットワークからのみリソースに アクセスできる	SCP	ID	aws:SourceIpp  aws:SourceVpc  aws:SourceVpce  aws:ViaAWSService
	リソースには想定されるネットワークからのみ アクセスできる	リソースベースのポリシー	リソース	aws:PrincipalAwsService

データ境界は、意図しないアクセスパターンを防ぐためにデータの周囲に強固な境界を作ることと考えることができます。データ境界は広範囲にわたる意図しないアクセスを防ぐことができますが、それでもきめ細かなアクセスコントロールに関する決定を行う必要があります。データ境界を設定して



も、[最小特権](#)の実現に向けた取り組みの一環として [IAM Access Analyzer](#) などのツールを使用してアクセス許可を継続的に微調整する必要性が軽減されることはありません。

## ID 境界

ID 境界は、信頼できる ID だけがリソースにアクセスでき、信頼できる ID だけがネットワークから許可されるようにするための、粗粒度の予防的アクセスコントロールのセットです。信頼できる ID には、AWS アカウントのプリンシパル (ロールまたはユーザー)、およびユーザーに代わって動作する AWS サービスが含まれます。それ以外のすべての ID は信頼できないと見なされ、明示的な例外が付与されない限り ID 境界によって阻止されます。

以下のグローバル条件キーは ID 境界コントロールの適用に役立ちます。これらのキーを[リソースベースのポリシー](#)で使用してリソースへのアクセスを制限したり、[VPC エンドポイントポリシー](#)でネットワークへのアクセスを制限したりします。

- [aws:PrincipalOrgID](#) — この条件キーを使用して、リクエストを行う IAM プリンシパルが AWS Organizations の指定された組織に属していることを確認できます。
- [aws:PrincipalOrgPaths](#) — この条件キーを使用して、リクエストを行う IAM ユーザー、IAM ロール、フェデレーションユーザー、または AWS アカウントのルートユーザーが、AWS Organizations で指定された組織単位 (OU) に属していることを確認できます。
- [aws:PrincipalAccount](#) — この条件キーを使用して、ポリシーで指定したプリンシパルアカウントだけがリソースにアクセスできることを確認できます。
- [aws:PrincipalsAWSService](#) および [aws:SourceOrgID](#) (または [aws:SourceOrgPaths](#) および [aws:SourceAccount](#)) — これらの条件キーを使用すると、[AWS のサービスプリンシパル](#)がリソースにアクセスするときに、指定された組織、組織単位、または AWS Organizations のアカウントのリソースに代わってのみアクセスできることを確認できます。

詳細については、「[Establishing a data perimeter on AWS: Allow only trusted identities to access company data](#)」を参照してください。

## リソース境界

リソース境界は、ID が信頼できるリソースにのみアクセスし、信頼できるリソースにのみネットワークからアクセスできるようにするための、粗粒度の予防的アクセスコントロールのセットです。信頼できるリソースには、AWS アカウントまたはユーザーに代わって動作する AWS サービスが所有するリソースが含まれます。

以下のグローバル条件キーは、リソース境界コントロールの適用に役立ちます。[サービスコントロールポリシー](#)でこれらのキーを使用して、ID がアクセスできるリソースを制限したり、[VPC エンドポイントポリシー](#)でネットワークからアクセスできるリソースを制限したりします。

- [aws:ResourceOrgID](#) — この条件キーを使用して、アクセスされているリソースが、AWS Organizations で指定された組織に属していることを確認できます。
- [aws:ResourceOrgPaths](#) — この条件キーを使用して、アクセスされているリソースが、AWS Organizations で指定された組織単位 (OU) に属していることを確認できます。
- [aws:ResourceAccount](#) — この条件キーを使用して、アクセスされているリソースが、AWS Organizations で指定されたアカウントに属していることを確認できます。

場合によっては、AWS 所有リソース (組織に属さず、プリンシパルまたはユーザーに代わって動作する AWS サービスがアクセスするリソース) へのアクセスの許可が必要になることがあります。これらのシナリオの詳細については、「[Establishing a data perimeter on AWS: Allow only trusted resources from my organization](#)」を参照してください。

## ネットワーク境界

ネットワーク境界とは、ID が想定されるネットワークからのリソースにのみアクセスし、リソースには想定されるネットワークからのみアクセスできるようにするための、粗粒度の予防的アクセスコントロールのセットです。想定されるネットワークには、オンプレミスデータセンター、仮想プライベートクラウド (VPC)、およびユーザーに代わって動作する AWS サービスのネットワークが含まれます。

以下のグローバル条件キーは、ネットワーク境界コントロールの適用に役立ちます。これらのキーを[サービスコントロールポリシー](#)で使用して ID の通信元となるネットワークを制限したり、[リソーススペースのポリシー](#)でリソースアクセスを想定されるネットワークに制限したりします。

- [aws:SourceIp](#) — この条件キーを使用して、リクエスト元の IP アドレスが指定の IP 範囲内にあることを確認できます。
- [aws:SourceVpc](#) — この条件キーを使用して、リクエストが通過する VPC エンドポイントが指定の VPC に属していることを確認できます。
- [aws:SourceVpcId](#) — この条件キーを使用して、リクエストが指定された VPC エンドポイントを通過することを確認できます。
- [aws:ViaAWSService](#) — この条件キーを使用すると、AWS のサービスが[転送アクセスセッション \(FAS\)](#)を使用するプリンシパルに代わってリクエストを送信できることを確認できます。

- [aws:PrincipalsAWSService](#) — この条件キーを使用して、AWS のサービスが [AWS サービスプリンシパル](#) を使用してリソースにアクセスできることを確認できます。

ネットワークの外部からリソースにアクセスする AWS のサービスへのアクセスを許可する必要があるシナリオは他にもあります。詳細については、「[Establishing a data perimeter on AWS: Allow access to company data only from expected networks](#)」を参照してください。

## データ境界について詳しく学ぶためのリソース

次のリソースは、AWS 全体のデータ境界について詳しく学ぶのに役立ちます。

- [Data perimeters on AWS](#) — データ境界およびその利点と使用例について説明します。
- [Whitepaper: Building a Data Perimeter on AWS](#) — このドキュメントでは、AWS の ID、リソース、ネットワークの周囲に境界を構築するためのベストプラクティスおよび利用可能なサービスについて説明しています。
- [Webinar: Building a data perimeter in AWS](#) — さまざまなリスクシナリオに基づいてデータ境界コントロールを適用する場所と方法を説明します。
- [Blog Post Series: Establishing a Data Perimeter on AWS](#) — これらのブログ記事では、セキュリティや実装に関する重要な考慮事項など、データ境界を大規模に確立するための規範的なガイダンスを取り上げています。
- [Data perimeter policy examples](#) — この GitHub リポジトリには、AWS でデータ境界を実装するのに役立つ、いくつかの一般的なパターンをカバーするポリシー例が含まれています。
- [Data perimeter helper](#) — このツールは、[AWS CloudTrail](#) ログ内のアクセスアクティビティを分析することで、データ境界コントロールの設計と影響の予測に役立ちます。

## IAM エンティティのアクセス許可境界

AWS は、IAM エンティティ (ユーザーまたはロール) のアクセス許可の境界をサポートしています。アクセス許可の境界は、管理ポリシーを使用してアイデンティティベースのポリシーが IAM エンティティに付与できるアクセス許可の上限を設定する高度な機能です。エンティティのアクセス許可の境界により、エンティティは、アイデンティティベースのポリシーとそのアクセス許可の境界の両方で許可されているアクションのみを実行できます。

ポリシーの詳細については、「[ポリシータイプ](#)」を参照してください。

**⚠ Important**

アクセス許可の境界ポリシーがアタッチされている IAM ユーザーまたはロールに対する Deny 効果を持つ NotPrincipal ポリシー要素を含むリソースベースのポリシーステートメントは使用しないでください。Deny 効果のある NotPrincipal 要素は、NotPrincipal 要素で指定されている値に関係なく、アクセス許可の境界ポリシーがアタッチされている IAM プリンシパルを常に拒否します。これにより、本来であればリソースにアクセスできたはずの IAM ユーザーまたはロールの一部がアクセスを失うこととなります。リソースベースのポリシーステートメントを変更して、NotPrincipal 要素ではなく [aws:PrincipalArn](#) コンテキストキーで条件演算子 [ArnNotEquals](#) を使用してアクセスを制限することをおすすめします。NotPrincipal 要素の詳細については、「[AWS JSON ポリシーの要素: NotPrincipal](#)」を参照してください。

IAM エンティティ (ユーザーまたはロール) の境界を設定するには、AWS 管理ポリシーまたはカスタマー管理ポリシーを使用します。このポリシーでは、ユーザーやロールのアクセス許可の上限を設定します。

たとえば、ShirleyRodriguez という名前の IAM ユーザーが Amazon S3、Amazon CloudWatch、および Amazon EC2 のみを管理できるようにする必要があるとします。このルールを適用するには、次のポリシーを使用して ShirleyRodriguez ユーザーのアクセス許可の境界を設定できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*"
      ],
      "Resource": "*"
    }
  ]
}
```

ポリシーを使用してユーザーのアクセス許可の境界を設定する場合、このポリシーではユーザーのアクセス許可は制限しますが、それ自体のアクセス許可は提供しません。この例のポリシーでは、ShirleyRodriguez のアクセス許可の上限を Amazon S3、CloudWatch、および Amazon EC2 のすべてのオペレーションに設定します。Shirley は、IAM を含む他のどのサービスでもオペレーションを実行することはできません。実行を許可するアクセス許可ポリシーが適用されている場合でも同様です。たとえば、ShirleyRodriguez ユーザーに次のポリシーを追加できます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

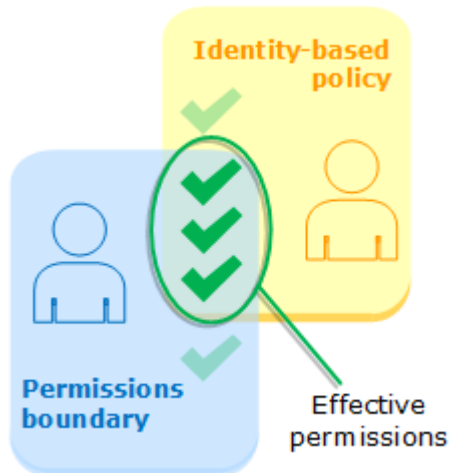
このポリシーでは、IAM のユーザーの作成を許可します。このアクセス許可ポリシーを ShirleyRodriguez ユーザーにアタッチすると、Shirley はユーザーを作成しようとしても、オペレーションは失敗します。アクセス許可の境界で iam:CreateUser オペレーションが許可されていないため、失敗します。これら 2 つのポリシーを考慮すると、Shirley には AWS でのオペレーションを実行するアクセス許可がありません。Amazon S3 などの他のサービスでのアクションを許可するには、別のアクセス許可ポリシーを追加する必要があります。または、IAM のユーザーの作成を許可するように、アクセス許可の境界を更新します。

## 境界を設定した場合の有効なアクセス許可の評価

IAM エンティティ (ユーザーまたはロール) のアクセス許可の境界では、エンティティに許可されるアクセス許可の上限が設定されます。これにより、ユーザーやロールの有効なアクセス許可が変わる場合があります。エンティティの有効なアクセス許可は、ユーザーやロールに影響するすべてのポリシーによって付与されるアクセス許可です。エンティティのアクセス許可は、アカウント内で、アイデンティティベースのポリシー、リソースベースのポリシー、アクセス許可の境界、Organizations SCP、またはセッションポリシーの影響を受ける場合があります。各種ポリシーの詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

これらのいずれかのポリシータイプによって、オペレーションへのアクセスが明示的に拒否された場合、そのリクエストは拒否されます。複数のアクセス許可タイプによってエンティティに付与されたアクセス許可はさらに複雑です。AWS でのポリシーの評価の詳細については、「[ポリシーの評価論理](#)」を参照してください。

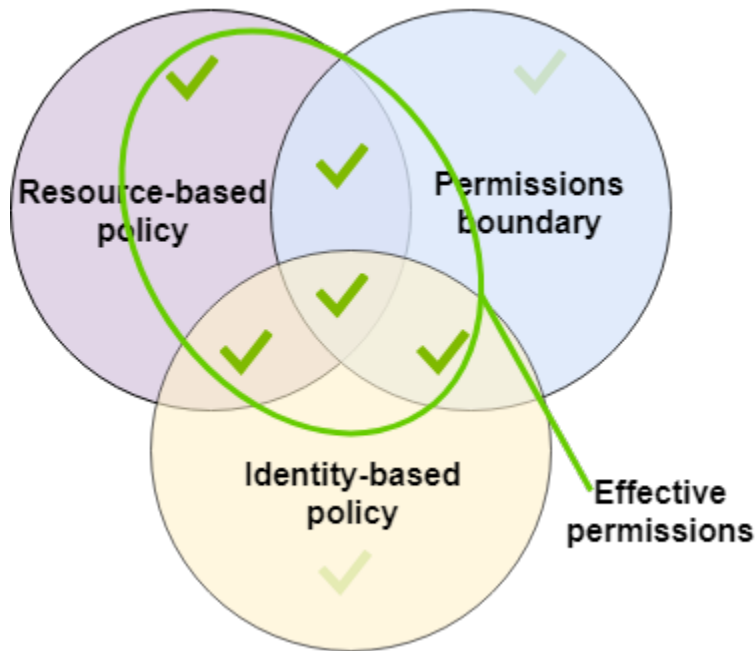
アイデンティティベースのポリシー (境界あり) – アイデンティティベースのポリシーは、ユーザー、ユーザーのグループ、またはロールにアタッチされているインラインポリシーまたは管理ポリシーです。アイデンティティベースのポリシーはエンティティにアクセス許可を付与し、アクセス許可の境界は、それらのアクセス許可を制限します。有効なアクセス許可は、両方のポリシータイプの共通部分です。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。



リソースベースのポリシー – リソースベースのポリシーでは、指定されたプリンシパルが、ポリシーがアタッチされているリソースにアクセスする方法を制御します。

#### IAM ユーザー用のリソースベースのポリシー

同じアカウント内では、IAM ユーザー ARN (フェデレーテッドユーザーセッションではない) へのアクセス許可を与えているリソースベースのポリシーは、ID ベースのポリシーまたはアクセス許可の境界による、暗黙的な拒否によって制限されません。



## IAM ロール用のリソースベースのポリシー

IAM ロール — IAM ロール ARN にアクセス許可を与えるリソースベースのポリシーは、アクセス許可の境界またはセッションポリシーの暗黙的な拒否によって制限されます。

IAM ロールセッション — 同じアカウント内で、IAM ロールセッション ARN にアクセス許可を与えるリソースベースのポリシーは、引き受けたロールセッションに、直接アクセス許可を与えます。セッションに直接付与されるアクセス許可は、ID ベースのポリシー、アクセス許可の境界、またはセッションポリシーの暗黙的な拒否によって制限されません。ロールを引き受けてリクエストを行う場合、リクエストを行うプリンシパルは IAM ロールセッション ARN であり、ロールそれ自体の ARN ではありません。

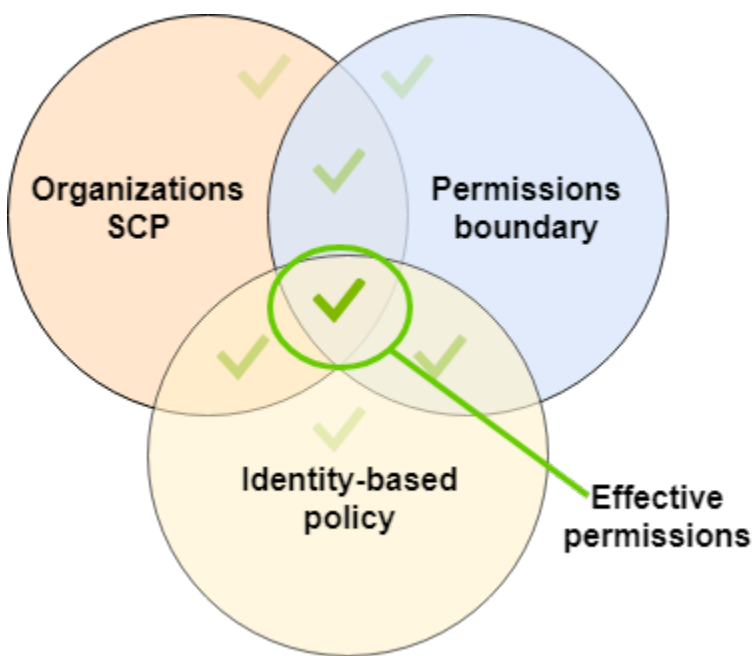
## IAM フェデレーティッドユーザーセッションのリソースベースのポリシー

IAM フェデレーティッドユーザーセッション — IAM フェデレーティッドユーザーセッションは、[GetFederationToken](#) を呼び出して作成されたセッションです。フェデレーティッドユーザーがリクエストを行う場合、リクエストを行うプリンシパルはフェデレーティッドユーザー ARN であり、フェデレーションした IAM ユーザーの ARN ではありません。同じアカウント内で、フェデレーティッドユーザー ARN にアクセス許可を与えるリソースベースのポリシーは、セッションに直接アクセス許可を与えます。セッションに直接付与されるアクセス許可は、ID ベースのポリシー、アクセス許可の境界、またはセッションポリシーの暗黙的な拒否によって制限されません。



ただし、リソースベースのポリシーがフェデレーションした IAM ユーザーの ARN にアクセス許可を与える場合、セッション中にフェデレートドユーザーによって行われたリクエストは、アクセス許可の境界またはセッションポリシーの、暗黙的な拒否によって制限されます。

Organizations SCP - SCP は、AWS アカウント 全体に適用されます。この場合、アカウント内のプリンシパルによって行われるすべてのリクエストのアクセス許可が制限されます。IAM エンティティ (ユーザーまたはロール) は、SCP、アクセス許可の境界、およびアイデンティティベースのポリシーの影響を受けるリクエストを行うことができます。この場合、そのリクエストは、その 3 つすべてのポリシータイプで許可されている場合にのみ許可されます。有効なアクセス許可は、その 3 つすべてのポリシータイプの共通部分です。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。

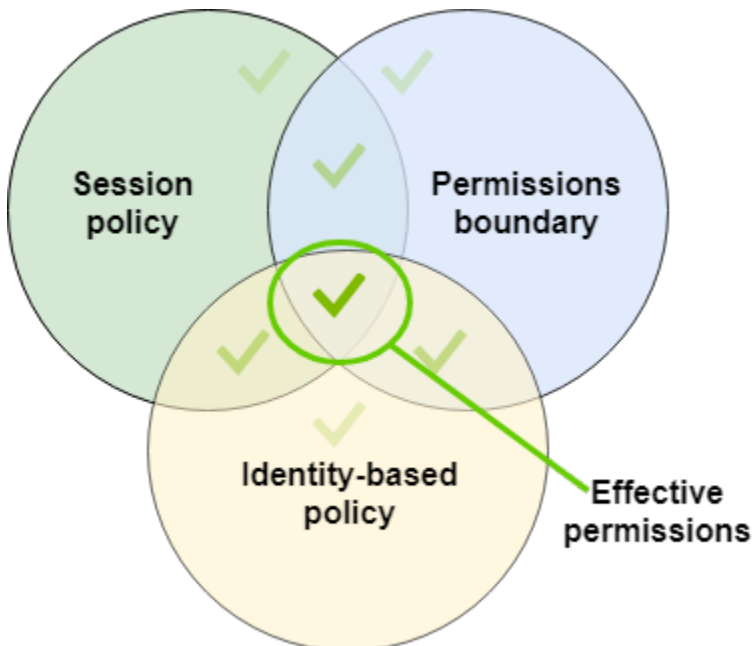


[自分のアカウントが AWS Organizations の組織のメンバーであるかどうか](#)を確認できます。組織のメンバーは、SCP による影響を受ける可能性があります。AWS CLI コマンドまたは AWS API オペレーションを使用してこのデータを表示するには、Organizations エンティティに対する `organizations:DescribeOrganization` アクションのアクセス許可が必要です。Organizations コンソールでオペレーションを実行するには、追加のアクセス許可が必要です。SCP が特定のリクエストへのアクセスを拒否しているかどうかを確認する、または有効なアクセス権限を変更するには、AWS Organizations 管理者に連絡してください。

セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。セッションのアクセス許可は、セッションの作成に使用する IAM エンティティ (ユーザーまたはロール) と、セッ



セッションポリシーから派生します。エンティティのアイデンティティベースのポリシーのアクセス許可は、セッションポリシーとアクセス許可の境界で制限されています。この一連のポリシータイプの有効なアクセス許可は、その3つすべてのポリシータイプの共通部分です。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。セッションポリシーの詳細については、「[セッションポリシー](#)」を参照してください。



## アクセス許可の境界を使用した他のユーザーへの責任の委任

アクセス許可の境界を使用して、アクセス許可の管理タスク (ユーザーの作成など) をアカウントの IAM ユーザーに委任できます。これにより、アクセス許可の特定の境界内で、ユーザーの代わりに他のユーザーがタスクを実行できます。

例えば、Maria は X-Company の AWS アカウント の管理者であるとして。彼女は、ユーザーの作成業務を Zhang を委任したいと考えます。しかし、Zhang が以下の社内ルールに従ってユーザーを作成することを確認する必要があります。

- ユーザーは、IAM を使用して管理ユーザー、グループ、グループロール、またはポリシーを作成できません。
- ユーザーは、Amazon S3 logs バケットへのアクセスを拒否される。また、Amazon EC2 インスタンス i-1234567890abcdef0 にはアクセスできない。
- ユーザーは各自の境界ポリシーを削除できない。

これらのルールを適用するために、María は以下のタスクを実行します (各タスクの詳細は後述します)。

1. María は、XCompanyBoundaries 管理ポリシーを作成し、これをアカウントのすべての新しいユーザーに対するアクセス許可の境界として使用します。
2. María は、DelegatedUserBoundary 管理ポリシーを作成し、これを Zhang のアクセス許可の境界として割り当てます。María は、管理者 ユーザーの ARN を書き留め、それをポリシーで使用して、Zhang がアクセスできないようにします。
3. María は、DelegatedUserPermissions 管理ポリシーを作成し、これを Zhang のアクセス許可ポリシーとしてアタッチします。
4. María は、Zhang に新しい責任と制限を伝えます。

タスク 1: María は、最初に管理ポリシーを作成して、新しいユーザーの境界を定義する必要があります。María は、必要なアクセス許可ポリシーをユーザーに付与することを Zhang に許可しますが、これらのユーザーを制限したいと思います。これを行うには、次のカスタマー管理ポリシーを XCompanyBoundaries という名前で作成します。このポリシーは以下の処理を実行します。

- 複数のサービスへのフルアクセスをユーザーに許可する
- IAM コンソールでの制限された自己管理アクセスを許可する つまり、ユーザーはコンソールにサインインした後にパスワードを変更できます。初期パスワードを設定することはできません。これを許可するには、"\*LoginProfile" アクションを AllowManageOwnPasswordAndAccessKeys ステートメントに追加します。
- Amazon S3 ログバケットまたは i-1234567890abcdef0 Amazon EC2 インスタンスへのユーザーアクセスを拒否します

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceBoundaries",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*",
        "dynamodb:*"
      ],
    }
  ],
}
```

```
    "Resource": "*"
  },
  {
    "Sid": "AllowIAMConsoleForCredentials",
    "Effect": "Allow",
    "Action": [
      "iam:ListUsers",
      "iam:GetAccountPasswordPolicy"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowManageOwnPasswordAndAccessKeys",
    "Effect": "Allow",
    "Action": [
      "iam:*AccessKey*",
      "iam:ChangePassword",
      "iam:GetUser",
      "iam:*ServiceSpecificCredential*",
      "iam:*SigningCertificate*"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
  },
  {
    "Sid": "DenyS3Logs",
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::logs",
      "arn:aws:s3:::logs/*"
    ]
  },
  {
    "Sid": "DenyEC2Production",
    "Effect": "Deny",
    "Action": "ec2:*",
    "Resource": "arn:aws:ec2::*:instance/i-1234567890abcdef0"
  }
]
}
```

ステートメントごとに用途は異なります。

1. このポリシーの `ServiceBoundaries` ステートメントでは、指定された AWS のサービスへのフルアクセスを許可します。つまり、これらのサービスにおける新しいユーザーのアクションは、ユーザーにアタッチされているアクセス許可ポリシーによってのみ制限されます。
2. `AllowIAMConsoleForCredentials` ステートメントは、すべての IAM ユーザーを一覧表示するためのアクセス権を付与します。このアクセス権は、AWS Management Console で [ユーザー] ページに移動するために必要です。また、このアクセス権では、アカウントのパスワード要件を表示することができます。これは、自分のパスワードを変更する場合に必要です。
3. `AllowManageOwnPasswordAndAccessKeys` ステートメントは、自分のパスワードおよびプログラムを使用したアクセスキーのみの管理を許可します。これは、Zhang や別の管理者が IAM へのフルアクセスを許可するアクセス許可ポリシーを新しいユーザーに割り当てる場合に重要です。この場合、そのユーザーが自分や他のユーザーのアクセス許可を変更することができます。このステートメントで、これを防止します。
4. `DenyS3Logs` ステートメントでは、logs バケットへのアクセスを明示的に拒否します。
5. `DenyEC2Production` ステートメントでは、`i-1234567890abcdef0` インスタンスへのアクセスを明示的に拒否します。

タスク 2: Maria は、すべての X-Company ユーザーを作成することを Zhang に許可しますが、条件として `XCompanyBoundaries` をアクセス許可の境界とします。そのために、次のカスタマー管理ポリシーを `DelegatedUserBoundary` という名前で作成します。このポリシーでは、Zhang に許可されるアクセス許可の上限を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateOrChangeOnlyWithBoundary",
      "Effect": "Allow",
      "Action": [
        "iam:AttachUserPolicy",
        "iam:CreateUser",
        "iam>DeleteUserPolicy",
        "iam:DetachUserPolicy",
        "iam:PutUserPermissionsBoundary",
        "iam:PutUserPolicy"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
        "iam:PermissionsBoundary": "arn:aws:iam::123456789012:policy/
XCompanyBoundaries"
    }
}
},
{
  "Sid": "CloudWatchAndOtherIAMTasks",
  "Effect": "Allow",
  "Action": [
    "cloudwatch:*",
    "iam:CreateAccessKey",
    "iam:CreateGroup",
    "iam:CreateLoginProfile",
    "iam:CreatePolicy",
    "iam>DeleteGroup",
    "iam>DeletePolicy",
    "iam>DeletePolicyVersion",
    "iam>DeleteUser",
    "iam:GetAccountPasswordPolicy",
    "iam:GetGroup",
    "iam:GetLoginProfile",
    "iam:GetPolicy",
    "iam:GetPolicyVersion",
    "iam:GetRolePolicy",
    "iam:GetUser",
    "iam:GetUserPolicy",
    "iam:ListAccessKeys",
    "iam:ListAttachedRolePolicies",
    "iam:ListAttachedUserPolicies",
    "iam:ListEntitiesForPolicy",
    "iam:ListGroups",
    "iam:ListGroupsForUser",
    "iam:ListMFADevices",
    "iam:ListPolicies",
    "iam:ListPolicyVersions",
    "iam:ListRolePolicies",
    "iam:ListSSHPublicKeys",
    "iam:ListServiceSpecificCredentials",
    "iam:ListSigningCertificates",
    "iam:ListUserPolicies",
    "iam:ListUsers",
    "iam:SetDefaultPolicyVersion",
    "iam:SimulateCustomPolicy",
    "iam:SimulatePrincipalPolicy",
```

```

        "iam:UpdateGroup",
        "iam:UpdateLoginProfile",
        "iam:UpdateUser"
    ],
    "NotResource": "arn:aws:iam::123456789012:user/Maria"
},
{
    "Sid": "NoBoundaryPolicyEdit",
    "Effect": "Deny",
    "Action": [
        "iam:CreatePolicyVersion",
        "iam>DeletePolicy",
        "iam>DeletePolicyVersion",
        "iam:SetDefaultPolicyVersion"
    ],
    "Resource": [
        "arn:aws:iam::123456789012:policy/XCompanyBoundaries",
        "arn:aws:iam::123456789012:policy/DelegatedUserBoundary"
    ]
},
{
    "Sid": "NoBoundaryUserDelete",
    "Effect": "Deny",
    "Action": "iam>DeleteUserPermissionsBoundary",
    "Resource": "*"
}
]
}

```

ステートメントごとに用途は異なります。

1. `CreateOrChangeOnlyWithBoundary` ステートメントでは、IAM ユーザーを作成することを Zhang に許可します。ただし、`XCompanyBoundaries` ポリシーを使用してアクセス許可の境界を設定することを条件とします。また、このステートメントでは、既存のユーザーにアクセス許可の境界を設定することを Zhang に許可します。ただし、同じポリシーを使用することを条件とします。最後に、このステートメントでは、このアクセス許可の境界が設定されたユーザーのアクセス許可ポリシーを管理することを Zhang に許可します。
2. `CloudWatchAndOtherIAMTasks` ステートメントでは、他のユーザー、グループ、およびポリシーの管理タスクを実行することを Zhang に許可します。Zhang には、`NotResource` ポリシー要素にリストされていないすべての IAM ユーザーのパスワードをリセットして、アクセスキーを作成する許可があります。これにより、ユーザーのサインインの問題を支援できます。

3. NoBoundaryPolicyEdit ステートメントでは、XCompanyBoundaries ポリシーを更新するためのアクセスを Zhang に拒否します。自分自身や他のユーザーのアクセス許可の境界を設定するために使用されているポリシーを変更することは許可されません。
4. NoBoundaryUserDelete ステートメントは、Zhang が自分自身あるいは他のユーザーのアクセス許可境界を削除するためにアクセスすることを拒否します。

次に Maria は、Zhang ユーザーの [アクセス許可の境界](#) として DelegatedUserBoundary ポリシーを割り当てます。

タスク 3: アクセス許可の境界ではアクセス許可の上限を設定するだけで、それ自体はアクセスを付与しないため、Maria は Zhang のアクセス許可ポリシーを作成する必要があります。次のポリシーを DelegatedUserPermissions という名前で作成します。このポリシーでは、設定された境界内で、Zhang が実行できるオペレーションを定義します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAM",
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchLimited",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetDashboard",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListDashboards",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketContents",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::ZhangBucket"
    }
  ]
}
```

```
]
}
```

ステートメントごとに用途は異なります。

1. このポリシーの IAM ステートメントでは、IAM へのフルアクセスを Zhang に許可します。ただし、許可される IAM オペレーションはアクセス許可の境界で制限されます。有効な IAM アクセス許可を制限するのはアクセス許可の境界のみです。
2. CloudWatchLimited ステートメントでは、CloudWatch で 5 つのアクションを実行することを Zhang に許可します。アクセス許可の境界ですべての CloudWatch アクションが許可されるため、有効な CloudWatch アクセス許可はアクセス許可ポリシーでのみ限定されます。
3. S3BucketContents ステートメントでは、Amazon S3 バケット ZhangBucket を表示することを Zhang に許可します。ただし、アクセス許可の境界で一切の Amazon S3 アクションが許可されないため、アクセス許可ポリシーにかかわらず、S3 オペレーションを実行することはできません。

#### Note

Zhang のポリシーにより、彼は自分がアクセスできない Amazon S3 リソースにアクセスできるユーザーを作成できます。これらの管理アクションを委任することによって、Maria は Zhang に Amazon S3 へのアクセス権を効果的に信頼します。

次に Maria は、DelegatedUserPermissions ユーザーのアクセス許可ポリシーとして Zhang ポリシーをアタッチします。

タスク 4: Maria は、新しいユーザーを作成する手順を Zhang に伝えます。新しいユーザーを作成して必要なアクセス権を付与できますが、アクセス許可の境界として XCompanyBoundaries ポリシーを割り当てる必要があることを説明します。

Zhang は以下のタスクを実行します。

1. Zhang は AWS Management Console を使用して [ユーザーを作成](#) します。ユーザー名として「Nikhil」と入力し、このユーザーに対してコンソールへのアクセスを有効にします。上記のポリシーではユーザーが IAM コンソールにサインインした後にのみパスワードを変更できるため、[Requires password reset] (パスワードのリセットが必要) の横にあるチェックボックスをオフにします。



2. [アクセス許可の設定] ページで、Zhang は Nikhil にタスクの実行を許可するアクセス許可ポリシーとして [IAMFullAccess] と [AmazonS3ReadOnlyAccess] を選択します。
3. Zhang は、María に教えられた手順を忘れて、[Set permissions boundary (アクセス許可の境界の設定)] セクションをスキップします。
4. Zhang はユーザーの詳細を確認し、[ユーザーの作成] を選択します。

オペレーションは失敗し、アクセスが拒否されます。Zhang の DelegatedUserBoundary アクセス許可の境界では、作成するユーザーにアクセス許可の境界として XCompanyBoundaries ポリシーが必要です。

5. Zhang は前のページに戻ります。[Set permissions boundary (アクセス許可の境界の設定)] セクションで、XCompanyBoundaries ポリシーを選択します。
6. Zhang はユーザーの詳細を確認し、[ユーザーの作成] を選択します。

ユーザーが作成されます。

Nikhil は、サインインすると、アクセス許可の境界で拒否されているオペレーションを除いて、IAM と Amazon S3 にアクセスできます。たとえば、IAM で自分のパスワードは変更できますが、別のユーザーを作成したり、自分のポリシーを編集したりすることはできません。Nikhil は Amazon S3 への読み取り専用アクセス権を持っています。

リソースベースのポリシーを logs バケットに追加して Nikhil がそのバケットにオブジェクトを配置できるようにしても、Nikhil はこのバケットにアクセスできません。理由は、logs バケットのアクションはすべて、彼のアクセス許可の境界によって明示的に拒否されているためです。いずれかのポリシータイプで明示的に拒否されていると、リクエストは拒否されません。ただし、Secrets Manager シークレットにアタッチされているリソースベースのポリシーで、secretsmanager:GetSecretValue アクションの実行を Nikhil に許可している場合、Nikhil は、そのシークレットを取得して復号できます。これは、Secrets Manager オペレーションは、彼のアクセス許可境界によって明示的に拒否されており、アクセス許可の境界の暗黙的な拒否により、リソースベースのポリシーが制限されないためです。

## アイデンティティベースおよびリソースベースのポリシー

ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けられると、これらの許可を定義します。アクセス許可ポリシーを作成してリソースへのアクセスを制限する場合、アイデンティティベースのポリシーまたはリソースベースのポリシーを選択できます。

アイデンティティベースのポリシーは、IAM ユーザー、グループ、ロールにアタッチされます。これらのポリシーを使用すると、そのアイデンティティが実行できる内容 (そのアクセス許可) を指定できます。たとえば、John という名前の IAM ユーザーに Amazon EC2 RunInstances アクションを実行することを許可するポリシーをアタッチできます。このポリシーでは、John は、MyCompany という名前の Amazon DynamoDB テーブルからアイテムを取得することもできます。また、John は、自分の IAM セキュリティ認証情報を管理することもできます。ID ベースのポリシーは [管理またはインライン](#) とすることができま

リソースベースのポリシーをリソースにアタッチします。例えば、リソースベースのポリシーを Amazon S3 バケット、Amazon SQS キュー、VPC エンドポイント、AWS Key Management Service 暗号化キー、Amazon DynamoDB テーブルとストリームにアタッチできます。リソースベースのポリシーをサポートするサービスのリストについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

リソースベースのポリシーによって、誰がリソースにアクセスでき、彼らがリソースでどのようなアクションを実行できるかを指定できます。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。リソースベースのポリシーはインラインのみで、管理ポリシーはありません。

#### Note

リソースベースのポリシーは、リソースレベルのアクセス許可とは異なっています。リソースベースのポリシーは、このトピックで説明しているように、リソースに直接アタッチできます。リソースレベルのアクセス許可は、[ARN](#) を使用してポリシー内で個別のリソースを指定する機能のことです。リソースベースのポリシーは、一部の AWS サービスでのみサポートされます。リソースベースのポリシーおよびリソースレベルのアクセス許可をサポートするサービスのリストについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

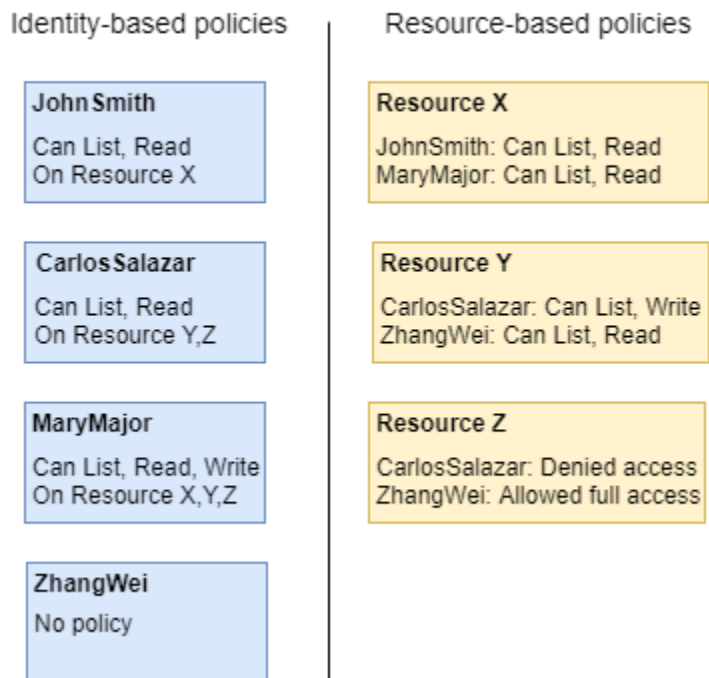
ID ベースのポリシーとリソースベースのポリシーが同じアカウント内でどのように相互作用するかについては、「[単一アカウント内のポリシーを評価する](#)」を参照してください

アカウント間でポリシーがどのようにやり取りされるかについては、[クロスアカウントポリシーの評価論理](#)を参照してください。

これらの概念を深く理解するには、以下の図を確認します。123456789012 アカウントの管理者は、アイデンティティベースのポリシーを JohnSmith、CarlosSalazar、および MaryMajor

ユーザーにアタッチしました。これらのポリシーのアクションの一部は、特定のリソースで実行できます。たとえば、ユーザー JohnSmith は、一部のアクションを Resource X で実行できます。これは、アイデンティティベースのポリシーのリソースレベルのアクセス許可です。また、管理者は、リソースベースのポリシーを Resource X、Resource Y、および Resource Z に追加しました。リソースベースのポリシーでは、リソースにアクセスできるユーザーを指定することができます。たとえば、Resource X のリソースベースのポリシーでは、JohnSmith および MaryMajor ユーザーは、リソースへのアクセス権を表示し、読み取ることができます。

Account ID: 123456789012



123456789012 アカウントの例では、次のユーザーは、表示されたアクションを実行することができます。

- JohnSmith – John は、Resource X のアクションを表示して、読み取ることができます。ユーザーにはアイデンティティベースのポリシー、Resource X にはリソースベースのポリシーによってこのアクセス許可が付与されます。
- CarlosSalazar – Carlos は、Resource Y のアクションの一覧表示、読み取り、書き込みを行うことができますが、Resource Z へのアクセスは拒否されています。Carlos のアイデンティティベースのポリシーでは、Resource Y のアクションの一覧表示および読み取りを行うことができます。Resource Y リソースベースのポリシーでは、アクセス許可の書き込みを行うことができます。ただし、アイデンティティベースのポリシーでは、Resource Z にアクセスすることができ、Resource Z リソースベースのポリシーでは、そのアクセスは拒否されます。明示的な Deny

によって、Allow は上書きされ、Resource Z へのアクセスは拒否されます。詳細については、「[ポリシーの評価論理](#)」を参照してください。

- MaryMajor Mary – は、Resource X、Resource Y、および Resource Z のオペレーションの一覧表示、読み取り、および書き込みを行うことができます。Mary のアイデンティティベースのポリシーでは、リソースベースのポリシーよりも多くのリソースで追加のアクションを実行できますが、アクセスが拒否されることはありません。
- ZhangWei – Zhang は、Resource Z に対する完全なアクセス許可が付与されています。Zhang にはアイデンティティベースのポリシーはありませんが、Resource Z リソースベースのポリシーでは、そのリソースにフルアクセスすることができます。Zhang は、Resource Y でリストアクションと読み取りアクションを実行することもできます。

アイデンティティベースのポリシーとリソースベースのポリシーはいずれも、アクセス許可ポリシーであり、一緒に評価されます。アクセス許可ポリシーのみが適用されるリクエストの場合、AWS はすべてのポリシーで Deny がないかチェックします。存在する場合、リクエストは拒否されます。AWS によって、それぞれの Allow がないかどうか確認されます。1 つ以上のポリシーステートメントで、リクエストのアクションが許可されている場合、そのリクエストは許可されます。Allow がアイデンティティベースのポリシーか、リソースベースのポリシーであるかは関係ありません。

#### Important

このロジックは、リクエストが単一の AWS アカウント 内で行われた場合にのみ適用されます。あるアカウントから別のアカウントに行われるリクエストの場合、Account A のリクエストには、Account B のリソースへのリクエストを許可するアイデンティティベースのポリシーが必要です。また、Account B のリソースベースのポリシーを使用して、Account A のリクエストによるリソースへのアクセスを許可する必要があります。両方のアカウントに、オペレーションを許可するポリシーが必要です。それ以外の場合、リクエストは失敗します。クロスアカウントアクセスでリソースベースのポリシーを使用する方法の詳細については、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

特定のアクセス許可を持つユーザーが、アクセス許可ポリシーが関連付けられたリソースを要求する場合があります。このような場合、AWS がリソースへのアクセスを許可するかどうかを判断する際、両方のアクセス許可セットが評価されます。ポリシーの評価方法に関する詳細については、「[ポリシーの評価論理](#)」を参照してください。

**Note**

Amazon S3 では、アイデンティティベースのポリシーとリソースベースのポリシー (バケットポリシーと呼ばれます) をサポートしています。さらに、Amazon S3 は、IAM ポリシーおよびアクセス許可から独立した、アクセスコントロールリスト (ACL) と呼ばれるアクセス許可メカニズムをサポートしています。IAM ポリシーは、Amazon S3 ACL と組み合わせて使用できます。Amazon S3 アクセス許可に関する詳細は、Amazon Simple Storage Service ユーザーガイドの「[Access Control](#)」を参照してください。

## ポリシーを使用して AWS リソースへのアクセスを制御します。

ポリシーを使用して、IAM や AWS のすべてのサービスのリソースへのアクセスをコントロールできます。

[ポリシー](#)を使用して AWS へのアクセスを制御するには、AWS がアクセスを許可する方法を理解する必要があります。AWS はリソースの集合で構成されています。IAM ユーザーはリソースです。Amazon S3 バケットはリソースです。AWS API、AWS CLI、または AWS Management Console を使用してオペレーション (ユーザーの作成など) を実行する場合、このオペレーションに対するリクエストを送信します。リクエストでは、アクション、リソース、プリンシパルエンティティ (ユーザーまたはロール)、プリンシパルアカウント、および必要なリクエスト情報を指定します。これらのすべての情報により、コンテキストが提供されます。

次に、AWSはユーザー (プリンシパル) が認証され (サインイン済み)、指定されたリソースで指定されたアクションの実行が許可されている (権限を持っている) ことを確認します。認可時、AWS は、リクエストのコンテキストに該当するすべてのポリシーをチェックします。通常、ポリシーは [JSON ドキュメント](#)として AWS に保存され、プリンシパルエンティティのアクセス許可を指定します。ポリシーのタイプと用途の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

AWS は、リクエストの各部分がポリシーで許可されている場合のみ、リクエストを許可します。このプロセスの図を表示するには、「[IAM の仕組み](#)」を参照してください。AWS でリクエストを承認するかどうかの決定方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。

IAM ポリシーを作成するときは、以下へのアクセスを制御できます。

- [プリンシパル](#) – リクエストを行っているユーザー ([プリンシパル](#)) に許可される操作を制御します。

- [IAM ID](#) – どの IAM ID (ユーザーグループ、ユーザー、ロール) にどのようにアクセスできるかを制御します。
- [IAM ポリシー](#) – だれがカスタマー管理ポリシーを作成、編集、削除でき、だれがすべての管理ポリシーをアタッチおよびデタッチできるかを制御します。
- [AWS リソース](#) – アイデンティティベースのポリシーまたはリソースベースのポリシーを使用して、リソースにアクセスできるユーザーを制御します。
- [AWS アカウント](#) – リクエストを特定のアカウントのメンバーのみに許可するかどうか制御します。

ポリシーは、誰が AWS リソースにアクセスできるか、またそれらのリソース上でどのようなアクションを実行できるかを指定することを可能にします。すべての IAM ユーザーには、初期状態ではアクセス許可はありません。言い換えると、デフォルト設定では、ユーザーは何もできず、そのユーザーのアクセスキーを参照することすらできません。ユーザーに何かの操作を実行するアクセス許可を付与する場合は、そのユーザーにアクセス許可を追加できます (つまり、ポリシーをユーザーにアタッチします)。または、目的のアクセス許可を持つユーザーグループにユーザーを追加できます。

例えば、自らのアクセスキーをリスト化するためのユーザー権限を付与することができます。また、権限を拡大し、各ユーザーが自らのキーを作成、更新、および削除できるようにすることもできます。

アクセス許可をユーザーグループに付与することで、そのユーザーグループ内のすべてのユーザーにアクセス許可を与えることができます。例えば、AWS アカウントのリソースに対してすべての IAM アクションを実行するアクセス許可を Administrators グループに付与できます。その他の例: AWS アカウントの Amazon EC2 インスタンスを定義するアクセス許可を Managers ユーザーグループに付与できます。

ユーザー、ユーザーグループ、およびロールに基本的な権限を委任する方法については、「[他の IAM リソースにアクセスするのに必要なアクセス許可](#)」を参照してください。基本的な権限を示すポリシーのさらに多くの例については、「[IAM リソースの管理に関するポリシーの例](#)」を参照してください。

## プリンシパルへのアクセスの制御

リクエスト元 (プリンシパル) に許可される操作を制御するには、ポリシーを使用します。そのためには、リクエスト元のアイデンティティ (ユーザー、ユーザーグループ、またはロール) にアイデンティティベースのポリシーをアタッチする必要があります。また、[アクセス許可の境界](#)を使用して、エンティティ (ユーザーまたはロール) に付与することのできるアクセス許可の上限を設定することもできます。



たとえば、CloudWatch、Amazon DynamoDB、Amazon EC2、および Amazon S3 に対するフルアクセスをユーザー Zhang Wei に許可するとします。この場合、2 つの異なるポリシーを作成し、後で別のユーザーにアクセス許可セットの 1 つが必要になった場合に、ポリシーを分割できます。または、両方のアクセス許可を 1 つのポリシーにまとめて、このポリシーを Zhang Wei という IAM ユーザーにアタッチできます。また、Zhang が属しているユーザーグループや、Zhang が引き受けることができるロールにポリシーをアタッチすることもできます。その結果、Zhang が S3 バケットのコンテンツを表示する場合、リクエストは許可されます。新しい IAM ユーザーを作成しようとすると、必要なアクセス許可がないため、このリクエストは拒否されます。

S3 バケット DOC-EXAMPLE-BUCKET1 へのアクセスを完全に遮断するには、Zhang に対してアクセス許可の境界を設定できます。そのためには、Zhang に付与するアクセス許可の上限を決定します。この場合、ユーザーの操作を制御するために、アクセス許可ポリシーを使用します。ここで必要なことは、ユーザーが機密バケットにアクセスしないことだけです。したがって、以下のポリシーを使用して Zhang の境界を定義し、Amazon S3 と他のいくつかのサービスに対するすべての AWS アクションを許可する一方で、S3 バケット DOC-EXAMPLE-BUCKET1 へのアクセスを拒否します。このアクセス許可の境界では、一切の IAM アクションが許可されないため、Zhang は自分 (または他のいかなるユーザー) の境界も削除できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PermissionsBoundarySomeServices",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "dynamodb:*",
        "ec2:*",
        "s3:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PermissionsBoundaryNoConfidentialBucket",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
      ]
    }
  ]
}
```

```
]
}
```

このようなポリシーをユーザーのアクセス許可の境界として割り当てる場合、境界自体はアクセス許可を付与しないことに注意してください。アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の上限を設定します。アクセス許可の境界の詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。

前述の手順の詳細については、以下のリソースを参照してください。

- プリンシパルにアタッチできる IAM ポリシーの作成の詳細については、「[IAM ポリシーの作成](#)」を参照してください。
- IAM ポリシーをプリンシパルにアタッチする方法については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。
- EC2 へのフルアクセス許可を付与するポリシーの例を参照するには、「[Amazon EC2: 特定のリージョンでの完全な EC2 アクセスをプログラムによりコンソールで許可する](#)」を参照してください。
- S3 バケットへの読み取り専用アクセスを許可するには、「[Amazon S3: S3 バケットのオブジェクトへの読み取りおよび書き込みアクセスをプログラムによりコンソールで許可する](#)」のポリシー例の最初の 2 つのステートメントを使用します。
- コンソールパスワード、プログラムによるアクセスキー、MFA デバイスなどの認証情報の設定をユーザーに許可するポリシー例を確認するには、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

## アイデンティティへのアクセスコントロール

IAM ポリシーを使用して、ユーザーグループ経由ですべてのユーザーにアタッチするポリシーを作成することで、アイデンティティに対してユーザーが実行できる操作を制御できます。これを行うには、アイデンティティに対して実行できる操作や、アイデンティティにアクセスできるユーザーを制限するポリシーを作成します。

たとえば、AllUsers という名前のユーザーグループを作成し、そのグループをすべてのユーザーにアタッチできます。ユーザーグループを作成するときに、前のセクションで説明したように、認証情報を設定するアクセス許可をすべてのユーザーに付与できます。次に、ポリシーの条件にユーザー名が含まれていない限り、ユーザーグループを変更するアクセス許可を拒否するポリシーを作成できます。ただし、ポリシーのその部分では、リストされたユーザーを除くすべてのユーザーへのアクセスが拒否されます。また、ユーザーグループのすべてのユーザーに、すべてのユーザーグループ管理ア



クシオンを許可するアクセス許可も含める必要があります。最後に、このポリシーをユーザーグループにアタッチし、すべてのユーザーに適用されるようにします。その結果、ポリシーで指定されていないユーザーがユーザーグループに変更を加えようとすると、リクエストは拒否されます。

ビジュアルエディタを使用してこのポリシーを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。

初めて [ポリシー] を選択する場合には、[管理ポリシーによるこそ] ページが表示されます。[Get Started] (今すぐ始める) を選択します。

3. [Create policy] を選択します。
4. [ポリシーエディタ] セクションで、[ビジュアル] オプションを選択します。
5. [サービスを選択] で [IAM] を選択します。
6. [許可されたアクション] で、検索ボックスに **group** を入力します。ビジュアルエディタで、group という語を含むすべての IAM アクションが表示されます。すべてのチェックボックスをオンにします。
7. [リソース] を選択して、ポリシーのリソースを指定します。選択したアクションに基づいて、[グループ] および [ユーザー] のリソースタイプが表示されます。
  - グループ – [ARN を追加] を選択します。[リソース] で、[任意のアカウント] オプションを選択します。[パス付きの任意のグループ名] チェックボックスを選択し、ユーザーグループ名 **AllUsers** を入力します。[ユーザーを追加] を選択します。
  - [ユーザー] – [このアカウント内のすべて] の横にあるチェックボックスをオンにします。

選択したアクションの 1 つである ListGroups は、特定のリソースの使用をサポートされていません。そのアクションに対して [All resources (すべてのリソース)] を選択する必要はありません。[JSON] エディタでポリシーを保存するか、ポリシーを表示すると、IAM によって自動的に新しいアクセス許可ブロックが作成され、すべてのリソースでこのアクションにアクセス許可が付与されることがわかります。

8. 別のアクセス許可ブロックを追加するには、[さらにアクセス許可を追加する] を選択します。
9. [サービスを選択] を選択してから、[IAM] を選択します。
10. [許可されるアクション]、[Sアクセス許可の拒否に切り替え] の順に選択します。この操作を行うと、ブロック全体を使用してアクセス許可が拒否されます。

11. 検索ボックスに「**group**」と入力します。ビジュアルエディタで、group という語を含むすべての IAM アクションが表示されます。次のアクションの横にあるチェックボックスをオンにします。

- CreateGroup
- DeleteGroup
- RemoveUserFromGroup
- AttachGroupPolicy
- DeleteGroupPolicy
- DetachGroupPolicy
- PutGroupPolicy
- UpdateGroup

12. [リソース] を選択して、ポリシーのリソースを指定します。選択したアクションに基づいて、[グループ] リソースタイプが表示されます。[ARN を追加] を選択します。[リソース] で、[任意のアカウント] オプションを選択します。[パス付きグループ名] にユーザーグループ名「**AllUsers**」を入力します。次に、[ARN を追加] を選択します。

13. [リクエスト条件 - オプション] を選択してから、[別の条件を追加] を選択します。次の値をフォームに入力します。

- 条件キー — aws:username を選択
- [限定条件] – [Default (デフォルト)] を選択します。
- [演算子] – [StringNotEquals] を選択します。
- 値 – 「**srodriguez**」と入力し、[追加] を選択します。「**mjackson**」と入力し、[追加] を選択して、別の値を入力します。「**adesai**」と入力し、[条件を追加] を選択します。

この条件により、呼び出しを実行しているユーザーがリストに含まれていない場合、アクセスは指定したユーザーグループ管理アクションに対して拒否されます。これによりアクセス許可が明示的に拒否されるため、ユーザーにアクションの呼び出しを許可した以前のブロックは上書きされます。リストのユーザーはアクセスを拒否され、最初のアクセス許可ブロックでアクセス許可を付与されるため、ユーザーはユーザーグループを完全に管理できます。

14. 完了したら、[Next] を選択します。

**Note**

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで変更を行うか [次へ] を択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

- [確認および作成] ページで、[ポリシー名] に「**LimitAllUserGroupManagement**」と入力します。[Description (説明)] に、「**Allows all users read-only access to a specific user group, and allows only specific users access to make changes to the user group**」と入力します。[このポリシーで定義されているアクセス許可] を確認し、意図したアクセス許可を付与したことを確認します。次に、[ポリシーの作成] を選択して新しいポリシーを保存します。
- ユーザーグループにポリシーをアタッチします。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

また、この JSON ポリシードキュメント例を使用して、同じポリシーを作成できます。この JSON ポリシーを表示するには、「[IAM:: 特定の IAM ユーザーによるグループの管理をプログラムによりコンソールで許可する](#)」を参照してください。JSON ドキュメントを使用してポリシーを作成する詳細な手順については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

## ポリシーへのアクセスの制御

ユーザーが AWS 管理ポリシーを適用する方法を制御できます。これを行うには、すべてのユーザーにこのポリシーをアタッチします。理想的には、ユーザーグループを使用してこれを行うことができます。

たとえば、[IAMUserChangePassword](#) と [PowerUserAccess](#) の AWS 管理ポリシーのみを新しい IAM ユーザー、ユーザーグループ、またはロールにアタッチするポリシーを作成できます。

カスタマー管理ポリシーの場合、それらのポリシーを作成、更新、削除できるユーザーを制御できます。プリンシパルエンティティ (ユーザーグループ、ユーザー、ロール) との間でポリシーをアタッチおよびデタッチできるユーザーを制御できます。ユーザーにどのエンティティのどのポリシーのアタッチまたはデタッチを許可するかも制御できます。

たとえば、アカウント管理者にポリシーを作成、更新、削除する権限を付与できます。次に、チームリーダーまたはその他の制限付き管理者に、管理対象となるプリンシパルエンティティのポリシーをアタッチおよびデタッチする権限を付与します。

詳細については、以下のリソースを参照してください。

- プリンシパルにアタッチできる IAM ポリシーの作成の詳細については、「[IAM ポリシーの作成](#)」を参照してください。
- IAM ポリシーをプリンシパルにアタッチする方法については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。
- 管理ポリシーの使用を制限するポリシー例については、「[IAM: ユーザー、グループ、またはロールに適用できる管理ポリシーを制限する](#)」を参照してください。

カスタマー管理ポリシーを作成、更新、削除する権限の制御

[IAM ポリシー](#)を使用して、だれが AWS アカウント のカスタマー管理ポリシーを作成、更新、削除できるかをコントロールできます。ポリシーまたはポリシーのバージョンの作成、更新、削除に直接関連する API オペレーションを以下に示します。

- [CreatePolicy](#)
- [CreatePolicyVersion](#)
- [DeletePolicy](#)
- [DeletePolicyVersion](#)
- [SetDefaultPolicyVersion](#)

以上の API オペレーションは、IAM ポリシーを使用して許可または拒否できる (権限を付与できる) アクションに対応しています。

次のポリシー例を考えます。この例では、AWS アカウント のすべてのカスタマー管理ポリシーのデフォルトバージョンを作成、更新 (新しいポリシーバージョンの作成)、削除、および設定することをユーザーに許可します。このポリシーの例では、ユーザーにポリシーの一覧表示とポリシーの取得も許可しています。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

## Example すべてのポリシーの作成、更新、削除、一覧表示、取得、デフォルトバージョンの設定を許可するポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:CreatePolicy",
      "iam:CreatePolicyVersion",
      "iam>DeletePolicy",
      "iam>DeletePolicyVersion",
      "iam:GetPolicy",
      "iam:GetPolicyVersion",
      "iam:ListPolicies",
      "iam:ListPolicyVersions",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
  }
}
```

指定した管理ポリシーのみに適用するポリシーを作成して、これらの API オペレーションの使用を制限することもできます。たとえば、特定のカスタマー管理ポリシーのみを対象にして、ユーザーにデフォルトバージョンの設定とポリシーバージョンの削除を許可することが考えられます。これは、権限を付与するポリシーの Resource 要素にポリシー ARN を指定することで実行できます。

次のポリシー例では、ポリシーバージョンを削除してデフォルトバージョンを設定することをユーザーに許可します。ただし、これらのアクションが許可されるのは、パス /TEAM-A/ が含まれているカスタマー管理ポリシーに限られます。カスタマー管理ポリシー ARN は、ポリシーの Resource 要素で指定されています (この例では、ARN にパスとワイルドカードが含まれているため、パス /TEAM-A/ を含むすべてのカスタマー管理ポリシーに一致します)。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

カスタマー管理ポリシーの名前にパスを使用する場合の詳細については、「[フレンドリ名とパス](#)」を参照してください。

## Example 特定のポリシーのみを対象にして、ポリシーバージョンの削除とデフォルトバージョンの設定を許可するポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "arn:aws:iam::account-id:policy/TEAM-A/*"
  }
}
```

### 管理ポリシーをアタッチおよびデタッチする権限の制御

また、IAM ポリシーを使用して、ユーザーが特定の管理ポリシーのみで作業することを許可できます。実際には、ユーザーがどのアクセス許可を他のプリンシパルエンティティに付与できるかをコントロールできます。

プリンシパルエンティティの管理ポリシーのアタッチとデタッチに直接関連する API オペレーションを以下に示します。

- [AttachGroupPolicy](#)
- [AttachRolePolicy](#)
- [AttachUserPolicy](#)
- [DetachGroupPolicy](#)
- [DetachRolePolicy](#)
- [DetachUserPolicy](#)

指定した管理ポリシーおよびプリンシパルエンティティのみに適用するポリシーを作成して、これらの API オペレーションの使用を制限することができます。たとえば、指定した管理ポリシーのみを対象にして、ユーザーに管理ポリシーのアタッチを許可することが考えられます。あるいは、指定したプリンシパルエンティティのみを対象にして、ユーザーに管理ポリシーのアタッチを許可することも考えられます。

以下のポリシー例では、ユーザーに、パス /TEAM-A/ を含むユーザーグループとロールのみに対して管理ポリシーのアタッチを許可します。ユーザーグループとロールの ARN はポリシーの Resource

要素で指定します (この例では、ARN にはパスとワイルドカード文字が含まれているため、パス /TEAM-A/ を含むすべてのユーザーグループとロールに一致します)。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

Example 特定のユーザーグループまたはロールのみに対する管理ポリシーのアタッチを許可するポリシー

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::account-id:group/TEAM-A/*",
      "arn:aws:iam::account-id:role/TEAM-A/*"
    ]
  }
}
```

前の例のアクションをさらに制限して、特定のポリシーにのみ影響を与えられるようにすることができます。つまり、ポリシーに条件を追加することで、ユーザーが他のプリンシパルエンティティにアタッチできる権限を制御できます。

次の例では、条件により、アタッチしたポリシーが指定したポリシーのいずれかに一致した場合のみに AttachGroupPolicy と AttachRolePolicy の権限を許可します。条件では、iam:PolicyARN [条件キー](#)を使用して、どのポリシーをアタッチできるかを決定しています。次のポリシー例は、前の例を拡張したものです。この例では、パス /TEAM-A/ が含まれている管理ポリシーを、パス /TEAM-A/ が含まれているユーザーグループとロールにのみアタッチすることをユーザーに許可します。この例の JSON ポリシードキュメントを使用してポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
```

```
    "iam:AttachRolePolicy"
  ],
  "Resource": [
    "arn:aws:iam::account-id:group/TEAM-A/*",
    "arn:aws:iam::account-id:role/TEAM-A/*"
  ],
  "Condition": {"ArnLike":
    {"iam:PolicyARN": "arn:aws:iam::account-id:policy/TEAM-A/*"}
  }
}
```

このポリシーでは、ArnLike 条件演算子を使用することもできますが、これら2つの条件演算子が同じように動作するため、ArnEquals 条件演算子を使用することもできます。ArnLike および ArnEquals の詳細については、「[Amazon リソースネーム \(ARN\) の条件演算子](#)ポリシー要素リファレンス」の「条件の種類」セクションの「」を参照してください。

たとえば、アクションの使用を制限して、指定した管理ポリシーのみを対象にすることもできます。これは、権限を付与するポリシーの Condition 要素にポリシー ARN を指定することで実行できます。たとえば、カスタマー管理ポリシーの ARN を指定するには、次のようにします。

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::123456789012:policy/POLICY-NAME"}
}
```

または、ポリシーの Condition 要素で AWS 管理ポリシーの ARN を指定することもできます。AWS 管理ポリシーの ARN では、以下の例に示すように、アカウント ID の代わりにポリシー ARN で aws という特別なエイリアスを使用します。

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"}
}
```

## リソースへのアクセスの制御

アイデンティティベースのポリシーまたはリソースベースのポリシーを使用してリソースにアクセスできるユーザーを制御できます。アイデンティティベースのポリシーでは、ポリシーをアイデンティティにアタッチし、そのアイデンティティがアクセスできるリソースを指定します。リソースベースのポリシーでは、制御するリソースにポリシーをアタッチします。ポリシーでは、リソースにアク



セスできるプリンシパルを指定します。ポリシーの両方のタイプの詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。

詳細については、以下のリソースを参照してください。

- プリンシパルにアタッチできる IAM ポリシーの作成の詳細については、「[IAM ポリシーの作成](#)」を参照してください。
- IAM ポリシーをプリンシパルにアタッチする方法については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。
- Amazon S3 は、バケットでリソースベースのポリシーの使用をサポートします。詳細については、「[バケットポリシーの例](#)」を参照してください。

リソース作成者であっても自動的にアクセス権限を有するわけではない

AWS アカウントのルートユーザー 認証情報を使用してサインインする場合、そのアカウントに属するリソースであらゆるアクションを実行する権限があります。ただし、それは IAM ユーザーには当てはまりません。IAM ユーザーはリソースを作成するためのアクセス許可を付与されることはありますが、そのユーザーの権限は、たとえ自ら作成したリソースに対するものであっても、明示的に付与された権限に限定されます。つまり、IAM ロールなどのリソースを作成するだけでは、そのロールを編集または削除するアクセス許可は自動的に与えられません。さらに、アクセス許可は、アカウント所有者またはユーザー権限を管理するアクセス許可を付与されたその他のユーザーによって、いつでも無効にすることができます。

## 特定のアカウント内のプリンシパルへのアクセスの制御

お客様は自らのアカウント内の IAM ユーザーに対し、お客様のリソースへのアクセス権限を直接付与できます。別のアカウントのユーザーがお客様のリソースへのアクセスを必要としている場合は、IAM ロールを作成できます。ロールは、アクセス許可を含むが、特定のユーザーに関連付けられていないエンティティです。これにより他のアカウントのユーザーはロールを引き受けて、ロールに割り当てられた権限に応じてリソースにアクセスできます。詳細については、「[所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供](#)」を参照してください。

### Note

一部のサービスは、[アイデンティティベースおよびリソースベースのポリシー](#) で説明されているリソースベースのポリシーをサポートしています (Amazon S3、Amazon SNS、Amazon SQS など)。これらのサービスでは、ロールを使用する代わりに、共有するリソース (バ

ケット、トピック、またはキュー)にポリシーをアタッチします。リソースベースのポリシーでは、AWSアカウントで、リソースへのアクセス許可を持ちます。

## タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御

次のセクションの情報を使用して、IAM ユーザーおよびロールにアクセスできるユーザーと、ユーザーとロールがアクセスできるリソースを制御します。他の IAM リソースを含め、他の AWS リソースへのアクセスを制御するための一般的な情報と例については、「[IAM リソースのタグ付け](#)」を参照してください。

### Note

タグキーとタグキー値での大文字と小文字の区別の詳細については、「[Case sensitivity](#)」を参照してください。

タグは、IAM リソースにアタッチするか、リクエストで渡すか、リクエストを行うプリンシパルにアタッチすることができます。IAM ユーザーまたはロールは、リソースとプリンシパルの両方にすることができます。たとえば、ユーザーのグループを一覧表示することをユーザーに許可するポリシーを記述できます。このオペレーションは、リクエストを行うユーザー (プリンシパル) に、表示しようとしているユーザーと同じ `project=blue` タグがある場合にのみ許可されます。この例では、ユーザーは、同じプロジェクトで作業している限り、自分自身を含む任意のユーザーのグループメンバシップを表示できます。

タグに基づいてアクセスを制御するには、ポリシーの[条件要素](#)でタグ情報を提供します。IAM ポリシーを作成するときは、IAM タグおよび関連付けられたタグ条件キーを使用して、以下のいずれかへのアクセスを制御できます。

- [リソース](#) – ユーザーまたはロールへのアクセスをそれらのタグに基づいて制御します。これを行うには、`aws:ResourceTag/key-name` 条件キーを使用して、リソースにアタッチする必要があるタグのキーバリューのペアを指定します。詳細については、「[AWS のリソースへのアクセスの制御](#)」を参照してください。
- [リクエスト](#) – IAM リクエストで渡すことができるタグを制御します。そのためには、`aws:RequestTag/key-name` 条件キーを使用して、IAM ユーザーまたはロールに対して追加、変更、または削除できるタグを指定します。このキーは、IAM リソースと他の AWS リソースでも

同じ方法で使用されます。詳細については、「[AWS リクエスト時のアクセスの制御](#)」を参照してください。

- [プリンシパル](#) – 個人の IAM ユーザーまたはロールにアタッチされたタグに基づき、リクエストを行っている個人 (プリンシパル) が実行できる操作を制御します。そのためには、aws:PrincipalTag/**key-name** 条件キーを使用して、リクエストを許可する前に、IAM ユーザーまたはロールにアタッチする必要のあるタグを指定します。
- [認証プロセスの一部](#) – aws:TagKeys 条件キーを使用して、特定のタグキーをリクエストまたはプリンシパルで使用できるかどうかを制御します。この場合、キーバリューは重要ではありません。このキーは、IAM およびその他の AWS サービスでも同様に動作します。ただし、IAM でユーザーにタグを付けると、これにより、プリンシパルが任意のサービスに対してリクエストを行うことができるかどうかも制御されます。詳細については、「[タグキーに基づいたアクセスの制御](#)」を参照してください。

ビジュアルエディタまたは JSON を使用するか、既存の管理ポリシーをインポートして、IAM ポリシーを作成できます。詳細については、「[IAM ポリシーの作成](#)」を参照してください。

#### Note

IAM ロールを引き受けるとき、またはユーザーをフェデレートするときに、[セッションタグ](#)を渡すこともできます。これらは、セッションの長さに対してのみ有効です。

## IAM プリンシパルへのアクセスの制御

ユーザーのアイデンティティに付けられたタグに基づき、プリンシパルが実行できる操作を制御できます。

この例では、同じプロジェクトで作業している限り、このアカウントのすべてのユーザーが自分自身を含むすべてのユーザーのグループメンバーシップを表示できるようにする ID ベースのポリシーを作成する方法を示しています。このオペレーションは、ユーザーのリソースタグと、プリンシパルのタグのタグキー project の値が同じである場合にのみ許可されます。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": "iam:ListGroupsWithUser",
    "Resource": "arn:aws:iam::111222333444:user/*",
    "Condition": {
      "StringEquals": {"aws:ResourceTag/project":
"${aws:PrincipalTag/project}"}
    }
  ]
}
```

## タグキーに基づいたアクセスの制御

IAM ポリシーでタグを使用して、リクエストまたはプリンシパルで特定のタグキーを使用できるかどうかを制御できます。

この例は、temporary キーを持つタグのみをユーザーから削除できる ID ベースのポリシーを作成する方法を示しています。このポリシーを使用するには、サンプルポリシーの########を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:UntagUser",
    "Resource": "*",
    "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": ["temporary"]}}
  ]
}
```

## タグを使用した AWS リソースへのアクセスの制御

タグ付けをサポートしている AWS リソース (IAM リソースを含む) へのアクセスを制御するには、タグを使用します。IAM ユーザーとロールにタグ付けして、アクセスできるユーザーとロールを制御することができます。IAM ユーザーとロールにタグ付けするには、「[IAM リソースのタグ付け](#)」を参照してください。さらに、以下の IAM リソースへのアクセスを制御できます。カスタマー管理ポリシー、IAM ID プロバイダー、インスタンスプロファイル、サーバー証明書、仮想 MFA デバイス。プリンシパルタグを持つ IAM ロールが、一致するタグを持つリソースにアクセスすることを許可するポリシーを作成およびテストするためのチュートリアルを表示するには、「[IAM チュートリ](#)

[アル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#) を参照してください。IAM ユーザーやロールにタグ付けせずに、他の AWS リソース (IAM リソースを含む) へのアクセスを制御するには、次のセクションの情報を使用します。

タグを使用して AWS リソースへのアクセスを制御する前に、AWS がアクセスを許可する方法を理解する必要があります。AWS はリソースの集合で構成されています。Amazon EC2 インスタンスはリソースです。Amazon S3 バケットはリソースです。AWS API、AWS CLI、または AWS Management Console を使用して、Amazon S3 でのバケットの作成などのオペレーションを実行できます。これを行う際、そのオペレーションのリクエストを送信します。リクエストでは、アクション、リソース、プリンシパルエンティティ (ユーザーまたはロール)、プリンシパルアカウント、および必要なリクエスト情報を指定します。これらのすべての情報により、コンテキストが提供されます。

次に、AWS はユーザー (プリンシパルエンティティ) が認証され (サインイン済み)、指定されたリソースで指定されたアクションの実行が許可されている (アクセス許可がある) ことを確認します。認可時、AWS は、リクエストのコンテキストに該当するすべてのポリシーをチェックします。通常、ポリシーは [JSON ドキュメント](#) として AWS に保存され、プリンシパルエンティティのアクセス許可を指定します。ポリシーのタイプと用途の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

AWS は、リクエストの各部分がポリシーで許可されている場合のみ、リクエストを許可します。図を表示して IAM インフラストラクチャの詳細について学習するには、「[IAM の仕組み](#)」を参照してください。IAM でリクエストの許可を決定する方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。

タグは、このプロセスにおけるもう 1 つの考慮事項です。これらは、リソースにアタッチするか、タグ付けをサポートするサービスへのリクエストで渡すことができるからです。タグに基づいてアクセスを制御するには、ポリシーの[条件要素](#)でタグ情報を提供します。AWS サービスがタグを使用したアクセスの制御をサポートしているかどうか確認するには、「[IAM と連携する AWS のサービス](#)」を参照し、「ABAC」列に「はい」と表示されているサービスを探します。サービスの名前を選択すると、そのサービスの認証とアクセスコントロールに関するドキュメントが表示されます。

その後、そのリソースのタグに基づいてリソースへのアクセスを許可または拒否する IAM ポリシーを作成できます。そのポリシーでは、タグ条件キーを使用して以下のいずれかへのアクセスを制御できます。

- [リソース](#) – それらのリソースのタグに基づいて、AWS サービスリソースへのアクセスを制御します。これを行うには、ResourceTag/**key-name** 条件キーを使用して、リソースにアタッチされたタグに基づいてリソースへのアクセスを許可するかどうか決定します。

- [リクエスト](#) – リクエストで渡すことができるタグを制御します。これを行うには、`aws:RequestTag/key-name` 条件キーを使用して、AWS リソースのタグ付けを行うリクエストで渡すことができるタグキーバリューのペアを指定します。
- [認証プロセスの一部](#) – `aws:TagKeys` 条件キーを使用して、特定のタグキーがリクエストに存在することができるかどうかを制御します。

JSON を使用するか、既存の管理ポリシーをインポートして、IAM ポリシーを視覚的に作成できます。詳細については、「[IAM ポリシーの作成](#)」を参照してください。

#### Note

一部のサービスでは、リソースを作成するアクションを使用するアクセス許可があれば、リソースを作成する際にタグを指定することができます。

## AWS のリソースへのアクセスの制御

IAM ポリシーで条件を使用して、そのリソースのタグに基づき、AWS リソースへのアクセスを制御できます。これを行うには、グローバルの `aws:ResourceTag/tag-key` 条件キー、またはサービス固有のキーを使用します。一部のサービスでは、このキーのサービス固有のバージョンのみがサポートされ、グローバルバージョンはサポートされていません。

#### Warning

ロールをタグ付けした後に、`iam:PassRole` アクションでポリシー内の `ResourceTag` 条件キーを使用してロールを渡せるユーザーを制御しないようにしてください。このアプローチでは信頼できる結果は得られません。ロールをサービスに渡すのに必要なアクセス許可の詳細については、「[AWS のサービスにロールを渡すアクセス権限をユーザーに付与する](#)」を参照してください。

この例では、Amazon EC2 インスタンスの起動または停止を許可する ID ベースのポリシーを作成する方法を示します。これらのオペレーションは、インスタンスのタグ `owner` がそのユーザーのユーザー名の値を含む場合に限り、許可されます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{  
  "Version": "2012-10-17",
```



```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:StartInstances",
      "ec2:StopInstances"
    ],
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:DescribeInstances",
    "Resource": "*"
  }
]
```

このポリシーはアカウントの IAM ユーザーにアタッチできます。richard-roe というユーザーが Amazon EC2 インスタンスを起動しようとした場合、そのインスタンスには Owner=richard-roe または owner=richard-roe というタグが付けられている必要があります。それ以外の場合、アクセスは拒否されます。条件キー名では大文字と小文字は区別されないため、タグキー Owner は Owner と owner に一致します。詳細については、「[IAM JSON ポリシー要素Condition](#)」を参照してください。

この例は、リソース ARN で team プリンシパルタグを使用する ID ベースポリシーを作成する方法を示しています。このポリシーでは、Amazon Simple Queue Service キューを削除する許可が付与されますが、キュー名がチーム名で始まり、-queue が続く場合に限られます。例えば、qa が team プリンシパルタグのチーム名なら qa-queue です。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AllQueueActions",
    "Effect": "Allow",
    "Action": "sqs:DeleteQueue",
    "Resource": "arn:aws:sqs:us-east-2:${aws:PrincipalTag/team}-queue"
  }
}
```

## AWS リクエスト時のアクセスの制御

IAM ポリシーの条件を使用して、AWS リソースにタグを適用するリクエストで渡すことができるタグのキー値のペアを制御することができます。

この例では、Amazon EC2 CreateTags アクションを使用してタグをインスタンスにアタッチできるようにする、ID ベースのポリシーを作成する方法を示しています。タグをアタッチできるのは、タグに environment キーと preprod または production の値が含まれている場合だけです。必要に応じて、ForAllValues 修飾子を aws:TagKeys 条件キーとともに使用して、リクエストでキー environment のみが許可されることを示します。これにより、environment の代わりに誤って Environment を使用するなど、ユーザーが他のキーを含めることがなくなります。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ]
      },
      "ForAllValues:StringEquals": {"aws:TagKeys": "environment"}
    }
  }
}
```

## タグキーに基づいたアクセスの制御

IAM ポリシーで条件を使用して、リクエストで特定のタグキーを使用できるかどうか制御できます。

ベストプラクティスとして、ポリシーでタグを使用してアクセスを制御する場合、[aws:TagKeys 条件キー](#)を使用することをお勧めします。タグをサポートする AWS のサービスでは、大文字小文字のみが異なる複数のタグキー名を作成できる可能性があります。たとえば、Amazon EC2 インスタンスに stack=production および Stack=test タグを付けるなどです。ポリシー条件のキー名では、大文字と小文字は区別されません。つまり、ポリシーの条件要素で "aws:ResourceTag/TagKey1": "Value1" で指定した場合、その条件は TagKey1 または tagkey1 という名前のリ



ソースタグキーに一致しますが、その両方には一致しません。大文字小文字のみが異なるキーを使用したタグの重複を防ぐには、aws:TagKeys 条件を使用して、ユーザーが適用できるタグキーを定義するか、AWS Organizations で利用できるタグポリシーを使用します。詳細については、「Organizations ユーザーガイド」の「[タグポリシー](#)」を参照してください。

この例では、Secrets Manager のシークレットの作成とタグ付けを許可する ID ベースのポリシーを作成する方法を示していますが、タグキー environment または cost-center のみを使用します。Null 条件を使用すると、リクエストにタグがない場合に条件が false と評価されます。

```
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:CreateSecret",
    "secretsmanager:TagResource"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "aws:TagKeys": "false"
    },
    "ForAllValues:StringEquals": {
      "aws:TagKeys": [
        "environment",
        "cost-center"
      ]
    }
  }
}
```

## IAM でのクロスアカウントのリソースへのアクセス

一部の AWS サービスでは、IAM を使用してリソースへのクロスアカウントアクセスを許可できます。これを行うには、共有するリソースにポリシーを直接アタッチするか、ロールをプロキシとして使用します。

リソースを直接共有するには、共有するリソースで[リソースベースのポリシー](#)がサポートされている必要があります。ロールの ID ベースのポリシーとは異なり、リソースベースのポリシーは、そのリソースにアクセスできるユーザー (プリンシパル) を指定します。

リソースベースのポリシーがサポートされていない別のアカウントのリソースにアクセスする場合は、ロールをプロキシとして使用します。

これらのポリシータイプの違いの詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。

#### Note

IAM ロールとリソースベースのポリシーは、単一のパーティション内のアカウント間でのみアクセスを委任します。例えば、標準 aws パーティションの米国西部 (北カリフォルニア) にアカウントがあるとします。aws-cn パーティションの中国にもアカウントがあります。中国のアカウントのリソースベースのポリシーを使用して、標準 AWS アカウントのユーザーにアクセスを許可することはできません。

## ロールを使用したクロスアカウントアクセス

すべての AWS のサービスがリソースベースのポリシーをサポートしているわけではありません。これらのサービスでは、複数のサービスへのクロスアカウントアクセスを提供する際に、クロスアカウント IAM ロールを使用して許可管理を一元化できます。クロスアカウント IAM ロールとは、別の AWS アカウントの IAM プリンシパルがそのロールを引き受けることを許可する [信頼ポリシー](#) を含む IAM ロールです。簡単に言えば、ある AWS アカウントで、特定のアクセス許可を別の AWS アカウントに委任するロールを作成できます。

ポリシーを IAM ID にアタッチする方法については、「[IAM ポリシーを管理する](#)」を参照してください。

#### Note

プリンシパルがロールのアクセス許可の一時的な使用のためにそのロールに切り替えた場合、プリンシパルは元のアクセス許可を放棄して、引き受けたロールに割り当てられたアクセス許可を引き継ぎます。

では、ユーザーアカウントにアクセスする必要がある APN パートナーソフトウェアに適用されるプロセス全体を見ていきましょう。

1. ユーザーは、APN パートナーが必要とする、Amazon S3 リソースへのアクセスを許可するポリシーを含む IAM ロールを、自分のアカウントに作成します。この例では、ロール名は APNPartner です。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::bucket-name"
    ]
  }
]
```

- 次に、ユーザーは、APNPartner ロールの[信頼ポリシー](#)に APN パートナーの AWS アカウント ID を提供して、パートナーの AWS アカウントがそのロールを引き受けることができるように指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::APN-account-ID:role/APN-user-name"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- ユーザーはロールの Amazon リソースネーム (ARN) を APN パートナーに渡します。ARN はロールの完全修飾名です。

```
arn:aws:iam::APN-ACCOUNT-ID:role/APNPartner
```

#### Note

マルチテナントの状況では、外部 ID を使用することをお勧めします。詳細については、「[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#)」を参照してください。

4. APN パートナーのソフトウェアがユーザーのアカウントにアクセスする必要がある場合、ソフトウェアはユーザーのアカウント内のロールの ARN を指定して AWS Security Token Service の [AssumeRole](#) API を呼び出します。STS は、ソフトウェアが処理を実行できるようにする一時的な AWS 認証情報を返します。

ロールを使用してクロスアカウントアクセスを付与する別の例については、「[所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供](#)」を参照してください。「[IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#)」の手順に従うこともできます。

## リソースベースのポリシーを使用したクロスアカウントアクセス

あるアカウントがリソースベースのポリシーを使用して別のアカウント経由でリソースにアクセスする場合、プリンシパルは引き続き信頼されたアカウントで動作するため、ロールのアクセス許可を受け取るためにプリンシパル自体のアクセス許可を放棄する必要はありません。つまり、プリンシパルは信頼されたアカウントのリソースに引き続きアクセスできるだけでなく、信頼されたアカウントのリソースにもアクセスできます。これは、他のアカウントに属する共有リソースから、また共有リソースへと情報をコピーするといったタスクにおいて便利です。

リソースベースのポリシーで指定できるプリンシパルには、アカウント、IAM ユーザー、フェデレーテッドユーザー、IAM ロール、引き受けたロールセッション、AWS サービスなどがあります。詳細については、「[プリンシパルの指定](#)」を参照してください。

信頼ゾーン (信頼された組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[外部エンティティと共有されているリソースを識別する](#)」を参照してください。

リソースベースのポリシーをサポートする AWS サービスの一部を以下に示します。プリンシパルではなくリソースへのアクセス許可ポリシーのアタッチをサポートする AWS サービスの数が増えている完全なリストについては、[IAM と連携する AWS のサービス](#) をご参照の上、[リソースベース] 列で [はい] のあるサービスをお探してください。

- Amazon S3 バケット – ポリシーはバケットにアタッチされますが、ポリシーによってバケットとバケット内のオブジェクトの両方へのアクセスが制御されます。Amazon S3 アクセス許可に関する詳細は、Amazon Simple Storage Service ユーザーガイドの「[Access Control](#)」を参照してください。状況によっては、Amazon S3 へのクロスアカウントアクセスにロールを使うのが最適な場合もあります。詳細については、Amazon Simple Storage Service ユーザーガイドの「[チュートリアル例](#)」を参照してください。

- Amazon Simple Notification Service (Amazon SNS) のトピック – 詳細については、「Amazon Simple Notification Service デベロッパーガイド」の「[Amazon SNS アクセスコントロールのケース例](#)」を参照してください。
- Amazon Simple Queue Service (Amazon SQS) キュー — 詳細については、[Amazon Simple Queue Service 開発者ガイド](#)の「付録:アクセスポリシー言語」を参照してください。

## リソースベースのポリシーでの AWS アクセス許可の委任

リソースがアカウントのプリンシパルにアクセス許可を付与する場合は、そのアクセス許可を特定の IAM ID に委任できます。ID とは、ユーザー、ユーザーのグループ、またはアカウント内のロールです。アクセス許可を委任するには、ポリシーを ID にアタッチします。リソース所有アカウントによって許可される最大数のアクセス許可を付与できます。

### Important

クロスアカウントアクセスでは、プリンシパルは ID ポリシーおよびリソースベースのポリシー内に Allow を必要とします。

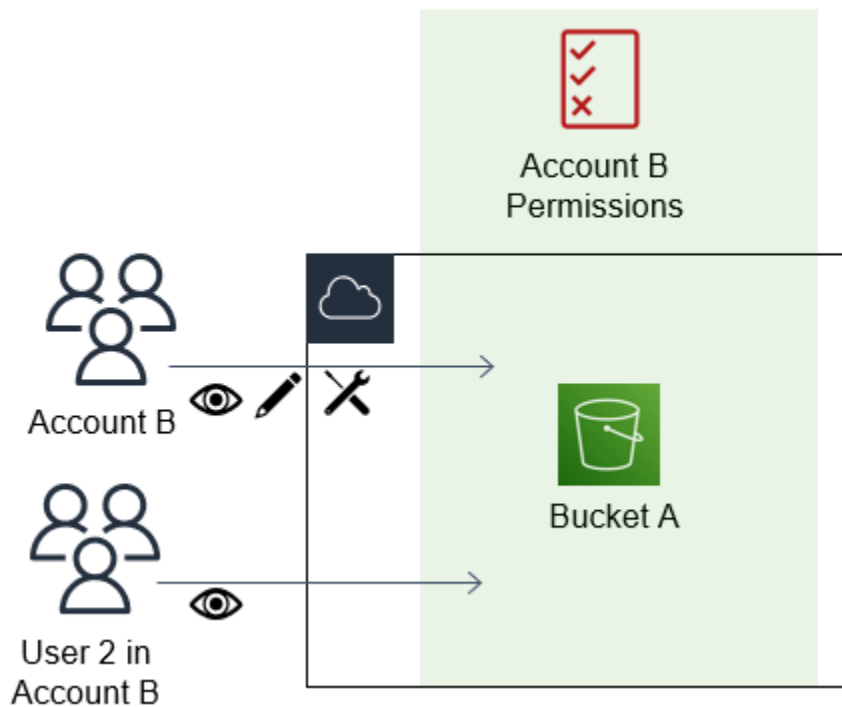
リソースベースのポリシーにより、アカウント内のすべてのプリンシパルがリソースへの完全な管理アクセスを許可するとします。すると、AWS アカウントで、プリンシパルへの完全アクセス、読み取り専用アクセス、またはその他の部分的なアクセスを委任できます。また、リソースベースのポリシーでリストアクセス許可のみが許可される場合は、リストアクセスのみを委任できます。アカウントが保持しているものよりも多くのアクセス許可を委任しようとしても、プリンシパルが保持できるのは一覧表示アクセスのみになります。

これらの決定方法の詳細については、「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。

### Note

IAM ロールとリソースベースのポリシーは、単一のパーティション内のアカウント間でのみアクセスを委任します。たとえば、標準 aws パーティションのアカウントと aws-cn パーティションのアカウントの間にクロスアカウントアクセスを追加することはできません。

たとえば、AccountA と AccountB を管理するとします。AccountA には、BucketA という名前の Amazon S3 バケットがあります。



1. AccountB のすべてのプリンシパルにバケット内のオブジェクトへのフルアクセスを許可するリソースベースのポリシーを BucketA にアタッチします。プリンシパルは、そのバケット内の任意のオブジェクトを作成、読み取り、または削除できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PrincipalAccess",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::AccountB:root"},
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::BucketA/*"
    }
  ]
}
```

AccountA は、リソースベースのポリシーで AccountB をプリンシパルとして指定することにより、AccountB に BucketA へのフルアクセスを許可します。その結果、AccountB は BucketA に対してあらゆるアクションを実行する権限が与えられるため、AccountB の管理者は AccountB のユーザーにアクセスを委任できるようになります。

AccountB ルートユーザーは、アカウントに付与されるすべてのアクセス許可を保持しています。したがって、ルートユーザーは BucketA へのフルアクセスを保持しています。

- AccountB で、User2 という名前の IAM ユーザーにポリシーをアタッチします。このポリシーにより、ユーザーは BucketA 内のオブジェクトへの読み取り専用アクセスが許可されます。つまり、User2 はオブジェクトを表示できますが、オブジェクトの作成、編集、または削除を行うことはできません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::BucketA/*"
    }
  ]
}
```

AccountB が委任できるアクセスの最大レベルは、アカウントに付与されるアクセスレベルです。この場合、リソースベースのポリシーにより AccountB へのフルアクセスが付与されますが、User2 は読み取り専用アクセスのみが付与されます。

AccountB の管理者は、User1 にはアクセス権を付与しません。デフォルトでは、ユーザーは明示的に付与されたアクセス許可以外のアクセス許可を保持していないため、User1 は BucketA にアクセスできません。

IAM では、プリンシパルがリクエストを行った時点でプリンシパルのアクセス許可が評価されます。ワイルドカード (\*) を使用してリソースへのフルアクセスをユーザーに付与すると、プリンシパルは AWS アカウントがアクセスできるすべてのリソースにアクセスできます。これは、ユーザーのポリシーの作成後に追加またはアクセスを得るリソースに対しても当てはまります。

前の例では、AccountB がすべてのアカウントのすべてのリソースへのフルアクセスを許可するポリシーを User2 にアタッチしていた場合、User2 は AccountB がアクセスできるすべてのリソースに自動的にアクセスできるようになります。これには、BucketA へのアクセスと AccountA のリソースベースのポリシーによって付与されたその他のリソースへのアクセスが含まれます。

アプリケーションやサービスへのアクセス権の付与など、ロールの複雑な使用方法の詳細については、「[ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス](#)」を参照してください。

#### Important

信頼できるエンティティにだけアクセスを許可し、必要最少レベルのアクセスを提供します。信頼されたエンティティが別の AWS アカウントである場合は常に、任意の IAM プリンシパルにリソースへのアクセスを許可できます。信頼された AWS アカウントは、アクセス権を付与された範囲でのみアクセス権を委任できますが、アカウント自身に付与された範囲を超えるアクセス権を委任することはできません。

権限、ポリシー、ポリシーを作成するのに使用するアクセス許可ポリシー言語の詳細については、「[AWS リソースのアクセス管理](#)」を参照してください。

## 転送アクセスセッション

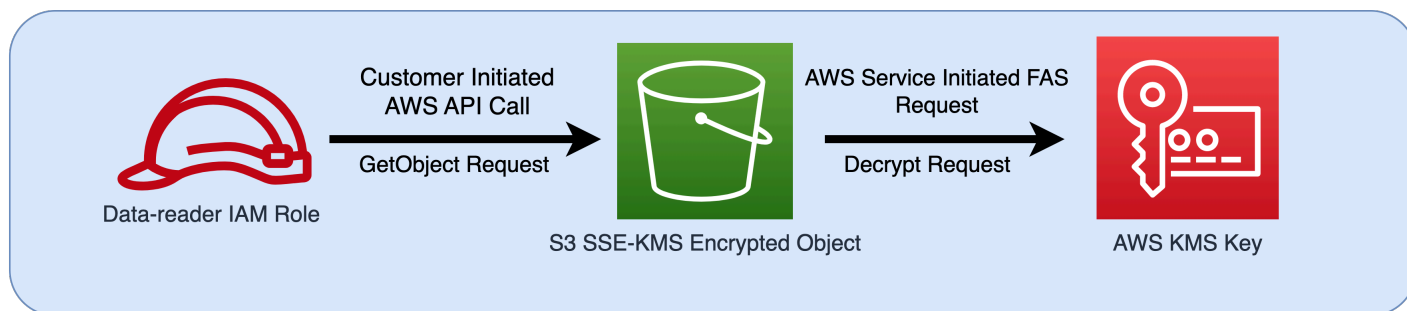
転送アクセスセッション (FAS) は、AWS サービスがユーザーに代わってリクエストを行ったときに ID、権限、セッション属性を渡すために AWS サービスが使用する IAM テクノロジーです。FAS は、AWS サービスを呼び出す ID の権限を使用し、AWS サービスの ID と組み合わせて、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、他の AWS サービスやリソースとのやり取りを完了する必要があるリクエストをサービスが受信した後、IAM プリンシパルに代わって AWS サービスに対してのみ行われます。FAS リクエストが行われた場合:

- IAM プリンシパルから最初のリクエストを受け取るサービスは、IAM プリンシパルの権限を確認します。
- 後続の FAS リクエストを受信するサービスも、同じ IAM プリンシパルの権限を確認します。

例えば、[SSE-KMS](#) を使用してオブジェクトを暗号化した場合、Amazon S3 は FAS を使用してオブジェクトを復号化するための呼び出しを AWS Key Management Service に対して行います。SSE-KMS 暗号化オブジェクトをダウンロードするとき、data-reader という名前のロールは Amazon S3 に対してオブジェクトの GetObject を呼び出し、直接 AWS KMS は呼び出しません。GetObject リクエストを受け取り、data-reader を承認すると、Amazon S3 は Amazon S3 オブジェクトを復号化するために FAS リクエストを AWS KMS に送信します。KMS は FAS リクエストを受信すると、ロールの権限を確認し、data-reader が KMS キーに対して正しい権限を持っている場合にのみ復号化リクエストを承認します。Amazon S3 と AWS KMS の両方へのリクエストは、ロールの権限を使



用して承認され、data-reader が Amazon S3 オブジェクトと AWS KMS キーの両方に対する権限を持っている場合にのみ成功します。

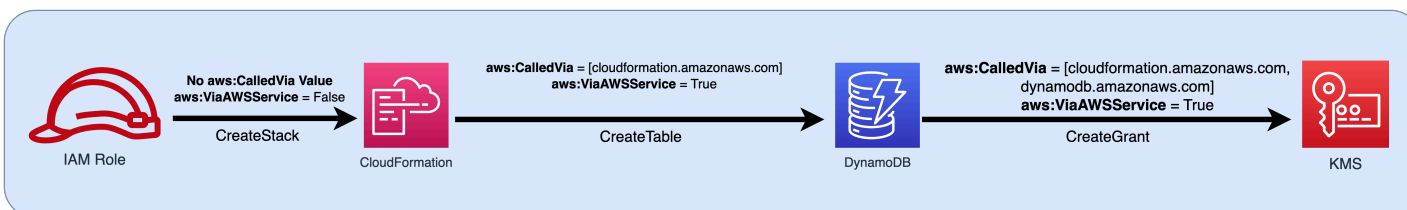


### Note

FAS リクエストを受け取ったサービスによって、追加の FAS リクエストを行うことができます。このような場合、要求元のプリンシパルは、FAS によって呼び出されるすべてのサービスに対する権限を持っている必要があります。

## FAS リクエストと IAM ポリシー条件

FAS リクエストが行われると、[aws:CalledVia](#)、[aws:CalledViaFirst](#)、および [aws:CalledViaLast](#) 条件キーには、FAS 呼び出しを開始したサービスのサービスプリンシパルが入力されます。[aws:ViaAWSService](#) 条件キーの値は、FAS リクエストが行われると常に true に設定されます。以下の図では、CloudFormation への直接のリクエストには、aws:CalledVia または aws:ViaAWSService 条件キーが設定されていません。CloudFormation と DynamoDB がロールに代わってダウンストリームの FAS リクエストを行うと、これらの条件キーの値が入力されます。



ソース IP アドレスまたはソース VPC をテストする条件キーを含む拒否ポリシーステートメントによって拒否される場合に FAS リクエストを行えるようにするには、条件キーを使用して拒否ポリシーで FAS リクエストの例外を設定する必要があります。これは、aws:ViaAWSService 条件キーを使用することですべての FAS リクエストに対して実行できます。特定の AWS サービスのみが FAS リクエストを行えるようにするには、aws:CalledVia を使用してください。

**⚠ Important**

VPC エンドポイントを介して最初のリクエストが行われた後に FAS リクエストが行われた場合、最初のリクエストの [aws:SourceVpce](#)、[aws:SourceVpc](#)、および [aws:VpcSourceIp](#) の条件キー値は FAS リクエストでは使用されません。aws:VPCSourceIP または aws:SourceVPCE を使用してポリシーを作成し、条件付きでアクセスを許可する場合は、aws:ViaAWSService または aws:CalledVia を使用して FAS リクエストを許可する必要があります。パブリック AWS サービスエンドポイントが最初のリクエストを受信した後に FAS リクエストが行われると、それ以降の FAS リクエストは同じ aws:SourceIP 条件キー値で行われます。

**例: VPC からの、または FAS を使用した Amazon S3 アクセスの許可**

次の IAM ポリシーの例では、Amazon S3 GetObject リクエストと Athena リクエストは、*example\_vpc* にアタッチされた VPC エンドポイントから送信された場合、またはリクエストが Athena によって作成された FAS リクエストである場合にのみ許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowMyIPs",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "athena:StartQueryExecution",
        "athena:GetQueryResults",
        "athena:GetWorkGroup",
        "athena:StopQueryExecution",
        "athena:GetQueryExecution"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceVPC": [
            "example_vpc"
          ]
        }
      }
    }
  ],
}
```

```
{
  "Sid": "OnlyAllowFAS",
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*"
  ],
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "aws:CalledVia": "athena.amazonaws.com"
    }
  }
}
```

条件キーを使用して FAS アクセスを許可するその他の例については、「[data perimeter example policy repo](#)」を参照してください。

## IAM アイデンティティベースのポリシーの例

[ポリシー](#)は AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、IAM プリンシパル (ユーザーまたはロール) によってリクエストが行われると、それらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。通常、ポリシーは、IAM エンティティ (ユーザー、ユーザーのグループ、ロール) にアタッチされている JSON ドキュメントとして AWS に保存されます。アイデンティティベースのポリシーには、AWS 管理ポリシー、カスタマー管理ポリシー、およびインラインポリシーがあります。これらの例の JSON ポリシードキュメントを使用して IAM ポリシーを作成する方法については、「[the section called “JSON エディターを使用したポリシーの作成”](#)」を参照してください。

デフォルトではすべてのリクエストが拒否されるため、その ID がアクセスするサービス、アクション、リソースへのアクセスを許可する必要があります。IAM コンソールで指定したアクションを完了するためのアクセスも許可する場合は、追加のアクセス許可を提供する必要があります。

以下のポリシーのライブラリは、IAM ID のアクセス許可を定義する参考になります。必要なポリシーを見つけたら、[View this policy (このポリシーを表示)] を選択してそのポリシーの JSON を表示します。JSON のポリシードキュメントをテンプレートとして使用して、独自のポリシーを作成できます。

**Note**

このリファレンスガイドに含めるポリシーを送信する場合は、このページの下部にある [フィードバック] ボタンを使用します。

## ポリシーの例: AWS

- 特定の日付範囲内のアクセスを許可します。 ([このポリシーを表示。](#))
- AWS リージョンの有効化と無効化 ([このポリシーを表示。](#))
- MFA で認証されたユーザーが [セキュリティ認証情報] ページで自分の認証情報を管理できるようにします。 ([このポリシーを表示。](#))
- 指定した日付の範囲内で MFA を使用したときに特定のアクセスを許可する。 ([このポリシーを表示。](#))
- ユーザーが [セキュリティ認証情報] ページで自分の認証情報を管理できるようにします。 ([このポリシーを表示。](#))
- ユーザーが [セキュリティ認証情報] ページで自分の MFA デバイスを管理できるようにします。 ([このポリシーを表示。](#))
- ユーザーが [セキュリティ認証情報] ページで自分のパスワードを管理できるようにします。 ([このポリシーを表示。](#))
- ユーザーが [セキュリティ認証情報] ページで自分のパスワード、アクセスキー、および SSH パブリックキーを管理できるようにします。 ([このポリシーを表示。](#))
- リクエストされたリージョンに基づいて、AWS へのアクセスを拒否する ([このポリシーを表示。](#))
- 送信元 IP に基づいて AWS へのアクセスを拒否する ([このポリシーを表示。](#))

## ポリシーの例: AWS Data Exchange

- AWS Data Exchange 以外のアカウント外の Amazon S3 リソースへのアクセスを拒否します。 ([このポリシーを表示。](#))

## ポリシーの例: AWS Data Pipeline

- ユーザーが作成していないパイプラインへのアクセスを拒否する ([このポリシーを表示。](#))

## ポリシーの例: Amazon DynamoDB

- 特定の Amazon DynamoDB テーブルへのアクセスを許可する ([このポリシーを表示](#))。
- 特定の Amazon DynamoDB 属性へのアクセスを許可する ([このポリシーを表示](#))。
- Amazon Cognito ID に基づいて Amazon DynamoDB への項目レベルのアクセスを許可する ([このポリシーを表示](#))。

## ポリシーの例: Amazon EC2

- タグに基づいて Amazon EC2 インスタンスに Amazon EBS ボリュームをアタッチまたはデタッチすることを許可する ([このポリシーを表示](#))。
- 特定のサブネットで、プログラムおよびコンソールで Amazon EC2 インスタンスを起動することを許可する ([このポリシーを表示](#))
- 特定の VPC に関連付けられた Amazon EC2 セキュリティグループを、プログラムによりコンソールで管理することを許可する ([このポリシーを表示](#))。
- ユーザーがタグ付けした Amazon EC2 インスタンスをプログラムによりコンソールで開始や停止を行うことを許可する ([このポリシーを表示](#))。
- Amazon EC2 インスタンスを、リソースおよびプリンシパルのタグに基づき、プログラムを使用する、およびコンソールで開始または停止することを許可する ([このポリシーを表示](#))。
- リソースとプリンシパルのタグが一致すると、Amazon EC2 インスタンスの開始または停止を許可する ([このポリシーを表示](#))。
- 特定のリージョンでの完全な Amazon EC2 アクセスをプログラムによりコンソールで許可する ([このポリシーを表示](#)。)
- プログラムおよびコンソールで特定の Amazon EC2 インスタンスの起動または停止、および特定のセキュリティグループの変更を許可する ([このポリシーを表示](#))
- MFA なしで特定の Amazon EC2 オペレーションへのアクセスを拒否する ([このポリシーを表示](#))。
- Amazon EC2 インスタンスの削除を特定の IP アドレス範囲に制限する ([このポリシーを表示](#))

## ポリシーの例: AWS Identity and Access Management (IAM)

- Policy Simulator API へのアクセスを許可する ([このポリシーを表示](#))。
- Policy Simulator コンソールへのアクセスを許可する ([このポリシーを表示](#))。
- 特定のタグを持つロールを引き受けることをプログラムによりコンソールで許可する ([このポリシーを表示](#))。

- 複数のサービスへのアクセスをプログラムによりコンソールで許可および拒否する ([このポリシーを表示](#))。
- 特定のタグを、別の特定のタグ、プログラム、およびコンソールで IAM ユーザーに追加することを許可する ([このポリシーを表示](#))。
- 任意の IAM ユーザーまたはロールに、特定のタグをプログラムによりコンソールで追加することを許可する ([このポリシーを表示](#))。
- 特定のタグでのみ新規ユーザーを作成することを許可する ([このポリシーを表示](#))。
- IAM 認証情報レポートの生成および取得を許可する ([このポリシーを表示](#))。
- グループメンバーをプログラムによりコンソールで管理することを許可する ([このポリシーを表示](#))。
- 特定のタグの管理を許可する ([このポリシーを表示](#))。
- IAM ロールを特定のサービスに渡すことを許可する ([このポリシーを表示](#))。
- レポートなしでの IAM コンソールへの読み取り専用アクセスを許可する ([このポリシーを表示](#))。
- IAM コンソールへの読み取り専用アクセスを許可する ([このポリシーを表示](#))。
- 特定のユーザーによるグループの管理をプログラムによりコンソールで許可する ([このポリシーを表示](#))。
- アカウントのパスワード要件の設定をプログラムによりコンソールで許可する ([このポリシーを表示](#))。
- 特定のパスがあるユーザーに Policy Simulator API の使用を許可する ([このポリシーを表示](#))。
- 特定のパスがあるユーザーに Policy Simulator コンソールの使用を許可する ([このポリシーを表示](#))。
- IAM: ユーザーに MFA デバイスの自己管理を許可する ([このポリシーを表示](#)。)
- IAM ユーザーが自分の認証情報をプログラムまたはコンソールで設定することを許可する。 ([このポリシーを表示](#)。)
- IAM コンソールで AWS Organizations ポリシーのサービスの最終アクセス情報を表示することを許可する。 ([このポリシーを表示](#)。)
- IAM ユーザー、グループ、またはロールに適用できる管理ポリシーを制限する ([このポリシーを表示](#))。
- アカウント内の IAM ポリシーにのみアクセスを許可します ([このポリシーを表示](#))。

## ポリシーの例: AWS Lambda

- Amazon DynamoDB テーブルにアクセスする AWS Lambda 関数を許可する ([このポリシーを表示](#))。

## ポリシーの例: Amazon RDS

- Amazon RDS は、特定のリージョン内で RDS データベースへのフルアクセスを許可します。 ([このポリシーを表示](#)。)
- Amazon RDS データベースをプログラムおよびコンソールで復元することを許可する ([このポリシーを表示](#))
- タグ所有者にタグ付けした Amazon RDS リソースへのフルアクセスを許可する ([このポリシーを表示](#))

## ポリシーの例: Amazon S3

- Amazon Cognito ユーザーが自分の Amazon S3 バケットのオブジェクトにアクセスすることを許可する ([このポリシーを表示](#))
- フェデレーションユーザーに Amazon S3 内の自分のホームディレクトリへのプログラマチックなアクセスとコンソールを使用したアクセスを許可する ([このポリシーを表示](#))。
- 完全な S3 アクセスを許可しても、管理者が過去 30 分以内に MFA を使用してサインインしていない場合は本番稼働用バケットへのアクセスを明示的に拒否する ([このポリシーを表示](#))。
- IAM ユーザーが Amazon S3 の自分のホームディレクトリにプログラムおよびコンソールでアクセスすることを許可する ([このポリシーを表示](#))
- ユーザーに 1 つの Amazon S3 バケットの管理を許可し、他のすべての AWS アクションおよびリソースを拒否する ([このポリシーを表示](#))。
- 特定の Amazon S3 バケットへの Read と Write アクセスを許可する ([このポリシーを表示](#))
- 特定の Amazon S3 バケットにプログラムおよびコンソールで Read および Write アクセスを許可する ([このポリシーを表示](#))

## AWS: 日付と時刻に基づいてアクセスを許可します

この例では、日付と時刻に基づいてアクションへのアクセスを許可する ID ベースポリシーを作成する方法を示します。このポリシーは、2020 年 4 月 1 日から 2020 年 6 月 30 日 (UTC) の間に発生するアクションへのアクセスを制限します。このポリシーでは、AWS API または AWS CLI から、こ



のアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

IAM ポリシーの Condition ブロック内で複数の条件を使用する方法については、「[条件内の複数の値](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "service-prefix:action-name",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}
      }
    }
  ]
}
```

#### Note

日付条件演算子でポリシー変数を使用することはできません。詳細については、「[条件の要素](#)」を参照してください。

## AWS: AWS リージョンの有効化と無効化を許可する

この例は、管理者がアジアパシフィック (香港) リージョン (ap-east-1) を有効化および無効化できるようにする ID ベースポリシーの作成方法を示しています。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。この設定は、AWS Management Console の [アカウント設定] ページに表示されます。このページには、アカウント管理者のみが表示および管理する必要がある機密アカウントレベルの情報が含まれています。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。



**⚠ Important**

デフォルトで有効になっているリージョンを有効または無効にすることはできません。デフォルトで無効になっているリージョンのみを含めることができます。詳細については、「AWS 全般のリファレンス」の「[AWS リージョンの管理](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableDisableHongKong",
      "Effect": "Allow",
      "Action": [
        "account:EnableRegion",
        "account:DisableRegion"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {"account:TargetRegion": "ap-east-1"}
      }
    },
    {
      "Sid": "ViewConsole",
      "Effect": "Allow",
      "Action": [
        "account:ListRegions"
      ],
      "Resource": "*"
    }
  ]
}
```

## AWS: MFA で認証された IAM ユーザーが [セキュリティ認証情報] ページで自分の認証情報を管理できるようにします

この例は、[多要素認証 \(MFA\)](#) を使用して認証された IAM ユーザーが、[セキュリティ認証情報] ページで自分の認証情報を管理できるようにする ID ベースポリシーを作成する方法を示しています。この AWS Management Console ページには、アカウント ID や正規ユーザー ID などのアカウント情報が表示されます。ユーザーは、自分のパスワード、アクセスキー、MFA デバイス、X.509 証明

書、SSH キー、および Git 認証情報を表示および編集することもできます。この例では、必要なアクセス許可がポリシーに含まれているページ上のすべての情報を表示および編集する手順について説明します。また、AWS で他のオペレーションを実行する前に、ユーザーに MFA を使用した設定と認証を要求します。ユーザーに MFA を使用せずに自らの認証情報を管理することを許可するには、「[AWS: IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

ユーザーが [セキュリティ認証情報] ページにアクセスする方法については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

### Note

- このポリシー例では、初めて AWS Management Console にサインインする際のパスワードのリセットをユーザーに許可していません。新しいユーザーがサインインするまで、当該ユーザーにアクセス権を許可しないことをお勧めします。詳細については、「[IAM ユーザーを安全に作成するにはどうすればよいですか?](#)」を参照してください。また、これにより失効したパスワードを持つユーザーは、サインイン中にパスワードをリセットできなくなります。この操作を許可するには、`iam:ChangePassword` と `iam:GetAccountPasswordPolicy` をステートメント `DenyAllExceptListedIfNoMFA` に追加します。ただし、ユーザーが多要素認証 (MFA) なしで自分のパスワードを変更できるようになると、セキュリティ上のリスクが生じる可能性があるためこれを推奨しません。
- このポリシーをプログラムによるアクセスに使用する場合は、[GetSessionToken](#) を呼び出して MFA で認証します。詳細については、[MFA 保護 API アクセスの設定](#) を参照してください。

### このポリシーで行うこと

- この `AllowViewAccountInfo` ステートメントでは、ユーザーにアカウントレベルの情報を表示します。これらのアクセス許可は、リソース ARN をサポートしていないか、または指定する必要がないため、独自のステートメントに含まれている必要があります。代わりに `"Resource" : "*"`  を指定するアクセス許可を使用します。このステートメントには、ユーザーが特定の情報を表示できるようにする以下のアクションが含まれています。
- `GetAccountPasswordPolicy` - IAM ユーザーパスワードを変更しながら、アカウントのパスワード要件を表示します。

- `ListVirtualMFADevices` – ユーザーに対して有効になっている仮想 MFA デバイスに関する詳細を表示します。
- `AllowManageOwnPasswords` ステートメントを使用すると、ユーザーは自分のパスワードを変更できます。このステートメントには `GetUser` アクションも含まれています。これは、[My security credentials] (セキュリティ認証情報) ページのほとんどの情報を表示するために必要です。
- この `AllowManageOwnAccessKeys` ステートメントにより、ユーザーは自分のアクセスキーを作成、更新、削除できます。ユーザーは指定されたアクセスキーの最後の使用時の情報を取得することもできます。
- この `AllowManageOwnSigningCertificates` ステートメントにより、ユーザーは自分のデジタル署名用証明書をアップロード、更新、削除できます。
- この `AllowManageOwnSSHPublicKeys` ステートメントにより、ユーザーは自分の CodeCommit の SSH パブリックキーをアップロード、更新、削除できます。
- `AllowManageOwnGitCredentials` ステートメントにより、ユーザーは自分の CodeCommit の Git 認証情報をアップロード、更新、削除できます。
- この `AllowManageOwnVirtualMFADevice` ステートメントにより、ユーザーは自分の仮想 MFA デバイスを作成できます。このステートメントのリソース ARN によって、ユーザーが任意の名前の MFA デバイスを作成できますが、ポリシー内の他のステートメントでは、ユーザーはデバイスを現在サインインしているユーザーにしかアタッチできません。
- この `AllowManageOwnUserMFA` ステートメントでは、ユーザーは自分のユーザーの仮想、U2F、またはハードウェア MFA デバイスを表示または管理できます。このステートメントのリソース ARN は、ユーザー自身の IAM ユーザーにのみアクセスを許可します。ユーザーは他のユーザーの MFA デバイスを表示または管理することはできません。
- `DenyAllExceptListedIfNoMFA` ステートメントは、ユーザーが MFA でサインインしていない場合のみ、いくつかのリストされたアクションを除いて、すべての AWS のサービスのすべてのアクションへのアクセスを拒否します。このステートメントでは、"Deny" と "NotAction" の組み合わせを使用して、表示されていないすべてのアクションへのアクセスを明示的に拒否しています。リストされている項目は、このステートメントによって拒否または許可されていません。ただし、アクションはポリシー内の他のステートメントによって許可されています。このステートメントのロジックの詳細については、「[Deny での NotAction の使用](#)」を参照してください。ユーザーが MFA でサインインしている場合、Condition テストは失敗し、このステートメントはアクションを拒否しません。この場合、ユーザーの他のポリシーまたは文によってユーザーのアクセス許可が決まります。

このステートメントは、ユーザーが MFA にサインインしていないときに、リストされているアクションのみを実行できることを保証します。さらに、他のステートメントまたはポリシーがそれらのアクションへのアクセスを許可している場合にのみ、リストされているアクションを実行できません。iam:ChangePassword アクションは MFA 認証なしには許可されないため、サインイン時にユーザーがパスワードを作成することはできません。

...IfExists バージョンの Bool 演算子により、[aws:MultiFactorAuthPresent](#) キーが見つからない場合、条件は必ず true を返します。つまり、アクセスキーなどの長期認証情報を使用して API にアクセスするユーザーは IAM 以外の API オペレーションへのアクセスを拒否されます。

このポリシーでは、IAM コンソールで [Users] (ユーザー) ページを表示したり、そのページを使用して自分のユーザー情報にアクセスすることはできません。これを許可するには、iam:ListUsers アクションを AllowViewAccountInfo ステートメントと DenyAllExceptListedIfNoMFA ステートメントに追加します。また、ユーザーが自分のユーザーページで自分のパスワードを変更することはできません。これを許可するには、iam:GetLoginProfile および iam:UpdateLoginProfile アクションを AllowManageOwnPasswords ステートメントに追加します。ユーザーが MFA を使用してサインインしなくても自分のユーザーページから自分のパスワードを変更できるようにするには、DenyAllExceptListedIfNoMFA ステートメントに iam:UpdateLoginProfile アクションを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:ListVirtualMFADevices"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswords",
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
      ],
    }
  ]
}
```

```
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnAccessKeys",
    "Effect": "Allow",
    "Action": [
      "iam:CreateAccessKey",
      "iam>DeleteAccessKey",
      "iam>ListAccessKeys",
      "iam:UpdateAccessKey",
      "iam:GetAccessKeyLastUsed"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnSigningCertificates",
    "Effect": "Allow",
    "Action": [
      "iam>DeleteSigningCertificate",
      "iam>ListSigningCertificates",
      "iam:UpdateSigningCertificate",
      "iam:UploadSigningCertificate"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnSSHPublicKeys",
    "Effect": "Allow",
    "Action": [
      "iam>DeleteSSHPublicKey",
      "iam:GetSSHPublicKey",
      "iam>ListSSHPublicKeys",
      "iam:UpdateSSHPublicKey",
      "iam:UploadSSHPublicKey"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnGitCredentials",
    "Effect": "Allow",
    "Action": [
      "iam:CreateServiceSpecificCredential",
      "iam>DeleteServiceSpecificCredential",
      "iam>ListServiceSpecificCredentials",
```

```
        "iam:ResetServiceSpecificCredential",
        "iam:UpdateServiceSpecificCredential"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "AllowManageOwnVirtualMFADevice",
    "Effect": "Allow",
    "Action": [
        "iam:CreateVirtualMFADevice"
    ],
    "Resource": "arn:aws:iam::*:mfa/*"
},
{
    "Sid": "AllowManageOwnUserMFA",
    "Effect": "Allow",
    "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "DenyAllExceptListedIfNoMFA",
    "Effect": "Deny",
    "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:GetMFADevice",
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice",
        "sts:GetSessionToken"
    ],
    "Resource": "*",
    "Condition": {
        "BoolIfExists": {
            "aws:MultiFactorAuthPresent": "false"
        }
    }
}
}
```

```

]
}

```

## AWS: 特定の日付内で MFA を使用する特定のアクセスを許可する

この例は、論理的な AND を使用して評価される複数の条件を使用する ID ベースポリシーを作成する方法を示しています。これにより、SERVICE-NAME-1 という名前のサービスにフルアクセスでき、ACTION-NAME-A という名前のサービスの ACTION-NAME-B および SERVICE-NAME-2 アクションにアクセスすることができます。これらのアクションは、[多要素認証 \(MFA\)](#) を使用してユーザーが認証された場合にのみ許可されます。アクセスは、2017 年 7 月 1 日から 2017 年 12 月 31 日 (UTC、7 月 1 日と 12 月 31 日を含む) までの間に発生するアクションに限定されます。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

IAM ポリシーの Condition ブロック内で複数の条件を使用する方法については、「[条件内の複数の値](#)」を参照してください。

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "service-prefix-1:*",
      "service-prefix-2:action-name-a",
      "service-prefix-2:action-name-b"
    ],
    "Resource": "*",
    "Condition": {
      "Bool": {"aws:MultiFactorAuthPresent": true},
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
  }
}

```

## AWS: IAM ユーザーが [セキュリティ認証情報] ページで自分の認証情報を管理できるようにします

この例は、IAM ユーザーが [セキュリティ認証情報] ページで自分のすべての認証情報を管理できるようにする ID ベースポリシーを作成する方法を示しています。この AWS Management Console

ページには、アカウント ID や正規ユーザー ID などのアカウント情報が表示されます。ユーザーは、自分のパスワード、アクセスキー、X.509 証明書、SSH キー、および Git 認証情報を表示および編集することもできます。この例では、ユーザーの MFA デバイス以外の必要なアクセス許可がポリシーに含まれているページ上のすべての情報を表示および編集する手順について説明します。ユーザーが MFA を使用して自らのすべての認証情報を管理することを許可するには、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

ユーザーが [セキュリティ認証情報] ページにアクセスする方法については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

### このポリシーで行うこと

- この AllowViewAccountInfo ステートメントでは、ユーザーにアカウントレベルの情報を表示します。これらのアクセス許可は、リソース ARN をサポートしていないか、または指定する必要がないため、独自のステートメントに含まれている必要があります。代わりに "Resource" : "\*" を指定するアクセス許可を使用します。このステートメントには、ユーザーが特定の情報を表示できるようにする以下のアクションが含まれています。
- GetAccountPasswordPolicy – IAM ユーザーパスワードを変更しながら、アカウントのパスワード要件を確認します。
- GetAccountSummary – アカウント ID とアカウントの表示 [正規ユーザー ID](#)。
- AllowManageOwnPasswords ステートメントを使用すると、ユーザーは自分のパスワードを変更できます。このステートメントには GetUser アクションも含まれています。これは、[My security credentials] (セキュリティ認証情報) ページのほとんどの情報を表示するために必要です。
- この AllowManageOwnAccessKeys ステートメントにより、ユーザーは自分のアクセスキーを作成、更新、削除できます。ユーザーは指定されたアクセスキーの最後の使用時の情報を取得することもできます。
- この AllowManageOwnSigningCertificates ステートメントにより、ユーザーは自分のデジタル署名用証明書をアップロード、更新、削除できます。
- この AllowManageOwnSSHPublicKeys ステートメントにより、ユーザーは自分の CodeCommit の SSH パブリックキーをアップロード、更新、削除できます。
- この AllowManageOwnGitCredentials ステートメントにより、ユーザーは自分の CodeCommit の Git 認証情報をアップロード、更新、削除できます。



このポリシーでは、ユーザーは自分の MFA デバイスを表示または管理できません。このポリシーでは、IAM コンソールで [Users] (ユーザー) ページを表示したり、そのページを使用して自分のユーザー情報にアクセスすることもできません。これを許可するには、iam:ListUsers アクションを AllowViewAccountInfo ステートメントに追加します。また、ユーザーが自分のユーザーページで自分のパスワードを変更することはできません。これを許可するには、iam:CreateLoginProfile、iam>DeleteLoginProfile、iam:GetLoginProfile、および iam:UpdateLoginProfile アクションを AllowManageOwnPasswords ステートメントに追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswords",
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam>DeleteAccessKey",
        "iam:ListAccessKeys",
        "iam:UpdateAccessKey",
        "iam:GetAccessKeyLastUsed"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ],
}
```

```
{
  "Sid": "AllowManageOwnSigningCertificates",
  "Effect": "Allow",
  "Action": [
    "iam:DeleteSigningCertificate",
    "iam:ListSigningCertificates",
    "iam:UpdateSigningCertificate",
    "iam:UploadSigningCertificate"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnSSHPublicKeys",
  "Effect": "Allow",
  "Action": [
    "iam:DeleteSSHPublicKey",
    "iam:GetSSHPublicKey",
    "iam:ListSSHPublicKeys",
    "iam:UpdateSSHPublicKey",
    "iam:UploadSSHPublicKey"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnGitCredentials",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceSpecificCredential",
    "iam:DeleteServiceSpecificCredential",
    "iam:ListServiceSpecificCredentials",
    "iam:ResetServiceSpecificCredential",
    "iam:UpdateServiceSpecificCredential"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
}
]
```

## AWS: MFA で認証された IAM ユーザーが [セキュリティ認証情報] ページで自分の MFA デバイスを管理できるようにします

この例は、[多要素認証 \(MFA\)](#) によって認証された IAM ユーザーが [セキュリティ認証情報] ページで自分の MFA デバイスを管理できるようにする ID ベースポリシーを作成する方法を示しています。

この AWS Management Console ページにはアカウントとユーザー情報が表示されますが、ユーザーは自分の MFA デバイスを表示および編集することしかできません。ユーザーに MFA を使用して自らのすべての認証情報を管理することを許可するには、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

#### Note

このポリシーを設定した IAM ユーザーが MFA 認証されていない場合、このポリシーは、MFA を使用した認証に必要なアクションを除いて、すべての AWS アクションへのアクセスを拒否します。AWS CLI および AWS API を使用するには、IAM ユーザーはまず AWS STS [GetSessionToken](#) オペレーションを使用して MFA トークンを取得し、次にそのトークンを使用して目的のオペレーションを認証する必要があります。リソースベースポリシーや他の ID ベースポリシーなどの他のポリシーを使用して、他のサービスでのアクションを許可できます。IAM ユーザーが MFA 認証されていない場合、同ポリシーはそのアクセスを拒否します。

ユーザーが [\[セキュリティ認証情報\] ページ](#) にアクセスする方法については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

#### このポリシーで行うこと

- この AllowViewAccountInfo ステートメントでは、ユーザーに対して有効になっている仮想 MFA デバイスに関する詳細を表示することを許可します。このアクセス許可は、リソース ARN の指定をサポートしていないため、独自のステートメント内になければなりません。または "Resource" : "\*" を指定する必要があります。
- この AllowManageOwnVirtualMFADevice ステートメントにより、ユーザーは自分の仮想 MFA デバイスを作成できます。このステートメントのリソース ARN によって、ユーザーが任意の名前の MFA デバイスを作成できますが、ポリシー内の他のステートメントでは、ユーザーはデバイスを現在サインインしているユーザーにしかアタッチできません。
- この AllowManageOwnUserMFA ステートメントでは、ユーザーは自分のユーザーの仮想、U2F、またはハードウェア MFA デバイスを表示または管理できます。このステートメントのリソース ARN は、ユーザー自身の IAM ユーザーにのみアクセスを許可します。ユーザーは他のユーザーの MFA デバイスを表示または管理することはできません。
- DenyAllExceptListedIfNoMFA ステートメントは、ユーザーが MFA でサインインしていない場合のみ、いくつかのリストされたアクションを除いて、すべての AWS のサービスのすべてのアクションへのアクセスを拒否します。このステートメントでは、"Deny" と "NotAction" の

組み合わせを使用して、表示されていないすべてのアクションへのアクセスを明示的に拒否しています。リストされている項目は、このステートメントによって拒否または許可されていません。ただし、アクションはポリシー内の他のステートメントによって許可されています。このステートメントのロジックの詳細については、「[Deny での NotAction の使用](#)」を参照してください。ユーザーが MFA でサインインしている場合、Condition テストは失敗し、このステートメントはアクションを拒否しません。この場合、ユーザーの他のポリシーまたは文によってユーザーのアクセス許可が決まります。

このステートメントは、ユーザーが MFA にサインインしていないときに、リストされているアクションのみを実行できることを保証します。さらに、他のステートメントまたはポリシーがそれらのアクションへのアクセスを許可している場合にのみ、リストされているアクションを実行できます。

...IfExists バージョンの Bool 演算子により、aws:MultiFactorAuthPresent キーが見つからない場合、条件は必ず true を返します。つまり、アクセスキーなどの長期認証情報を使用して API オペレーションにアクセスするユーザーは IAM 以外の API オペレーションへのアクセスを拒否されます。

このポリシーでは、IAM コンソールで [ユーザー] ページを表示したり、そのページを使用して自分のユーザー情報にアクセスすることはできません。これを許可するには、iam:ListUsers アクションを AllowViewAccountInfo ステートメントと DenyAllExceptListedIfNoMFA ステートメントに追加します。

#### Warning

MFA 認証なしで MFA デバイスを削除するためのアクセス許可を追加しないでください。このポリシーを持つユーザーは、仮想化 MFA デバイスを割り当てようとする場合に、iam:DeleteVirtualMFADevice を実行する権限がないというエラーが表示されることがあります。このような場合には、そのアクセス許可を DenyAllExceptListedIfNoMFA ステートメントに追加しないようにします。MFA を使用して認証されていないユーザーは、MFA デバイスを削除してはいけません。ユーザーが以前に自分のユーザーに仮想 MFA デバイスの割り当てを開始して処理をキャンセルした場合、このエラーが表示されることがあります。この問題を解決するには、ユーザーまたは他の管理者が AWS CLI または AWS API を使用してユーザーの既存の仮想化 MFA デバイスを削除する必要があります。詳細については、「[iam:DeleteVirtualMFADevice を実行する権限がありません](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": "iam:ListVirtualMFADevices",
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnVirtualMFADevice",
      "Effect": "Allow",
      "Action": [
        "iam:CreateVirtualMFADevice"
      ],
      "Resource": "arn:aws:iam::*:mfa/*"
    },
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:GetMFADevice",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "DenyAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice",
        "sts:GetSessionToken"
      ],
      "Resource": "*"
    }
  ]
}
```

```
        "Condition": {
            "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
        }
    ]
}
```

## AWS: IAM ユーザーが [セキュリティ認証情報] ページで自分のコンソールパスワードを変更できるようにします

この例は、IAM ユーザーが [セキュリティ認証情報] ページで自分の AWS Management Console パスワードを変更できるようにする ID ベースポリシーを作成する方法を示しています。この AWS Management Console ページにはアカウントとユーザー情報が表示されますが、ユーザーは自分のパスワードにアクセスすることしかできません。ユーザーに MFA を使用して自らのすべての認証情報を管理することを許可するには、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。ユーザーに MFA を使用せずに自らの認証情報を管理することを許可するには、「[AWS: IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

ユーザーが [セキュリティ認証情報] ページにアクセスする方法については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

### このポリシーで行うこと

- ViewAccountPasswordRequirements ステートメントでは、ユーザーは自分の IAM ユーザーパスワードを変更しながらアカウントパスワードの要件を確認できます。
- ChangeOwnPassword ステートメントを使用すると、ユーザーは自分のパスワードを変更できます。このステートメントには GetUser アクションも含まれています。これは、[My security credentials] (セキュリティ認証情報) ページのほとんどの情報を表示するために必要です。

このポリシーでは、IAM コンソールで [ユーザー] ページを表示したり、そのページを使用して自分のユーザー情報にアクセスすることはできません。これを許可するには、iam:ListUsers アクションを ViewAccountPasswordRequirements ステートメントに追加します。また、ユーザーが自分のユーザーページで自分のパスワードを変更することはできません。これを許可するには、iam:GetLoginProfile および iam:UpdateLoginProfile アクションを ChangeOwnPasswords ステートメントに追加します。

```
{
    "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Sid": "ViewAccountPasswordRequirements",  
    "Effect": "Allow",  
    "Action": "iam:GetAccountPasswordPolicy",  
    "Resource": "*"  
  },  
  {  
    "Sid": "ChangeOwnPassword",  
    "Effect": "Allow",  
    "Action": [  
      "iam:GetUser",  
      "iam:ChangePassword"  
    ],  
    "Resource": "arn:aws:iam::*:user/${aws:username}"  
  }  
]  
}
```

## AWS: IAM ユーザーが [セキュリティ認証情報] ページで自分のパスワード、アクセスキー、および SSH パブリックキーを管理できるようにします

この例は、IAM ユーザーが [セキュリティ認証情報] ページで自分のパスワード、アクセスキー、および X.509 証明書を管理できるようにする ID ベースポリシーを作成する方法を示しています。この AWS Management Console ページには、アカウント ID や正規ユーザー ID などのアカウント情報が表示されます。ユーザーは、自分のパスワード、アクセスキー、MFA デバイス、X.509 証明書、SSH キー、および Git 認証情報を表示および編集することもできます。このポリシー例には、パスワード、アクセスキー、および X.509 証明書のみを表示および編集するために必要なアクセス許可が含まれています。ユーザーに MFA を使用して自らのすべての認証情報を管理することを許可するには、「[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。ユーザーに MFA を使用せずに自らの認証情報を管理することを許可するには、「[AWS: IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)」を参照してください。

ユーザーが [セキュリティ認証情報] ページにアクセスする方法については、「[IAM ユーザー自身によるパスワードの変更方法 \(コンソール\)](#)」を参照してください。

### このポリシーで行うこと

- この AllowViewAccountInfo ステートメントでは、ユーザーにアカウントレベルの情報を表示します。これらのアクセス許可は、リソース ARN をサポートしていないか、または指定する必

要がないため、独自のステートメントに含まれている必要があります。代わりに "Resource" : "\*" を指定するアクセス許可を使用します。このステートメントには、ユーザーが特定の情報を表示できるようにする以下のアクションが含まれています。

- GetAccountPasswordPolicy – IAM ユーザーパスワードを変更しながら、アカウントのパスワード要件を確認します。
- GetAccountSummary – アカウント ID とアカウントの表示 [正規ユーザー ID](#)。
- AllowManageOwnPasswords ステートメントを使用すると、ユーザーは自分のパスワードを変更できます。このステートメントには GetUser アクションも含まれています。これは、[My security credentials] (セキュリティ認証情報) ページのほとんどの情報を表示するために必要です。
- この AllowManageOwnAccessKeys ステートメントにより、ユーザーは自分のアクセスキーを作成、更新、削除できます。ユーザーは指定されたアクセスキーの最後の使用時の情報を取得することもできます。
- この AllowManageOwnSSHPublicKeys ステートメントにより、ユーザーは自分の CodeCommit の SSH パブリックキーをアップロード、更新、削除できます。

このポリシーでは、ユーザーは自分の MFA デバイスを表示または管理できません。このポリシーでは、IAM コンソールで [Users] (ユーザー) ページを表示したり、そのページを使用して自分のユーザー情報にアクセスすることもできません。これを許可するには、iam:ListUsers アクションを AllowViewAccountInfo ステートメントに追加します。また、ユーザーが自分のユーザーページで自分のパスワードを変更することはできません。これを許可するには、iam:GetLoginProfile および iam:UpdateLoginProfile アクションを AllowManageOwnPasswords ステートメントに追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    },
    {
```



```

    "Sid": "AllowManageOwnPasswords",
    "Effect": "Allow",
    "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnAccessKeys",
    "Effect": "Allow",
    "Action": [
        "iam:CreateAccessKey",
        "iam>DeleteAccessKey",
        "iam>ListAccessKeys",
        "iam:UpdateAccessKey",
        "iam:GetAccessKeyLastUsed"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnSSHPublicKeys",
    "Effect": "Allow",
    "Action": [
        "iam>DeleteSSHPublicKey",
        "iam:GetSSHPublicKey",
        "iam>ListSSHPublicKeys",
        "iam:UpdateSSHPublicKey",
        "iam:UploadSSHPublicKey"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  }
]
}

```

## AWS: リクエストされたリージョンに基づいて、AWS へのアクセスを拒否する

この例は、NotAction を使用して指定したサービスのアクションを除く、[aws:RequestedRegion 条件キー](#)を使用して指定したリージョン外のアクションへのアクセスを拒否する ID ベースポリシーを作成する方法を示しています。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシー

の#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーは、ステートメントにリストされていないすべてのアクションへのアクセスを拒否する NotAction 効果がある Deny 要素を使用します。CloudFront、IAM、Route 53、および AWS Support サービスでのアクションは、物理的に AWS リージョンにある単一のエンドポイントを持つ一般的な us-east-1 グローバルサービスであるため、拒否しないでください。これらのサービスに対するすべてのリクエストは us-east-1 リージョンに対して行われるため、リクエストは NotAction 要素なしでは拒否されます。この要素を編集して、使用する他の AWS グローバルサービス (budgets、globalaccelerator、importexport、organizations、または waf など) のアクションを含めます。AWS Chatbot や AWS Device Farm など、その他のグローバルサービスは、エンドポイントが us-west-2 リージョンに物理的に配置されているグローバルサービスです。単一のグローバルエンドポイントを持つすべてのサービスについては、「AWS 全般のリファレンス」の「[AWS リージョンとエンドポイント](#)」を参照してください。NotAction 効果を持つ Deny 要素の使用の詳細については、「[IAM JSON ポリシー要素NotAction](#)」を参照してください。

#### Important

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllOutsideRequestedRegions",
      "Effect": "Deny",
      "NotAction": [
        "cloudfront:*",
        "iam:*",
        "route53:*",
        "support:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:RequestedRegion": [
            "eu-central-1",
            "eu-west-1",

```

```
    "eu-west-2",  
    "eu-west-3"  
  ]  
}  
}  
]  
}
```

## AWS: 送信元 IP に基づいて AWS へのアクセスを拒否する

この例は、指定した IP 範囲外のプリンシパルからリクエストが送信された場合に、アカウントのすべての AWS のアクションへのアクセスを拒否する ID ベースポリシーを作成する方法を示しています。ポリシーは、会社の IP アドレスが指定された範囲内にある場合に有効です。この例では、リクエストは、CIDR 範囲 192.0.2.0/24 または 203.0.113.0/24 以外からでない限り拒否されます。ポリシーでは、オリジナルリクエストの IP アドレスが保持されるため、[転送アクセスセッション](#)を使用した AWS サービスからのリクエストは拒否されません。

"Effect": "Deny" と同じポリシーステートメントで負の条件を使用する際は注意してください。負の条件を使用すると、ポリシーステートメントで指定されたアクションは、指定されたものを除くすべての条件で明示的に拒否されます。

### Important

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

他のポリシーでアクションが許可されている場合、プリンシパルは IP アドレス範囲内からリクエストを実行することができます。AWS サービスは、プリンシパルの認証情報を使用してリクエストを実行することもできます。プリンシパルが IP 範囲外からリクエストを実行すると、リクエストは拒否されます。

ポリシーで `aws:SourceIp` キーが機能しない場合の情報など、`aws:SourceIp` 条件キーの使用の詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

```
{  
  "Version": "2012-10-17",  
  "Statement": {
```

```
"Effect": "Deny",
"Action": "*",
"Resource": "*",
"Condition": {
  "NotIpAddress": {
    "aws:SourceIp": [
      "192.0.2.0/24",
      "203.0.113.0/24"
    ]
  }
}
}
```

## AWS: AWS Data Exchange を除くアカウント外の Amazon S3 リソースへのアクセスを拒否する

この例では、AWS Data Exchange が通常のオペレーションに必要とするリソースを除いた、アカウントに属さない AWS のすべてのリソースへのアクセスを拒否する ID ベースのポリシーを作成する方法を示します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

条件キー `aws:ResourceOrgPaths` および `aws:ResourceOrgID` を使用して AWS Data Exchange 所有のリソースを考慮しながら、組織または組織単位内のリソースへのアクセスを制限するために、類似ポリシーを作成できます。

ご自分の環境で AWS Data Exchange を使用する場合、サービスはサービスアカウントが所有する Amazon S3 バケットなどのリソースを作成し、操作します。たとえば、AWS Data Exchange は、IAM プリンシパル (ユーザーまたはロール) に代わって AWS Data Exchange API を呼び出す AWS Data Exchange サービスが所有する Amazon S3 バケットに要求を送信します。その場合は、`aws:ResourceAccount`、`aws:ResourceOrgPaths`、`aws:ResourceOrgID` ポリシーを使用して、AWS Data Exchange が所有するリソースを考慮せず、サービスアカウントが所有するバケットへのアクセスを拒否します。

- ステートメント `DenyAllAwsResourcesOutsideAccountExceptS3` では、ステートメントにリスト表示されておらず、またリスト表示されたアカウントに属していないすべてのアクションへのアクセスを明確に[拒否](#)する効果を持つ `NotAction` 要素を使用します。NotAction 要素は、このステートメントの例外を示します。これらのアクションは、ステートメントの例外を示します。これは AWS Data Exchange によって作成されたリソースでアクションが実行されると、ポリシーによってアクションが拒否されるためです。

- このステートメント `DenyAllS3ResourcesOutsideAccountExceptDataExchange` は、`ResourceAccount` および `CalledVia` の条件の組み合わせを使用して、前のステートメントで除外された 3 つの Amazon S3 アクションへのアクセスを拒否します。このステートメントは、そのリソースがリスト表示されたアカウントに属していない場合、そして呼び出しサービスが AWS Data Exchange でない場合にアクションを拒否します。このステートメントは、リソースがリスト表示されたアカウントに属しているか、リスト表示されているサービスプリンシパル `dataexchange.amazonaws.com` が操作を行う場合には、アクションを拒否しません。

#### Important

このポリシーでは、アクションを許可しません。それは、リスト表示されたアカウントに属さないステートメントでリスト表示された、すべてのリソースへのアクセスを明確に拒否する Deny の効果を使用します。特定のリソースへのアクセスを許可する他のポリシーと組み合わせてこのポリシーを使用します。

次の例は、必要な Amazon S3 バケットへのアクセスを許可するためにポリシーを設定する方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllAwsResourcesOutsideAccountExceptAmazonS3",
      "Effect": "Deny",
      "NotAction": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ],
  {
```

```
"Sid": "DenyAllS3ResourcesOutsideAccountExceptDataExchange",
"Effect": "Deny",
"Action": [
  "s3:GetObject",
  "s3:PutObject",
  "s3:PutObjectAcl"
],
"Resource": "*",
"Condition": {
  "StringNotEquals": {
    "aws:ResourceAccount": [
      "111122223333"
    ]
  },
  "ForAllValues:StringNotEquals": {
    "aws:CalledVia": [
      "dataexchange.amazonaws.com"
    ]
  }
}
}
```

## AWS Data Pipeline: ユーザーが作成していない DataPipeline パイプラインへのアクセスを拒否する

この例では、ユーザーが作成していないパイプラインへのアクセスを拒否する ID ベースポリシーを作成する方法を示します。PipelineCreator フィールドの値が IAM ユーザー名と一致する場合、指定されたアクションは拒否されません。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

### Important

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

    {
      "Sid": "ExplicitDenyIfNotTheOwner",
      "Effect": "Deny",
      "Action": [
        "datapipeline:ActivatePipeline",
        "datapipeline:AddTags",
        "datapipeline:DeactivatePipeline",
        "datapipeline>DeletePipeline",
        "datapipeline:DescribeObjects",
        "datapipeline:EvaluateExpression",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:PollForTask",
        "datapipeline:PutPipelineDefinition",
        "datapipeline:QueryObjects",
        "datapipeline:RemoveTags",
        "datapipeline:ReportTaskProgress",
        "datapipeline:ReportTaskRunnerHeartbeat",
        "datapipeline:SetStatus",
        "datapipeline:SetTaskStatus",
        "datapipeline:ValidatePipelineDefinition"
      ],
      "Resource": ["*"],
      "Condition": {
        "StringNotEquals": {"datapipeline:PipelineCreator": "${aws:userid}"}
      }
    }
  ]
}

```

## Amazon DynamoDB: 特定のテーブルへのアクセスの許可

この例は、MyTable DynamoDB テーブルへのフルアクセスを許可する ID ベースポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

### Important

このポリシーは、DynamoDB テーブルで実行できるすべてのアクションを許可します。これらのアクションを確認するには、Amazon DynamoDB デベロッパーガイドの「[DynamoDB API の許可: アクション、リソース、条件リファレンス](#)」を参照してください。個別のアク

ションを一覧表示して、同じアクセス許可を提供することができます。ただし、Action 要素でワイルドカード (\*) を使用している場合 ("dynamodb:List\*" など)、DynamoDB で新しい List アクションが追加されても、ポリシーを更新する必要はありません。

このポリシーは、指定された名前で存在する DynamoDB テーブルでのみアクションを許可します。ユーザーに [DynamoDB 全体への Read アクセスを許可するため](#) に、[AmazonDynamoDBReadOnlyAccess](#) AWS 管理ポリシーをアタッチすることもできます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAndDescribe",
      "Effect": "Allow",
      "Action": [
        "dynamodb:List*",
        "dynamodb:DescribeReservedCapacity*",
        "dynamodb:DescribeLimits",
        "dynamodb:DescribeTimeToLive"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SpecificTable",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchGet*",
        "dynamodb:DescribeStream",
        "dynamodb:DescribeTable",
        "dynamodb:Get*",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchWrite*",
        "dynamodb:CreateTable",
        "dynamodb>Delete*",
        "dynamodb:Update*",
        "dynamodb:PutItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
    }
  ]
}
```



```
}
```

## Amazon DynamoDB: 特定の属性へのアクセスの許可

この例では、特定の DynamoDB 属性へのアクセスを許可する ID ベースポリシーを作成する方法を示します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

dynamodb:Select 要件により、インデックスの射影からなど、API アクションが許可されていない属性を返さないようにします。DynamoDB 条件キーの詳細については、『Amazon DynamoDB 開発者ガイド』の「[条件の指定: 条件キーの使用](#)」を参照してください。IAM ポリシーの Condition ブロック内の複数条件および複数条件キーの使用については、[条件内の複数の値](#)を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": ["arn:aws:dynamodb:*:*:table/table-name"],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:Attributes": [
            "column-name-1",
            "column-name-2",
            "column-name-3"
          ]
        },
        "StringEqualsIfExists": {"dynamodb:Select": "SPECIFIC_ATTRIBUTES"}
      }
    }
  ]
}
```

```
]
}
```

## Amazon DynamoDB: Amazon Cognito ID に基づいて DynamoDB への項目レベルのアクセスを許可する

この例は、Amazon Cognito アイデンティティプールのユーザー ID に基づいて MyTable DynamoDB テーブルへのアイテムレベルのアクセスを許可する ID ベースポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

このポリシーを使用するには、Amazon Cognito アイデンティティプールのユーザー ID がパーティションキーとなるように DynamoDB テーブルを作成する必要があります。詳細については、Amazon DynamoDB 開発者ガイドの [テーブルの作成](#)を参照してください。

条件キーの詳細については、『[Amazon DynamoDB 開発者ガイド](#)』の「条件の指定: 条件キーの使用」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
      ],
      "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

## Amazon EC2: タグに基づいて EC2 インスタンスに Amazon EBS ボリュームをアタッチまたはデタッチする

この例は、EBS ボリュームの所有者によって、開発インスタンス (Department=Development) としてタグ付けされた EC2 インスタンスのタグ VolumeUser を使用して定義された EBS ボリュームをアタッチまたはデタッチすることを許可する ID ベースポリシーの作成方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

Amazon EC2 リソースへのアクセスを制御する IAM ポリシーの作成の詳細については、「[Amazon EC2 ユーザーガイド](#)」の [Amazon EC2 リソースへのアクセスの制御](#)に関するページを参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Department": "Development"}
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Resource": "arn:aws:ec2:*:*:volume/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/VolumeUser": "${aws:username}"}
      }
    }
  ]
}
```

```
}
```

## Amazon EC2: 特定のサブネットで EC2 インスタンスを起動することをプログラムによりコンソールで許可する

この例では、特定のサブネットですべての EC2 オブジェクトと EC2 インスタンスの起動を許可する ID ベースのポリシーを作成する方法を示します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "ec2:GetConsole*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:subnet/subnet-subnet-id",
        "arn:aws:ec2:*:*:network-interface/*",
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:volume/*",
        "arn:aws:ec2:*:*:image/ami-*",
        "arn:aws:ec2:*:*:key-pair/*",
        "arn:aws:ec2:*:*:security-group*"
      ]
    }
  ]
}
```

## Amazon EC2: 特定のキーと値のペアでタグ付けされた EC2 セキュリティグループの管理をプログラムによりコンソールで許可する

この例では、同じタグを持つセキュリティグループに対して、特定のアクションを実行するためのアクセス許可をユーザーに与える、ID ベースのポリシーを作成する方法を示します。このポリシーは、Amazon EC2 コンソールでセキュリティグループを表示し、インバウンドおよびアウトバウンドのルールを追加および削除し、タグ Department=Test を含む既存のセキュリティグループのルール説明を変更するアクセス許可をユーザーに付与します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSecurityGroupRules",
      "ec2:DescribeTags"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:RevokeSecurityGroupIngress",
      "ec2:AuthorizeSecurityGroupEgress",
      "ec2:RevokeSecurityGroupEgress",
      "ec2:ModifySecurityGroupRules",
      "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
      "ec2:UpdateSecurityGroupRuleDescriptionsEgress"
    ],
    "Resource": [
      "arn:aws:ec2:region:111122223333:security-group/*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Department": "Test"
      }
    }
  }
}
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:ModifySecurityGroupRules"
      ],
      "Resource": [
        "arn:aws:ec2:region:111122223333:security-group-rule/*"
      ]
    }
  ]
}
```

## Amazon EC2: プログラムおよびコンソールでユーザーがタグ付けされた EC2 インスタンスの起動や停止を許可する

この例は、IAM ユーザーが EC2 インスタンスを開始または停止できるようにする ID ベースのポリシーを作成する方法を示していますが、インスタスタグ Owner にそのユーザーのユーザー名の値がある場合に限り、このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeInstances",
      "Resource": "*"
    }
  ]
}
```

```
]
}
```

## EC2: タグに基づくインスタンスの開始または停止

この例は、タグのキーバリューのペア `Department = Data` を持つプリンシパルによってのみ、タグのキーバリューペア `Project = DataAnalytics` を持つインスタンスの開始または停止を許可する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

ポリシーの条件によって、条件のいずれも `true` の場合には `true` が返ります。インスタンスには `Project=DataAnalytics` タグが必要です。さらに、リクエストを行う IAM プリンシパル (ユーザーまたはロール) には、`Department=Data` タグが必要です。

### Note

ベストプラクティスとして、ユーザーの特定のタグの有無に対応できるように、`aws:PrincipalTag` 条件キーを持つポリシーを IAM グループにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StartStopIfTags",
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:region:account-id:instance/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "DataAnalytics",
          "aws:PrincipalTag/Department": "Data"
        }
      }
    }
  ]
}
```

```
]
}
```

## EC2: 一致するプリンシパルタグとリソースタグに基づいてインスタンスを開始または停止する

この例では、インスタンスのリソースタグとプリンシパルのタグのタグキー `CostCenter` の値が同じである場合にプリンシパルが Amazon EC2 インスタンスの起動または停止を許可する ID ベースのポリシーを作成する方法を示します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

### Note

ベストプラクティスとして、ユーザーの特定のタグの有無に対応できるように、`aws:PrincipalTag` 条件キーを持つポリシーを IAM グループにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ec2:startInstances",
      "ec2:stopInstances"
    ],
    "Resource": "*",
    "Condition": {"StringEquals":
      {"aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter"}
    }}
  }
}
```

## Amazon EC2: 特定のリージョンでの完全な EC2 アクセスをプログラムによりコンソールで許可する

この例では、特定のリージョン内で完全な EC2 アクセスを許可する ID ベースのポリシーを作成する方法を示します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に



置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。リージョンコードの一覧については、[Amazon EC2 ユーザーガイド](#)の「使用可能なリージョン」を参照してください。

または、グローバル条件キー [aws:RequestedRegion](#), を使用することもできます。これは、すべての Amazon EC2 API アクションによってサポートされています。詳細については、[Amazon EC2 ユーザーガイド](#)の「例: 特定のリージョンへのアクセスの制限」をご参照ください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ec2:*",
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "ec2:Region": "us-east-2"
        }
      }
    }
  ]
}
```

## Amazon EC2: EC2 インスタンスの起動または停止、およびセキュリティグループの変更を、プログラムによりコンソールで許可する

この例では、特定の EC2 インスタンスの起動または停止、特定のセキュリティグループの変更を許可する ID ベースのポリシーを作成する方法を示します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの **#####** を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "*",
    "Effect": "Allow"
  },
  {
    "Action": [
      "ec2:AuthorizeSecurityGroupEgress",
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:RevokeSecurityGroupEgress",
      "ec2:RevokeSecurityGroupIngress",
      "ec2:StartInstances",
      "ec2:StopInstances"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:instance/i-instance-id",
      "arn:aws:ec2:*:*:security-group/sg-security-group-id"
    ],
    "Effect": "Allow"
  }
]
```

Amazon EC2: 特定の EC2 オペレーション (GetSessionToken) に対して MFA を必要とします。

この例は、Amazon EC2 のすべての AWS API オペレーションへのフルアクセスを許可する ID ベースのポリシーを作成する方法を示しています。ただし、ユーザーが[多要素認証 \(MFA\)](#) を使用して認証されていない場合は、StopInstances および TerminateInstances API オペレーションへのアクセスを明示的に拒否します。これをプログラムで行うには、GetSessionToken オペレーションを呼び出すときにユーザーはオプションの SerialNumber および TokenCode 値を含める必要があります。このオペレーションでは、MFA を使用して認証された一時認証情報を返します。GetSessionToken の詳細については、「[GetSessionToken — 信頼されていない環境にあるユーザー向けの一時的認証情報](#)」を参照してください。

このポリシーで行うこと

- この AllowAllActionsForEC2 ステートメントではすべての Amazon EC2 アクションが許可されます。
- DenyStopAndTerminateWhenMFAIsNotPresent ステートメントは、MFA コンテキストが欠落している場合に、StopInstances および TerminateInstances のアクションを拒否しま

す。これは、多要素認証コンテキストがない場合 (MFA が使用されなかったことを意味します) にアクションが拒否されることを意味します。拒否が許可に優先します。

### Note

Deny ステートメントの `MultiFactorAuthPresent` の条件チェックは、MFA が使用されていないければ該当キーが存在せず、評価できないため、`{"Bool": {"aws:MultiFactorAuthPresent": false}}` となりません。その代わりに、`BoolIfExists` チェックを使用して、値をチェックする前にキーが存在するかどうか確認します。詳細については、「[IfExists 条件演算子](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllActionsForEC2",
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
      "Effect": "Deny",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*",
      "Condition": {
        "BoolIfExists": {"aws:MultiFactorAuthPresent": false}
      }
    }
  ]
}
```

## Amazon EC2: 終了する EC2 インスタンスを IP アドレス範囲に制限する

この例は、アクションを許可することで EC2 インスタンスを制限する ID ベースのポリシーを作成する方法を示していますが、リクエストが指定された IP 範囲外から送信された場合は明示的にアク

セスを拒否します。ポリシーは、会社の IP アドレスが指定された範囲内にある場合に有効です。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーを、ec2:TerminateInstances アクションを許可する他のポリシー ([AmazonEC2FullAccess](#) AWS 管理ポリシーなど) と組み合わせて使用すると、アクセスは拒否されます。これは、明示的な拒否のステートメントは許可のステートメントより優先されるからです。詳細については、「[the section called “アカウント内でのリクエストの許可または拒否の決定”](#)」を参照してください。

#### Important

aws:SourceIp 条件キーは、お客様に代わって呼び出しを行う AWS CloudFormation などの AWS サービスへのアクセスを拒否します。aws:SourceIp 条件キーの使用に関する詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource": ["*"]
    },
    {
      "Effect": "Deny",
      "Action": ["ec2:TerminateInstances"],
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24",
            "203.0.113.0/24"
          ]
        }
      },
      "Resource": ["*"]
    }
  ]
}
```

```
}
```

## IAM: Policy Simulator API へのアクセス

この例では、ユーザー、グループ、現在の AWS アカウント のロールにアタッチされたポリシーに、ポリシーシミュレータ API の使用を許可する ID ベースのポリシーを作成する方法を示します。このポリシーは、文字列として API に渡される機密性の低いポリシーをシミュレートするアクセスも許可します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetContextKeysForCustomPolicy",
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:SimulateCustomPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

### Note

ユーザーに、現在の AWS アカウント のユーザー、グループ、ロールにアタッチされたポリシーをシミュレートする Policy Simulator コンソールへのアクセスを許可するには、「[IAM: Policy Simulator のコンソールへのアクセス](#)」を参照してください。

## IAM: Policy Simulator のコンソールへのアクセス

この例では、ユーザー、グループ、現在の AWS アカウント のロールにアタッチされたポリシーに、ポリシーシミュレータのコンソールの使用を許可する ID ベースのポリシーを作成する方法を示します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

IAM Policy Simulator コンソールは <https://policysim.aws.amazon.com/> でアクセスできます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetGroup",
        "iam:GetGroupPolicy",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:GetUser",
        "iam:GetUserPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListAttachedRolePolicies",
        "iam:ListAttachedUserPolicies",
        "iam:ListGroups",
        "iam:ListGroupPolicies",
        "iam:ListGroupsForUser",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:ListUserPolicies",
        "iam:ListUsers"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

## IAM: 特定のタグを持つロールを引き受ける

この例では、IAM ユーザーがタグのキーバリューのペア `Project = ExampleCorpABC` を持つロールを引き受けることを許可する ID ベースのポリシーの作成方法を示します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

このタグを持つロールがユーザーと同じアカウントに存在する場合、ユーザーはそのロールを引き受けることができます。このタグを持つロールがユーザー以外のアカウントに存在する場合は、追加の

アクセス許可が必要です。クロスアカウントロールの信頼ポリシーでは、ユーザーまたはユーザーのアカウントのすべてのメンバーがそのロールを引き受けることも許可する必要があります。クロスアカウントアクセスのロールの使用に関する詳細情報は、「[所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeTaggedRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {"iam:ResourceTag/Project": "ExampleCorpABC"}
      }
    }
  ]
}
```

## IAM: 複数のサービスへのアクセスをプログラムによりコンソールで許可および拒否する

この例では、IAM で複数のサービスへのフルアクセスと IAM の制限された自己管理アクセスを許可する ID ベースのポリシーを作成する方法を示します。また、Amazon S3 の logs バケットまたは Amazon EC2 の i-1234567890abcdef0 インスタンスへのアクセスを拒否します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

### Warning

このポリシーは、複数のサービスのすべてのアクションとリソースへのフルアクセスを許可します。このポリシーは、信頼されている管理者にのみ適用する必要があります。

このポリシーをアクセス許可の境界として使用して、アイデンティティベースのポリシーが IAM ユーザーに付与できる最大のアクセス許可の上限を定義できます。詳細については、「[アクセス許可の境界を使用した他のユーザーへの責任の委任](#)」を参照してください。ポリシーがユーザーのアクセス許可の境界として使用される場合、ステートメントは次の境界を定義します。

- この AllowServices ステートメントでは、指定された AWS のサービスへのフルアクセスを許可します。つまり、これらのサービスにおけるユーザーのアクションは、ユーザーにアタッチされているアクセス許可ポリシーによってのみ制限されます。
- AllowIAMConsoleForCredentials ステートメントは、すべての IAM ユーザーを一覧表示するためのアクセス権を付与します。このアクセス権は、AWS Management Console で [ユーザー] ページに移動するために必要です。また、このアクセス権では、アカウントのパスワード要件を表示することができます。これは、ユーザーが自分のパスワードを変更する場合に必要です。
- AllowManageOwnPasswordAndAccessKeys ステートメントは、自分のパスワードおよびプログラムを使用したアクセスキーの管理のみ許可します。これが重要であるのは、別のポリシーが IAM へのフルアクセスをユーザーに与えた場合、このユーザーは自分や他のユーザーのアクセス許可を変更できるようになるためです。このステートメントで、これを防止します。
- DenyS3Logs ステートメントでは、logs バケットへのアクセスを明示的に拒否します。このポリシーはユーザーに会社の制限を適用します。
- DenyEC2Production ステートメントでは、i-1234567890abcdef0 インスタンスへのアクセスを明示的に拒否します。

このポリシーでは、他のサービスまたはアクションへのアクセスは許可されません。ポリシーがユーザーのアクセス許可の境界として使用されている場合、そのユーザーに関連付けられている他のポリシーがそれらのアクションを許可していても、AWS によってリクエストが拒否されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowServices",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowIAMConsoleForCredentials",
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetAccountPasswordPolicy"
      ]
    }
  ]
}
```



```

    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowManageOwnPasswordAndAccessKeys",
    "Effect": "Allow",
    "Action": [
      "iam:*AccessKey*",
      "iam:ChangePassword",
      "iam:GetUser",
      "iam:*LoginProfile*"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
  },
  {
    "Sid": "DenyS3Logs",
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::logs",
      "arn:aws:s3:::logs/*"
    ]
  },
  {
    "Sid": "DenyEC2Production",
    "Effect": "Deny",
    "Action": "ec2:*",
    "Resource": "arn:aws:ec2::*:instance/i-1234567890abcdef0"
  }
]
}

```

## IAM: 特定のタグを持つユーザーに特定のタグを追加する

この例は、タグ値 Department、Marketing、または Development を使用して、IAM ユーザーにタグキー QualityAssurance を追加できる ID ベースのポリシーを作成する方法を示しています。そのユーザーには、タグのキーバリューのペア JobFunction = manager がすでに含まれている必要があります。このポリシーを使用して、管理者が 3 つのいずれかの部門にのみ属していることを必須とすることができます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

ListTagsForAllUsers ステートメントでは、アカウントのすべてのユーザーのタグを表示することができます。

TagManagerWithSpecificDepartment ステートメントの最初の条件では、StringEquals 条件演算子を使用します。この条件によって、条件のいずれも true の場合には true が返ります。タグ付けされているユーザーには、すでに JobFunction=Manager タグが含まれます。リクエストには、一覧表示されているタグ値のいずれかを持つタグキー Department が含まれている必要があります。

2 番目の条件では、ForAllValues:StringEquals 条件演算子を使用します。この条件では、リクエストのすべてのタグキーが、ポリシーのキーと一致する場合に true が返ります。つまり、リクエストのタグキーのみ Department である必要があります。ForAllValues の使用の詳細については、「[複数値のコンテキストキー](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListTagsForAllUsers",
      "Effect": "Allow",
      "Action": [
        "iam:ListUserTags",
        "iam:ListUsers"
      ],
      "Resource": "*"
    },
    {
      "Sid": "TagManagerWithSpecificDepartment",
      "Effect": "Allow",
      "Action": "iam:TagUser",
      "Resource": "*",
      "Condition": {"StringEquals": {
        "iam:ResourceTag/JobFunction": "Manager",
        "aws:RequestTag/Department": [
          "Marketing",
          "Development",
          "QualityAssurance"
        ]
      }},
      "ForAllValues:StringEquals": {"aws:TagKeys": "Department"}
    }
  ]
}
```

```
]
}
```

## IAM: 特定の値を持つ特定のタグを追加する

この例は、タグキー `CostCenter` とタグ値 `A-123` またはタグ値 `B-456` のいずれかのみを IAM ユーザーまたはロールに追加できる ID ベースのポリシーを作成する方法を示しています。このポリシーを使用して、特定のタグキーとタグ値のセットにタグ付けを制限することができます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

`ConsoleDisplay` ステートメントでは、アカウントのすべてのユーザーとロールのタグを表示することができます。

`AddTag` ステートメントの最初の条件では、`StringEquals` 条件演算子を使用します。この条件によって、一覧表示されているいずれかのタグ値を持つ `CostCenter` タグキーが含まれている場合に `true` が返ります。

2 番目の条件では、`ForAllValues:StringEquals` 条件演算子を使用します。この条件では、リクエストのすべてのタグキーが、ポリシーのキーと一致する場合に `true` が返ります。つまり、リクエストのタグキーのみ `CostCenter` である必要があります。`ForAllValues` の使用の詳細については、「[複数値のコンテキストキー](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConsoleDisplay",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:GetUser",
        "iam:ListRoles",
        "iam:ListRoleTags",
        "iam:ListUsers",
        "iam:ListUserTags"
      ],
      "Resource": "*"
    },
    {
```

```
    "Sid": "AddTag",
    "Effect": "Allow",
    "Action": [
        "iam:TagUser",
        "iam:TagRole"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/CostCenter": [
                "A-123",
                "B-456"
            ]
        },
        "ForAllValues:StringEquals": {"aws:TagKeys": "CostCenter"}
    }
}
]
```

## IAM: 特定のタグのみで新しいユーザーを作成する

この例は、IAM ユーザーの作成を許可する ID ベースのポリシーを作成する方法を示していますが、タグキー Department と JobFunction の一方のみまたは両方を使用します。Department タグキーに、Development または QualityAssurance タグ値が含まれている必要があります。JobFunction タグキーに、Employee タグ値が含まれている必要があります。このポリシーを使用して、新しいユーザーに特定のジョブ機能および部門が含まれるよう義務付けることができます。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

ステートメントの最初の条件では、StringEqualsIfExists 条件演算子を使用します。キーが Department または JobFunction のタグがリクエストに存在する場合は、指定された値がタグに含まれている必要があります。どちらのキーも存在しない場合、この条件は true と評価されます。指定された条件キーのいずれかがリクエストに存在していても、許可されている値とは異なる値が含まれている場合にのみ、条件は、false と評価されます。IfExists の使用の詳細については、「[IfExists 条件演算子](#)」を参照してください。

2 番目の条件では、ForAllValues:StringEquals 条件演算子を使用します。この条件では、リクエストの指定されたすべてのタグキーと、ポリシーの 1 つ以上の値が一致する場合に

true が返ります。つまり、リクエストのすべてのタグがこのリストに含まれていることになります。ただし、リクエストには、リストのいずれかのタグのみ含めることができます。たとえば、Department=QualityAssurance タグのみを使用して、IAM ユーザーを作成できます。ただし、JobFunction=employee タグと Project=core タグを使用して、IAM ユーザーを作成することはできません。ForAllValues の使用の詳細については、「[複数値のコンテキストキー](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TagUsersWithOnlyTheseTags",
      "Effect": "Allow",
      "Action": [
        "iam:CreateUser",
        "iam:TagUser"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "aws:RequestTag/Department": [
            "Development",
            "QualityAssurance"
          ],
          "aws:RequestTag/JobFunction": "Employee"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "Department",
            "JobFunction"
          ]
        }
      }
    }
  ]
}
```

## IAM: 認証情報レポートを生成して取得する

この例は、ユーザーに、AWS アカウント のすべての IAM ユーザーを一覧表示するレポートを生成してダウンロードすることを許可する ID ベースのポリシーを作成する方法を示しています。このレポートには、パスワード、アクセスキー、MFA デバイス、署名証明書など、ユーザー認証情報のス

タスクが含まれています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

認証情報レポートの詳細については、「[AWS アカウントの認証情報レポートの取得](#)」を参照してください

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GenerateCredentialReport",
      "iam:GetCredentialReport"
    ],
    "Resource": "*"
  }
}
```

## IAM:: グループのメンバーシップをプログラムによりコンソールで管理することを許可する

この例では、MarketingTeam という名前のグループのメンバーシップを更新することを許可する ID ベースのポリシーの作成方法を示します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

### このポリシーで行うこと

- この ViewGroups ステートメントにより、ユーザーは AWS Management Console 内のすべてのユーザーとグループを一覧表示できます。また、アカウント内のユーザーに関する基本情報を表示することもできます。これらのアクセス許可は、リソース ARN をサポートしていないが、または指定する必要がないため、独自のステートメントに含まれている必要があります。代わりに "Resource" : "\*" を指定するアクセス許可を使用します。
- ViewEditThisGroup ステートメントを使用すると、ユーザーは MarketingTeam グループに関する情報を表示したり、そのグループにユーザーを追加したり削除したりできます。

このポリシーでは、ユーザーはユーザーまたは MarketingTeam グループのアクセス許可を表示または編集できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewGroups",
      "Effect": "Allow",
      "Action": [
        "iam:ListGroups",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:ListGroupsForUser"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ViewEditThisGroup",
      "Effect": "Allow",
      "Action": [
        "iam:AddUserToGroup",
        "iam:RemoveUserFromGroup",
        "iam:GetGroup"
      ],
      "Resource": "arn:aws:iam::*:group/MarketingTeam"
    }
  ]
}
```

## IAM: 特定のタグを管理する

この例は、Department IAM エンティティ (ユーザーおよびロール) からタグキーを使用して IAM タグを追加または削除することを許可する ID ベースのポリシーの、作成方法を示しています。このポリシーでは、Department タグの値は制限されません。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:TagUser",
```

```
        "iam:TagRole",
        "iam:UntagUser",
        "iam:UntagRole"
    ],
    "Resource": "*",
    "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": "Department"}}
}
}
```

## IAM: IAM ロールを特定の AWS のサービスに渡す

この例は、IAM サービスロールを Amazon CloudWatch サービスに渡すことを許可する ID ベースのポリシーの作成方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

サービスロールは、IAM ロールを引き受けることができるプリンシパルとして AWS サービスを指定するロールです。これにより、サービスはユーザーに代わってロールを引き受け、他のサービスのリソースにアクセスできます。Amazon CloudWatch がユーザーが渡すロールを引き受けるようにするには、ユーザーのロールの信頼ポリシーのプリンシパルとして `cloudwatch.amazonaws.com` サービスプリンシパルを指定する必要があります。サービスプリンシパルはサービスによって定義されます。サービスのサービスプリンシパルについては、そのサービスのドキュメントを参照してください。一部のサービスについては、[IAM と連携する AWS のサービス](#) を参照し、[サービスにリンクされたロール] 列で [はい] になっているサービスを探してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。amazonaws.com を検索してサービスプリンシパルを表示します。

サービスにサービスロールを渡す方法の詳細については、「[AWS のサービスにロールを渡すアクセス権限をユーザーに付与する](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {"iam:PassedToService": "cloudwatch.amazonaws.com"}
      }
    }
  ]
}
```



```
    }  
  }  
]  
}
```

## IAM: レポートなしでの IAM コンソールへの読み取り専用アクセスを許可します

この例は、IAM ユーザーが文字列 `Get` または `List` で始まる IAM アクションを実行できるようにする ID ベースのポリシーを作成する方法を示しています。ユーザーがコンソールを操作すると、コンソールは IAM にグループ、ユーザー、ロール、およびポリシーを一覧表示し、それらのリソースに関するレポートを生成するようにリクエストします。

アスタリスクは、ワイルドカードとして機能します。ポリシーで `iam:Get*` を使用すると、結果のアクセス許可には、`Get` や `GetUser` など、`GetRole` で始まるすべての IAM アクションが含まれます。ワイルドカードは、今後、新しいタイプのエンティティが IAM に追加される場合に役立ちます。この場合、ポリシーによって付与されたアクセス許可によって、ユーザーは自動的にそれらの新しいエンティティに関する詳細を一覧表示して取得できます。

このポリシーは、レポートまたはサービスの最終アクセス詳細を生成するために使用できません。これを許可する別のポリシーについては、[IAM: IAM コンソールへの読み取り専用アクセスを許可する](#)を参照してください。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "iam:Get*",  
        "iam>List*"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

## IAM: IAM コンソールへの読み取り専用アクセスを許可する

この例は、IAM ユーザーが文字列 `Get`、`List`、または `Generate` で始まる IAM アクションを実行できるようにする ID ベースのポリシーを作成する方法を示しています。ユーザーが IAM コンソールを操作すると、コンソールはグループ、ユーザー、ロール、およびポリシーを一覧表示し、それらのリソースに関するレポートを生成するようにリクエストします。

アスタリスクは、ワイルドカードとして機能します。ポリシーで `iam:Get*` を使用すると、結果のアクセス許可には、`Get` や `GetUser` など、`GetRole` で始まるすべてのアクションが含まれます。特に新しい種類のエンティティが今後 IAM に追加される場合は、ワイルドカードを使用すると便利です。この場合、ポリシーによって付与されたアクセス許可によって、ユーザーは自動的にそれらの新しいエンティティに関する詳細を一覧表示して取得できます。

このポリシーは、レポートまたはサービスの最終アクセス詳細を生成する権限を含むコンソールアクセスに使用します。アクションの生成を許可しない別のポリシーについては、[IAM: レポートなしでの IAM コンソールへの読み取り専用アクセスを許可します](#)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:Get*",
      "iam:List*",
      "iam:Generate*"
    ],
    "Resource": "*"
  }
}
```

## IAM:: 特定の IAM ユーザーによるグループの管理をプログラムによりコンソールで許可する

この例は、特定の IAM ユーザーが `AllUsers` グループを管理するための、特定の ID ベースのポリシーを作成する方法を示しています。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの `#####` `#` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

### このポリシーで行うこと

- `AllowAllUsersToListAllGroups` ステートメントでは、すべてのグループを一覧表示します。これはコンソールのアクセスに必要です。このアクセス許可は、リソース ARN をサポートしていないため、独自のステートメント内にある必要があります。代わりに `"Resource" : "*"`  を指定するアクセス許可を使用します。
- `AllowAllUsersToViewAndManageThisGroup` ステートメントは、グループリソースタイプに対して実行できるすべてのグループアクションを許可します。グループリソースタイプでは

なくユーザリソースタイプで実行できる ListGroupsForUser アクションは許可されていません。IAM アクションに指定できるリソースタイプの詳細については、「[AWS Identity and Access Management のアクション、リソース、条件キー](#)」を参照してください。

- LimitGroupManagementAccessToSpecificUsers ステートメントは、指定した名前を持つユーザーが書き込みおよびアクセス許可管理グループのアクションにアクセスすることを拒否します。ポリシーで指定されたユーザーがグループに変更しようとする、このステートメントはリクエストを拒否しません。このリクエストは、AllowAllUsersToViewAndManageThisGroup ステートメントで許可されます。他のユーザーがこれらのオペレーションを実行しようとする、リクエストは拒否されます。IAM コンソールでこのポリシーを作成している間、[書き込み] または [アクセス許可管理] アクセスレベルで定義されている IAM アクションを表示できます。これを行うには、[JSON] タブから [Visual editor (ビジュアルエディタ)] タブに切り替えます。アクセスレベルの詳細については、「[AWS Identity and Access Management のアクション、リソース、および条件キー](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllUsersToListAllGroups",
      "Effect": "Allow",
      "Action": "iam:ListGroups",
      "Resource": "*"
    },
    {
      "Sid": "AllowAllUsersToViewAndManageThisGroup",
      "Effect": "Allow",
      "Action": "iam:*Group*",
      "Resource": "arn:aws:iam::*:group/AllUsers"
    },
    {
      "Sid": "LimitGroupManagementAccessToSpecificUsers",
      "Effect": "Deny",
      "Action": [
        "iam:AddUserToGroup",
        "iam:CreateGroup",
        "iam:RemoveUserFromGroup",
        "iam>DeleteGroup",
        "iam:AttachGroupPolicy",
        "iam:UpdateGroup",
        "iam:DetachGroupPolicy",
```

```
        "iam:DeleteGroupPolicy",
        "iam:PutGroupPolicy"
    ],
    "Resource": "arn:aws:iam::*:group/AllUsers",
    "Condition": {
        "StringNotEquals": {
            "aws:username": [
                "srodriguez",
                "mjackson",
                "adesai"
            ]
        }
    }
}
]
```

## IAM: アカウントのパスワード要件の設定をプログラムによりコンソールで許可する

この例では、ユーザーがアカウントのパスワード要件を表示して更新することを許可する ID ベースのポリシーの作成方法を示します。パスワード要件は、アカウントメンバーのパスワードの複雑さの要件と必須のローテーション期間を指定します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

アカウントのパスワードポリシー要件の設定方法については、「[IAM ユーザー用のアカウントパスワードポリシーの設定](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GetAccountPasswordPolicy",
      "iam:UpdateAccountPasswordPolicy"
    ],
    "Resource": "*"
  }
}
```

## IAM: ユーザーパスに基づいた Policy Simulator API へのアクセス

この例は、パス Department/Development を持つユーザーにのみ Policy Simulator API の使用を許可する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:user/Department/Development/*"
    }
  ]
}
```

### Note

パス Department/Development があるユーザーにのみ Policy Simulator コンソールの使用を許可するポリシーを作成するには、[IAM: ユーザーパスに基づく Policy Simulator のコンソールへのアクセス](#)を参照してください。

## IAM: ユーザーパスに基づく Policy Simulator のコンソールへのアクセス

この例では、パス Department/Development を持つユーザーにのみ Policy Simulator コンソールの使用を許可する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

IAM ポリシーシミュレーターは <https://policysim.aws.amazon.com/> でアクセスできます。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "iam:GetPolicy",
      "iam:GetUserPolicy"
    ],
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Action": [
      "iam:GetUser",
      "iam:ListAttachedUserPolicies",
      "iam:ListGroupsForUser",
      "iam:ListUserPolicies",
      "iam:ListUsers"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:user/Department/Development/*"
  }
]
```

## IAM: IAM ユーザーに MFA デバイスの自己管理を許可する

この例は、IAM ユーザーが [多要素認証 \(MFA\)](#) デバイスを自己管理することを許可する ID ベースのポリシーの作成方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

### Note

このポリシーを設定した IAM ユーザーが MFA 認証されていない場合、このポリシーは、MFA を使用した認証に必要なアクションを除いて、すべての AWS アクションへのアクセスを拒否します。AWS にサインインしているユーザーにこれらのアクセス許可を追加する場合、これらの変更を確認するには、サインアウトしてからサインインする必要がある場合があります。

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Sid": "AllowListActions",
    "Effect": "Allow",
    "Action": [
      "iam:ListUsers",
      "iam:ListVirtualMFADevices"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowUserToCreateVirtualMFADevice",
    "Effect": "Allow",
    "Action": [
      "iam:CreateVirtualMFADevice"
    ],
    "Resource": "arn:aws:iam::*:mfa/*"
  },
  {
    "Sid": "AllowUserToManageTheirOwnMFA",
    "Effect": "Allow",
    "Action": [
      "iam:EnableMFADevice",
      "iam:GetMFADevice",
      "iam:ListMFADevices",
      "iam:ResyncMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowUserToDeactivateTheirOwnMFAOnlyWhenUsingMFA",
    "Effect": "Allow",
    "Action": [
      "iam:DeactivateMFADevice"
    ],
    "Resource": [
      "arn:aws:iam::*:user/${aws:username}"
    ],
    "Condition": {
      "Bool": {
        "aws:MultiFactorAuthPresent": "true"
      }
    }
  }
],
```

```
{
  "Sid": "BlockMostAccessUnlessSignedInWithMFA",
  "Effect": "Deny",
  "NotAction": [
    "iam:CreateVirtualMFADevice",
    "iam:EnableMFADevice",
    "iam:ListMFADevices",
    "iam:ListUsers",
    "iam:ListVirtualMFADevices",
    "iam:ResyncMFADevice"
  ],
  "Resource": "*",
  "Condition": {
    "BoolIfExists": {
      "aws:MultiFactorAuthPresent": "false"
    }
  }
}
```

## IAM: IAM ユーザーが自分の認証情報をプログラムまたはコンソールで更新することを許可する

この例では、IAM ユーザーが自分のアクセスキー、署名入り証明書、サービス固有の認証情報、パスワードを更新することを許可するアイデンティティベースのポリシーを作成する方法を示します。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",

```



```

        "iam:ChangePassword",
        "iam:GetUser",
        "iam:*ServiceSpecificCredential*",
        "iam:*SigningCertificate*"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
}
]
}

```

コンソールでユーザーが自分のパスワードを変更できる方法についての詳細は、「[the section called “IAM ユーザーが自分のパスワードを変更する方法”](#)」を参照してください。

## IAM: Organizations ポリシーのサービスの最終アクセス情報を表示

この例では、特定の Organizations ポリシーについて、サービスの最後にアクセスした情報の表示を許可する ID ベースのポリシーを作成する方法を示します。このポリシーでは、p-policy123 ID を使用してサービスコントロールポリシー (SCP) のデータを取得できます。レポートを生成して表示する人物は、AWS Organizations 管理アカウントの認証情報を使用して認証される必要があります。このポリシーにより、リクエストは組織内の任意の IAM エンティティのデータを取得できます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

必要なアクセス許可、トラブルシューティング、およびサポートされているリージョンを含む、最終アクセス情報に関する重要な情報については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowOrgsReadOnlyAndIamGetReport",
      "Effect": "Allow",
      "Action": [
        "iam:GetOrganizationsAccessReport",
        "organizations:Describe*",
        "organizations:List*"
      ],
      "Resource": "*"
    }
  ],
}

```

```
{
  "Sid": "AllowGenerateReportOnlyForThePolicy",
  "Effect": "Allow",
  "Action": "iam:GenerateOrganizationsAccessReport",
  "Resource": "*",
  "Condition": {
    "StringEquals": {"iam:OrganizationsPolicyId": "p-policy123"}
  }
}
```

## IAM: ユーザー、グループ、またはロールに適用できる管理ポリシーを制限する

この例は、IAM ユーザー、グループ、またはロールに適用できるカスタマー管理ポリシーと AWS 管理ポリシーを制限する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachUserPolicy",
      "iam:DetachUserPolicy"
    ],
    "Resource": "*",
    "Condition": {
      "ArnEquals": {
        "iam:PolicyARN": [
          "arn:aws:iam::*:policy/policy-name-1",
          "arn:aws:iam::*:policy/policy-name-2"
        ]
      }
    }
  }
}
```

## AWS: AWS マネージド IAM ポリシー以外のアカウント外のリソースへのアクセスを拒否します

ID ベースポリシーの `aws:ResourceAccount` を使用すると、ユーザーまたはロールでサービスが所有するアカウント内のリソースとのやり取りを要求するサービスを利用する機能に影響を与える可能性があります。

例外を用いてポリシーを作成して、AWS マネージド IAM ポリシーを許可することができます。AWS Organizations の外にあるサービス管理アカウントは、マネージド IAM ポリシーを所有しています。一覧表示して AWS マネージドポリシーを取得する IAM アクションは 4 つあります。ポリシー内のステートメント `AllowAccessToS3ResourcesInSpecificAccountsAndSpecificService1` の [NotAction](#) 要素でこれらのアクションを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToResourcesInSpecificAccountsAndSpecificService1",
      "Effect": "Deny",
      "NotAction": [
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListEntitiesForPolicy",
        "iam:ListPolicies"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ]
}
```

## AWS Lambda: Lambda 関数に Amazon DynamoDB テーブルへのアクセスを許可する

この例では、特定の Amazon DynamoDB テーブルへの読み取りおよび書き込みアクセスを許可する ID ベースのポリシーを作成する方法を示します。このポリシーでは、CloudWatch Logs をログファイルに書き込むこともできます。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーを使用するには、Lambda [サービスロール](#)にポリシーをアタッチします。サービスロールは、サービスがユーザーに代わってアクションを実行できるようにするためにアカウントに作成するロールです。このサービスのロールには、信頼ポリシーのプリンシパルとして AWS Lambda を含める必要があります。このポリシーの使用の詳細については、AWS セキュリティブログの「[Amazon DynamoDB テーブルへの AWS Lambda のアクセスを許可する AWS IAM ポリシーを作成する方法](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteTable",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchGetItem",
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchWriteItem",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/SampleTable"
    },
    {
      "Sid": "GetStreamRecords",
      "Effect": "Allow",
      "Action": "dynamodb:GetRecords",
      "Resource": "arn:aws:dynamodb:*:*:table/SampleTable/stream/* "
    },
    {
      "Sid": "WriteLogStreamsAndGroups",
      "Effect": "Allow",
      "Action": [
```

```

        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CreateLogGroup",
    "Effect": "Allow",
    "Action": "logs:CreateLogGroup",
    "Resource": "*"
  }
]
}

```

## Amazon RDS: 特定のリージョン内で RDS データベース全体へのアクセスを許可する

この例では、特定のリージョン内で RDS データベースへのフルアクセスを許可する ID ベースのポリシーを作成する方法を示します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "rds:*",
      "Resource": ["arn:aws:rds:region:*:*"]
    },
    {
      "Effect": "Allow",
      "Action": ["rds:Describe*"],
      "Resource": ["*"]
    }
  ]
}

```

## Amazon RDS: RDS データベースの復元をプログラムによりコンソールで許可する

この例は、RDS データベースの復元を許可する ID ベースのポリシーを作成する方法を示しています。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSnapshot",
        "rds>DeleteDBSnapshot",
        "rds:Describe*",
        "rds:DownloadDBLogFilePortion",
        "rds:List*",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyOptionGroup",
        "rds:RebootDBInstance",
        "rds:RestoreDBInstanceFromDBSnapshot",
        "rds:RestoreDBInstanceToPointInTime"
      ],
      "Resource": "*"
    }
  ]
}
```

## Amazon RDS: タグ所有者がタグ付けした RDS リソースへのフルアクセスを許可する

この例は、タグ付けした RDS リソースへのフルアクセスをタグの所有者に許可する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "rds:Describe*",
        "rds:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ],
```

```
{
  "Action": [
    "rds:DeleteDBInstance",
    "rds:RebootDBInstance",
    "rds:ModifyDBInstance"
  ],
  "Effect": "Allow",
  "Resource": "*",
  "Condition": {
    "StringEqualsIgnoreCase": {"rds:db-tag/Owner": "${aws:username}"}
  }
},
{
  "Action": [
    "rds:ModifyOptionGroup",
    "rds>DeleteOptionGroup"
  ],
  "Effect": "Allow",
  "Resource": "*",
  "Condition": {
    "StringEqualsIgnoreCase": {"rds:og-tag/Owner": "${aws:username}"}
  }
},
{
  "Action": [
    "rds:ModifyDBParameterGroup",
    "rds:ResetDBParameterGroup"
  ],
  "Effect": "Allow",
  "Resource": "*",
  "Condition": {
    "StringEqualsIgnoreCase": {"rds:pg-tag/Owner": "${aws:username}"}
  }
},
{
  "Action": [
    "rds:AuthorizeDBSecurityGroupIngress",
    "rds:RevokeDBSecurityGroupIngress",
    "rds>DeleteDBSecurityGroup"
  ],
  "Effect": "Allow",
  "Resource": "*",
  "Condition": {
    "StringEqualsIgnoreCase": {"rds:secgrp-tag/Owner": "${aws:username}"}
  }
}
```

```
    }
  },
  {
    "Action": [
      "rds:DeleteDBSnapshot",
      "rds:RestoreDBInstanceFromDBSnapshot"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:snapshot-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds:ModifyDBSubnetGroup",
      "rds>DeleteDBSubnetGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:subgrp-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds:ModifyEventSubscription",
      "rds:AddSourceIdentifierToSubscription",
      "rds:RemoveSourceIdentifierFromSubscription",
      "rds>DeleteEventSubscription"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:es-tag/Owner": "${aws:username}"}
    }
  }
]
}
```



## Amazon S3: Amazon Cognito ユーザーにバケット内のオブジェクトへのアクセスを許可する

この例では、Amazon Cognito ユーザーが特定の S3 バケット内のオブジェクトへのアクセスを許可する ID ベースのポリシーを作成する方法を示します。このポリシーは、名前に cognito、アプリケーションの名前、フェデレーテッドユーザーの ID (`${cognito-identity.:sub}` 変数) を含むオブジェクトのみへのアクセスを許可します。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

### Note

オブジェクトキーで使用される「サブ」値は、ユーザープール内のユーザーのサブ値ではなく、ID プール内のユーザーに関連付けられた ID です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListYourObjects",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
          ]
        }
      }
    },
    {
      "Sid": "ReadWriteDeleteYourObjects",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",

```

```
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::bucket-name/cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
  ]
}
]
```

Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、およびユーザー管理機能を提供します。ユーザーは、ユーザー名とパスワードを使用して直接サインインするか、Facebook、Amazon、Google などのサードパーティーを通じてサインインできます。

Amazon Cognito の主な 2 つのコンポーネントは、ユーザープールと ID プールです。ユーザープールは、アプリユーザーのサインアップとサインインオプションを提供するユーザーディレクトリです。ID プールは、AWS の他のサービスに対するアクセスをユーザーに許可します。ID プールとユーザープールは別々に使用することも、一緒に使用することもできます。

Amazon Cognito の詳細については、「[Amazon Cognito ユーザーガイド](#)」を参照してください。

## Amazon S3: フェデレーティッドユーザーが自分の S3 ホームディレクトリにプログラムおよびコンソールでアクセスすることを許可する

この例では、フェデレーティッドユーザーに S3 の自分のホームディレクトリバケットオブジェクトへのアクセスを許可する ID ベースのポリシーを作成する方法を示します。ホームディレクトリは、home フォルダや個々のフェデレーティッドユーザーのフォルダを含むバケットです。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーの `${aws:userid}` 変数は、`role-id:specified-name` に解決されます。フェデレーティッドユーザー ID の `role-id` の部分は、作成時にこのフェデレーティッドユーザーに割り当てられた一意の識別子です。詳細については、「[一意の識別子](#)」を参照してください。specified-name は、フェデレーションユーザーがロールを引き受けたときに AssumeRoleWithWebIdentity リクエストに渡される [RoleSessionName パラメータ](#) です。

ロール ID を表示するには、AWS CLI コマンド `aws iam get-role --role-name specified-name` を使用します。たとえば、フレンドリ名を John と指定すると、CLI はロール ID として AROAXXT2NJT7D3SIQN7Z6 を返します。この場合、フェデレーティッドユーザー ID は

AROAXXT2NJT7D3SIQN7Z6:John です。このポリシーは、フェデレーテッドユーザー John がプレフィックス AROAXXT2NJT7D3SIQN7Z6:John を使用して Amazon S3 バケットにアクセスすることを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::bucket-name",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "",
            "home/",
            "home/${aws:userid}/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::bucket-name/home/${aws:userid}",
        "arn:aws:s3:::bucket-name/home/${aws:userid}/*"
      ]
    }
  ]
}
```

```
    }  
  ]  
}
```

## Amazon S3: S3 バケットアクセス、しかし最新の MFA がなく、本番バケットが拒否された

この例では、オブジェクトの更新、追加、削除など、Amazon S3 管理者が任意のバケットへのアクセスを許可する ID ベースのポリシーを作成する方法を示します。ただし、過去 30 分以内にユーザーが [多要素認証 \(MFA\)](#) を使用してサインインしていない場合、Production バケットへのアクセスは明示的に拒否されます。このポリシーは、コンソールで、AWS CLI または AWS API を使用してプログラムでこのアクションを実行するために必要なアクセス許可を付与します。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、[ポリシーの作成](#) または [ポリシーの編集](#) の手順に従います。

このポリシーでは、長期的なユーザーアクセスキーを使用して Production バケットへのプログラムによるアクセスを許可することはありません。これは、`aws:MultiFactorAuthAge` 条件キーと `NumericGreaterThanIfExists` 条件演算子を使用して行います。このポリシー条件は、MFA が存在しない場合、または MFA の経過時間が 30 分を超える場合に `true` を返します。このような場合、アクセスが拒否されます。Production バケットにプログラムでアクセスするには、S3 管理者は [GetSessionToken](#) API オペレーションを使用して直近 30 分以内に生成された一時的な認証情報を使用する必要があります。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "ListAllS3Buckets",  
      "Effect": "Allow",  
      "Action": ["s3:ListAllMyBuckets"],  
      "Resource": "arn:aws:s3:::*"  
    },  
    {  
      "Sid": "AllowBucketLevelActions",  
      "Effect": "Allow",  
      "Action": [  
        "s3:ListBucket",  
        "s3:GetBucketLocation"  
      ],  
      "Resource": "arn:aws:s3:::*"  
    },  
  ],  
}
```

```

    {
      "Sid": "AllowBucketObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3::*/*"
    },
    {
      "Sid": "RequireMFAForProductionBucket",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::Production/*",
        "arn:aws:s3:::Production"
      ],
      "Condition": {
        "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "1800"}
      }
    }
  ]
}

```

## Amazon S3: IAM ユーザーが自分の S3 ホームディレクトリにプログラムによりコンソールでアクセスすることを許可する

この例では、IAM ユーザーが S3 の自分のホームディレクトリバケットオブジェクトへのアクセスを許可する ID ベースのポリシーの、作成方法を示します。ホームディレクトリは、home フォルダや個々のユーザーのフォルダを含むバケットです。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーは、IAM ロールを使用するときに `aws:username` 変数を使用できないため、IAM ロールを使用するときには機能しません。プリンシパルキーバリューの詳細については、「[プリンシパルキーの値](#)」をご参照ください。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "S3ConsoleAccess",
    "Effect": "Allow",
    "Action": [
      "s3:GetAccountPublicAccessBlock",
      "s3:GetBucketAcl",
      "s3:GetBucketLocation",
      "s3:GetBucketPolicyStatus",
      "s3:GetBucketPublicAccessBlock",
      "s3:ListAccessPoints",
      "s3:ListAllMyBuckets"
    ],
    "Resource": "*"
  },
  {
    "Sid": "ListObjectsInBucket",
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::bucket-name",
    "Condition": {
      "StringLike": {
        "s3:prefix": [
          "",
          "home/",
          "home/${aws:username}/*"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::bucket-name/home/${aws:username}",
      "arn:aws:s3:::bucket-name/home/${aws:username}/*"
    ]
  }
]
}
```

## Amazon S3: 特定の S3 バケットに対する管理の制限

この例では、Amazon S3 バケットの管理を特定のバケットに制限する ID ベースのポリシーを作成する方法を示します。このポリシーは、すべての Amazon S3 アクションを実行するアクセス権限を付与しますが、Amazon S3 以外のすべての AWS のサービス へのアクセスを拒否します。次の例を参照してください。このポリシーによると、S3 バケットまたは S3 オブジェクトリソースで実行できる Amazon S3 アクションにのみアクセスできます。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

このポリシーが、このポリシーによって拒否されたアクションを許可する他のポリシー ([AmazonS3FullAccess](#) または [AmazonEC2FullAccess](#) AWS 管理ポリシーなど) と組み合わせられて使用される場合、アクセスは拒否されます。これは、明示的な拒否のステートメントは許可のステートメントより優先されるからです。詳細については、「[the section called “アカウント内でのリクエストの許可または拒否の決定”](#)」を参照してください。

### Warning

[NotAction](#) および [NotResource](#) は、注意して使用する必要がある高度なポリシー要素です。このポリシーは、Amazon S3 を除くすべての AWS サービスへのアクセスを拒否します。このポリシーをユーザーにアタッチすると、他のサービスにアクセス許可を付与する他のポリシーは無視され、アクセスは拒否されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::bucket-name",
        "arn:aws:s3:::bucket-name/*"
      ]
    },
    {
      "Effect": "Deny",
      "NotAction": "s3:*",
      "NotResource": [
```

```
        "arn:aws:s3:::bucket-name",
        "arn:aws:s3:::bucket-name/*"
    ]
}
]
```

## Amazon S3: S3 バケットのオブジェクトへの読み取りおよび書き込みアクセスを許可する

この例は、特定の S3 バケット内のオブジェクトへの Read および Write アクセスを許可する ID ベースのポリシーを作成する方法を示しています。このポリシーでは、AWS API または AWS CLI から、このアクションをプログラムで完了するために必要なアクセス権を許可します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

s3:\*Object アクションは、アクション名の一部にワイルドカードを使用します。この AllObjectActions ステートメントでは、GetObject、DeleteObject、PutObject、および「Object」という単語で終わるその他の Amazon S3 アクションを使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::bucket-name"]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": ["arn:aws:s3:::bucket-name/*"]
    }
  ]
}
```



**Note**

Amazon S3 バケットのオブジェクトへの Read と Write アクセスを許可し、コンソールアクセスのための追加のアクセス許可も含めるには、「[Amazon S3: S3 バケットのオブジェクトへの読み取りおよび書き込みアクセスをプログラムによりコンソールで許可する](#)」を参照してください。

## Amazon S3: S3 バケットのオブジェクトへの読み取りおよび書き込みアクセスをプログラムによりコンソールで許可する

この例は、特定の S3 バケット内のオブジェクトへの Read および Write アクセスを許可する ID ベースのポリシーを作成する方法を示しています。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、[ポリシーの作成](#)または[ポリシーの編集](#)の手順に従います。

s3:\*Object アクションは、アクション名の一部にワイルドカードを使用します。この AllObjectActions ステートメントでは、GetObject、DeleteObject、PutObject、および「Object」という単語で終わるその他の Amazon S3 アクションを使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
```

```
        "Action": "s3:ListBucket",
        "Resource": ["arn:aws:s3:::bucket-name"]
    },
    {
        "Sid": "AllObjectActions",
        "Effect": "Allow",
        "Action": "s3:*Object",
        "Resource": ["arn:aws:s3:::bucket-name/*"]
    }
]
}
```

## IAM ポリシーを管理する

IAM は、すべてのタイプの IAM ポリシー (管理ポリシーとインラインポリシー) を作成および管理するためのツールを提供します。IAM ID (IAMユーザー、グループ、またはロール) にアクセス許可を追加するには、ポリシーを作成し、ポリシーを検証してから、ポリシーを ID にアタッチします。アイデンティティに複数のポリシーをアタッチし、各ポリシーに複数のアクセス許可を含めることができます。

詳細については、以下のリソースを参照してください。

- 各種 IAM ポリシーの詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。
- IAM 内でのポリシーの使用についての一般情報は、「[AWS リソースのアクセス管理](#)」を参照してください。
- 特定の IAM アイデンティティに対して複数のポリシーが有効な場合のアクセス許可の評価方法については、「[ポリシーの評価論理](#)」を参照してください。
- AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。

### トピック

- [IAM ポリシーの作成](#)
- [IAM ポリシーの検証](#)
- [アクセスアクティビティに基づいてポリシーを生成する](#)
- [IAM Policy Simulator を使用した IAM ポリシーのテスト](#)
- [IAM ID のアクセス許可の追加および削除](#)

- [IAM ポリシーのバージョニング](#)
- [IAM ポリシーの編集](#)
- [IAM ポリシーを削除する](#)
- [最終アクセス情報を使用した AWS のアクセス許可の調整](#)

## IAM ポリシーの作成

[ポリシー](#) は、ID またはリソースにアタッチされたときに、そのアクセス許可を定義するエンティティです。AWS Management Console、AWS CLI、または AWS API を使用して、IAM でカスタマー管理ポリシーを作成できます。カスタマー管理ポリシーは、独自の AWS アカウント で管理するスタンドアロンポリシーです。その後、ポリシーを AWS アカウント のアイデンティティ (ユーザー、グループ、またはロール) にアタッチできます。

IAM のアイデンティティにアタッチされたポリシーは、アイデンティティベースのポリシーと呼ばれます。アイデンティティベースのポリシーには、AWS 管理ポリシー、カスタマー管理ポリシー、およびインラインポリシーがあります。AWS 管理ポリシーは、AWS によって作成されて管理されます。それらを使用することはできますが、管理することはできません。インラインポリシーは、IAM グループ、ユーザー、またはロールに直接作成して埋め込むポリシーです。インラインポリシーは、他のアイデンティティで再利用したり、それが存在するアイデンティティ以外で管理したりすることはできません。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

インラインポリシーではなくカスタマー管理ポリシーを使用します。また、AWS 管理ポリシーではなくカスタマー管理ポリシーを使用することもお勧めします。通常、AWS 管理ポリシーでは、広範な管理アクセス許可または読み取り専用アクセス許可が提供されます。セキュリティを最大限に高めるには、[最小特権を付与](#)します。最小特権は、特定のジョブタスクの実行に必要なアクセス許可のみを付与します。

IAM ポリシーを作成または編集すると、AWS は、ポリシー検証を自動的に実行し、最小限の権限で効果的なポリシーを作成するのに役立ちます。AWS Management Console では、IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックを提供します。ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。IAM Access Analyzer のポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

AWS Management Console、AWS CLI、または AWS API を使用して、IAM でカスタマー管理ポリシーを作成できます。AWS CloudFormation テンプレートを使用してポリシーを追加または更

新する方法については、「AWS CloudFormation ユーザーガイド」の「[AWS Identity and Access Management リソースタイプリファレンス](#)」を参照してください。

## トピック

- [IAM ポリシーの作成 \(コンソール\)](#)
- [IAM ポリシーの作成 \(AWS CLI\)](#)
- [IAM ポリシーの作成 \(AWS API\)](#)

## IAM ポリシーの作成 (コンソール)

[ポリシー](#) は、ID またはリソースにアタッチされたときに、そのアクセス許可を定義するエンティティです。AWS Management Console を使用して、IAM でカスタマー管理ポリシーを作成できます。カスタマー管理ポリシーは、独自の AWS アカウント で管理するスタンドアロンポリシーです。その後、ポリシーを AWS アカウント のアイデンティティ (ユーザー、グループ、またはロール) にアタッチできます。

## トピック

- [IAM ポリシーの作成](#)
- [JSON エディターを使用したポリシーの作成](#)
- [ビジュアルエディタでのポリシーの作成](#)
- [既存の管理ポリシーのインポート](#)

## IAM ポリシーの作成

AWS Management Console でカスタマー管理ポリシーを作成するには、次のいずれかの方法を使用します。

- [JSON](#) — 発行された [アイデンティティベースのポリシーの例](#) を貼り付けてカスタマイズします。
- [ビジュアルエディタ](#) — ビジュアルエディタで最初から新しいポリシーを構築できます。ビジュアルエディタを使用する場合は、JSON 構文を理解する必要はありません。
- [インポート](#) — アカウント内から管理ポリシーをインポートしてカスタマイズします。以前に作成した AWS 管理ポリシーまたはカスタマー管理ポリシーをインポートできます。

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。

## JSON エディターを使用したポリシーの作成

[JSON] オプションを選択して、JSON でポリシーを入力または貼り付けることができます。この方法は、アカウントで使用する[ポリシー例](#)をコピーするのに便利です。または、独自の JSON ポリシードキュメントを JSON エディタに入力することもできます。[JSON] オプションを使用してビジュアルエディタと JSON を切り替えて、ビューを比較することもできます。

JSON エディターでポリシーを作成または編集すると、IAM はポリシー検証を実行し、効果的なポリシーの作成に役立ちます。IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つアクション可能な推奨事項を含む追加のポリシーチェックを提供します。

JSON [ポリシードキュメント](#)は 1 つ以上のステートメントで構成されます。各ステートメントには、同じ効果エフェクト (Allow または Deny) を共有するすべてのアクションが含まれ、同じリソースと条件がサポートされている必要があります。1 つのアクションですべてのリソース ("\*") を指定する必要があり、別のアクションが特定のリソースの Amazon リソースネーム (ARN) に対応している場合、それらは 2 つの別々の JSON ステートメントに含める必要があります。ARN 形式の詳細については、「AWS 全般のリファレンス ガイド」の「[Amazon リソースネーム \(ARN\)](#)」を参照してください。IAM ポリシーの一般情報については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。IAM ポリシー言語については、「[IAM JSON ポリシーリファレンス](#)」を参照してください。

JSON ポリシーエディタを使用してポリシーを作成するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [Create policy] (ポリシーを作成) を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. JSON ポリシードキュメントを入力するか貼り付けます。IAM ポリシー言語の詳細については、「[IAM JSON ポリシーリファレンス](#)」を参照してください。
6. [ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。

### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して

visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

- ( オプション ) AWS Management Console でポリシーを作成または編集するときに、AWS CloudFormation テンプレートで使用できる JSON または YAML ポリシーテンプレートを生成できます。  
  
これを行うには、ポリシーエディタで [アクション] を選択し、次に [CloudFormation テンプレートを生成] を選択します。AWS CloudFormation の詳細については、AWS CloudFormation ユーザーガイドの「[AWS Identity and Access Management リソースタイプリファレンス](#)」を参照してください。
- ポリシーにアクセス権限を追加し終えたら、[次へ] を選択します。
- [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。[このポリシーで定義されているアクセス許可] を確認して、ポリシーによって付与されたアクセス許可を確認します。
- (オプション) タグをキー - 値のペアとしてアタッチして、メタデータをポリシーに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
- [Create Policy] (ポリシーの作成) をクリックして、新しいポリシーを保存します。

ポリシーを作成したら、グループ、ユーザー、またはロールにアタッチできます。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

## ビジュアルエディタでのポリシーの作成

IAM コンソールのビジュアルエディタは、JSON 構文を記述することなくポリシーを作成する方法をガイドします。ビジュアルエディタを使用してポリシーを作成する方法の例については、「[the section called “アイデンティティへのアクセスコントロール”](#)」を参照してください。

ビジュアルエディタを使用してポリシーを作成するには

- AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
- 左側のナビゲーションペインで、[ポリシー] を選択します。
- [Create policy] (ポリシーを作成) を選択します。
- [ポリシーエディター] セクションで、[サービスを選択] セクションを見つけて、AWS サービスを選択します。上部の検索ボックスを使用して、サービスのリストの結果を制限することができ

まず、ビジュアルエディタのアクセス許可ブロック内で選択できるサービスは 1 つだけです。複数のサービスにアクセス許可を付与するには、[さらにアクセス許可を追加する] を選択して、複数のアクセス許可ブロックを追加します。

5. [許可されるアクション] で、ポリシーに追加するアクションを選択します。アクションは次の方法で選択できます。

- すべてのアクションのチェックボックスをオンにします。
- [アクションを追加] を選択して、特定のアクションの名前を入力します。ワイルドカード (\*) を使用して、複数のアクションを指定できます。
- [アクセスレベル] グループの 1 つを選択して、アクセスレベルのすべてのアクション ([読み取り]、[書き込み]、または [リスト] など) を選択します。
- それぞれの [アクセスレベル] グループを展開して、個々のアクションを選択します。

デフォルトでは、作成しているポリシーが選択するアクションを許可します。その代わりに選択したアクションを拒否するには、[Switch to deny permissions (アクセス許可の拒否に切り替え)] を選択します。[IAM はデフォルトでは拒否](#)されるため、ユーザーが必要とするアクションとリソースのみに対してアクセス許可を許可することを、セキュリティのベストプラクティスとしてお勧めします。別のステートメントまたはポリシーによって個別に許可されるアクセス許可を上書きする場合のみ、アクセス許可を拒否する JSON ステートメントを作成します。これにより、アクセス許可のトラブルシューティングがより困難になる可能性があるため、拒否ステートメントの数は最小限に制限することをお勧めします。

6. [リソース] では、前のステップで選択したサービスとアクションが[特定のリソース](#)の選択をサポートしていない場合は、すべてのリソースが許可され、このセクションを編集することはできません。

[リソースレベルのアクセス許可](#)をサポートする 1 つ以上のアクションを選択した場合、ビジュアルエディタでそれらのリソースが一覧表示されます。[リソース] を展開して、ポリシーのリソースを指定できます。

リソースは次の方法で指定できます。

- [ARN を追加] を選択して、それらの Amazon リソースネーム (ARN) 別にリソースを指定します。ビジュアル ARN エディタを使用するか、ARN を手動でリストすることができます。ARN 構文の詳細については、「AWS 全般のリファレンス Guide」の「[Amazon リソースネーム \(ARN\)](#)」を参照してください。ポリシーの Resource 要素で ARN を使用する方法については、「[IAM JSON ポリシー要素Resource](#)」を参照してください。



- リソースの横にある [このアカウント内のすべて] を選択して、そのタイプのすべてのリソースにアクセス許可を付与します。
  - [すべて] を選択し、そのサービスのすべてのリソースを選択します。
7. (オプション) [リクエスト条件 - オプション] を選択して、作成するポリシーに条件を追加します。条件によって JSON ポリシーステートメントの効果が制限されます。例えば、特定の時間範囲内でそのユーザーのリクエストが発生した場合にのみ、ユーザーがリソースに対してアクションを実行できるように指定できます。一般的に使用される条件を使用して、Multi-Factor Authentication(MFA) デバイスでユーザーを認証する必要があるかどうかを制限することもできます。または、リクエストの発行元を特定範囲内の IP アドレスに限定できます。ポリシー条件で使用できるすべてのコンテキストキーのリストについては、「[サービス認証リファレンス](#)」の「[AWS サービスのアクション、リソース、および条件キー](#)」を参照してください。

条件は次の方法で選択できます。

- 一般的に使用される条件を選択するには、チェックボックスを使用します。
- 他の条件を指定するには、[別の条件を追加] を選択します。条件の [条件キー]、[修飾子]、[演算子] を選択し、[値] に入力します。複数の値を追加するには、[追加] を選択します。値は、論理 "OR" 演算子によって接続されていると見なすことができます。完了したら、[条件を追加] を選択します。

複数の条件を追加するには、[別の条件を追加] を選択します。必要に応じて操作を繰り返します。各条件は、この 1 つのビジュアルエディタのアクセス許可ブロックにのみ適用されます。アクセス許可ブロックが一致すると見なされるためには、すべての条件が満たされている必要があります。つまり、論理 "AND" 演算子によって接続される条件を考慮してください。

Condition 要素の詳細については、「[IAM JSON ポリシーリファレンス](#)」の「[IAM JSON ポリシー要素Condition](#)」を参照してください。

8. さらにアクセス許可ブロックを追加するには、[さらにアクセス許可を追加] を選択します。各ブロックに対して、ステップ 2 から 5 を繰り返します。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。



9. (オプション) AWS Management Console でポリシーを作成または編集するときに、AWS CloudFormation テンプレートで使用できる JSON または YAML ポリシーテンプレートを生成できます。

これを行うには、ポリシーエディタで [アクション] を選択し、次に [CloudFormation テンプレートを生成] を選択します。AWS CloudFormation の詳細については、AWS CloudFormation ユーザーガイドの「[AWS Identity and Access Management リソースタイプリファレンス](#)」を参照してください。

10. ポリシーにアクセス権限を追加し終えたら、[次へ] を選択します。
11. [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。[このポリシーで定義されているアクセス許可] を確認し、意図したアクセス許可を付与したことを確認します。
12. (オプション) タグをキー - 値のペアとしてアタッチして、メタデータをポリシーに追加します。IAM におけるタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
13. [Create Policy] (ポリシーの作成) をクリックして、新しいポリシーを保存します。

ポリシーを作成したら、グループ、ユーザー、またはロールにアタッチできます。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

### 既存の管理ポリシーのインポート

新しいポリシーを作成する簡単な方法は、必要なアクセス許可の少なくとも一部を持っているアカウント内の既存の管理ポリシーをインポートすることです。次に、新しい要件に合わせてそのポリシーをカスタマイズできます。

インラインポリシーをインポートすることはできません。管理ポリシーとインラインポリシーの違いについては、「[管理ポリシーとインラインポリシー](#)」を参照してください。

ビジュアルエディタで既存の管理ポリシーをインポートするには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [Create policy] (ポリシーを作成) を選択します。
4. [ポリシーエディタ] で [Visual] を選択し、ページの右側で [アクション]、[ポリシーをインポート] と選択します。

5. [ポリシーをインポート] ウィンドウで、新しいポリシーに含めるポリシーと最もよく一致する管理ポリシーを選択します。上部の検索ボックスを使用して、ポリシーのリストの結果を制限することができます。

6. [ポリシーをインポート] を選択します。

インポートされたポリシーは、ポリシーの下部にあるアクセス許可ブロックに追加されます。

7. ビジュアルエディタを使用して、ポリシーをカスタマイズする JSON を選択します。続いて、[Next] (次へ) を選択します。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

8. [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。これらの設定は後で編集できません。[このポリシーで定義されているアクセス許可] を確認し、[ポリシーを作成] を選択して作業を保存します。

[JSON] エディタで既存の管理ポリシーをインポートするには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [Create policy] (ポリシーを作成) を選択します。
4. [ポリシーエディタ] セクションで [JSON] オプションを選択し、ページの右側で [アクション]、[ポリシーをインポート] と選択します。
5. [ポリシーをインポート] ウィンドウで、新しいポリシーに含めるポリシーと最もよく一致する管理ポリシーを選択します。上部の検索ボックスを使用して、ポリシーのリストの結果を制限することができます。
6. [ポリシーをインポート] を選択します。

インポートされたポリシーのステートメントは、JSON ポリシーの最後に追加されます。

7. JSON でポリシーをカスタマイズします。[ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。JSON でポリシーを

カスタマイズするか、[Visual editor (ビジュアルエディタ)] を選択します。続いて、[Next] (次へ) を選択します。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

8. [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。これらを後で編集することはできません。[このポリシーで定義されているアクセス許可] ポリシーを確認し、[ポリシーを作成] を選択して作業を保存します。

ポリシーを作成したら、グループ、ユーザー、またはロールにアタッチできます。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。

## IAM ポリシーの作成 (AWS CLI)

[ポリシー](#) は、ID またはリソースにアタッチされたときに、そのアクセス許可を定義するエンティティです。AWS CLI を使用して、IAM でカスタマー管理ポリシーを作成できます。カスタマー管理ポリシーは、独自の AWS アカウント で管理するスタンドアロンポリシーです。[ベストプラクティス](#)として、IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保することをお勧めします。[ポリシーを検証する](#)ことで、ポリシーを AWS アカウント のアイデンティティ (ユーザー、グループ、またはロール) にアタッチする前に、エラーやレコメンデーションに対応できます。

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。

### IAM ポリシーの作成 (AWS CLI)

AWS Command Line Interface (AWS CLI) を使用して、IAM カスタマー管理ポリシーまたはインラインポリシーを作成できます。

カスタマー管理ポリシーを作成するには (AWS CLI)

以下のコマンドを使用します。

- [create-policy](#)

IAM アイデンティティ (グループ、ユーザー、またはロール) のインラインポリシーを作成するには (AWS CLI)

以下のいずれかのコマンドを使用します。

- [put-group-policy](#)
- [put-role-policy](#)
- [put-user-policy](#)

**Note**

IAM を使用して [サービスリンクロール](#) にインラインポリシーを埋め込むことはできません。

カスタマー管理ポリシーを検証するには (AWS CLI)

次の IAM Access Analyzer コマンドを使用します。

- [ポリシーの検証](#)

## IAM ポリシーの作成 (AWS API)

[ポリシー](#) は、ID またはリソースにアタッチされたときに、そのアクセス許可を定義するエンティティです。AWS API を使用して、IAM でカスタマー管理ポリシーを作成できます。カスタマー管理ポリシーは、独自の AWS アカウント で管理するスタンドアロンポリシーです。[ベストプラクティス](#)として、IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保することをお勧めします。[ポリシーを検証する](#)ことで、ポリシーを AWS アカウント のアイデンティティ (ユーザー、グループ、またはロール) にアタッチする前に、エラーやレコメンデーションに対応できます。

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

## IAM ポリシーの作成 (AWS API)

AWS API を使用して、IAM カスタマー管理ポリシーまたはインラインポリシーを作成できます。

カスタマー管理ポリシーを作成するには (AWS API)

次のオペレーションを呼び出します。

- [CreatePolicy](#)

IAM アイデンティティ (グループ、ユーザー、またはロール) のインラインポリシーを作成するには (AWS API)

以下のいずれかのオペレーションを呼び出します。

- [PutGroupPolicy](#)
- [PutRolePolicy](#)
- [PutUserPolicy](#)

#### Note

IAM を使用して「[サービスにリンクされたロール](#)」にインラインポリシーを埋め込むことはできません。

カスタマー管理ポリシーを編集するには (AWS API)

次の IAM Access Analyzer オペレーションを呼び出します。

- [検証ポリシー](#)

## IAM ポリシーの検証

[ポリシー](#)は、[IAM ポリシーの文法](#)を使用して記述された JSON ドキュメントです。ユーザー、グループ、ロールなどの IAM エンティティにポリシーをアタッチすると、そのエンティティへのアクセス許可が付与されます。

AWS Management Console を使用して IAM アクセスコントロールポリシーを作成または編集すると、AWS はそれらを自動的に調べて、IAM ポリシーの文法に準拠していることを確認します。AWS により、ポリシーが文法に基づいていないと判断された場合、ポリシーの修正が求められます。

IAM Access Analyzer では、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックが提供されます。IAM Access Analyzer のポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」IAM Access Analyzer ポリシーの検証を参照して

ください。IAM Access Analyzerによって返される警告、エラー、および提案のリストを表示するには、[IAM Access Analyzer ポリシーチェックリファレンス](#)を参照してください。

## 検証スコープ

AWS は、JSON ポリシーの構文と文法を確認します。ARN の形式が適切で、アクション名と条件キーが正しいことも確認します。

## ポリシーの検証へのアクセス

JSON ポリシーを作成するか、AWS Management Console で既存のポリシーを編集すると、ポリシーが自動的に検証されます。ポリシー構文が有効でない場合は、通知を受け取り、続行する前に問題を修正する必要があります。AWS Management Console のアクセス許可がある場合、IAM Access Analyzer ポリシー検証の結果は `access-analyzer:ValidatePolicy` に自動的に返されます。AWS API または AWS CLI を使用してポリシーを検証することもできます。

## 既存のポリシー

ポリシーエンジンへの最新の更新の前に作成されたか、最後に保存されたために、有効でない既存のポリシーがある可能性があります。[ベストプラクティス](#)として、IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保することをお勧めします。既存のポリシーを開き、生成されたポリシー検証結果を確認することをお勧めします。ポリシー構文のエラーを修正せずに既存のポリシーを編集して保存することはできません。

## アクセスアクティビティに基づいてポリシーを生成する

管理者またはデベロッパーは、IAM エンティティ (ユーザーまたはロール) に必要な権限を超えるアクセス許可を付与することがあります。IAM には、付与するアクセス許可を絞り込むのに役立つオプションがいくつか用意されています。1 つのオプションは、エンティティのアクセスアクティビティに基づく IAM ポリシーを生成することです。IAM Access Analyzer は AWS CloudTrail ログを確認し、指定した日付範囲内のロールが使用したアクセス許可を含むポリシーテンプレートを生成します。テンプレートを使用して、特定のユースケースをサポートするために必要なアクセス許可のみを付与する、きめ細かなアクセス許可を持つポリシーを作成できます。

たとえば、自分がデベロッパーであり、エンジニアリングチームがプロジェクトに取り組んで新しいアプリケーションを作成しているとします。実験が積極的にできるように、チームが迅速に活動できるように、アプリケーションの開発中は幅広い権限を持つロールを設定します。アプリケーションを稼働させる準備が整ったとします。アプリケーションを本番稼働用アカウントで起動する前に、アプリケーションが機能するためにロールに必要な最低限のアクセス許可を特定する必要があります。これにより、[最小アクセス権限の付与](#)のベストプラクティスに準拠できるようになります。開発アカウ

ントでアプリケーションに使用していたロールのアクセスアクティビティに基づいて、ポリシーを生成できます。生成されたポリシーをさらに絞り込み、そのポリシーを本番稼働用アカウントのエンティティにアタッチできます。

IAM Access Analyzer ポリシーの生成の詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

## IAM Policy Simulator を使用した IAM ポリシーのテスト

IAM ポリシーを使用する方法と理由の詳細については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

IAM Policy Simulator コンソールは <https://policysim.aws.amazon.com/> でアクセスできます。

### Important

Policy Simulator は、AWS での実際の環境とは異なる結果を出力することがあります。Policy Simulator によるテスト後は、実際の AWS 環境とポリシーを照合して、得られた結果が目的にあっているか確認することをお勧めします。詳細については、「[IAM Policy Simulator のしくみ](#)」を参照してください。

### [IAM Policy Simulator の開始方法](#)

IAM Policy Simulator を使用すると、アイデンティティベースのポリシーおよび IAM アクセス許可の境界をテストして、トラブルシューティングが行えます。以下に示しているのは、Policy Simulator で可能ないくつかの一般的な処理です。

- AWS アカウントの IAM ユーザー、ユーザーグループ、またはロールにアタッチされている、アイデンティティベースのポリシーをテストします。複数のポリシーがユーザー、ユーザーグループ、またはロールにアタッチされている場合は、すべてのポリシーまたは選択した個別のポリシーをテストできます。特定のリソースで選択されたポリシーについて、どのアクションが許可または拒否されているかをテストできます。
- IAM エンティティに対する[アクセス許可の境界](#)の効果をテストし、トラブルシューティングします。一度にシミュレートできるアクセス許可の境界は 1 つのみです。
- Amazon S3 バケット、Amazon SQS キュー、Amazon SNS トピック、Amazon S3 Glacier ポールトなど、AWS リソースにアタッチされている IAM ユーザーのリソースベースポリシーの効果をテストします。ポリシーシミュレーター内で、IAM ユーザーに対しリソースベースのポリシーを使



用するには、シミュレーションにリソースを含める必要があります。また、チェックボックスをオンにして、シミュレーションにそのリソースのポリシーも含める必要もあります。

**Note**

リソースベースのポリシーに関するシミュレーションでは、IAM ロールはサポートされていません。

- [AWS Organizations](#) の組織のメンバーである AWS アカウント であれば、アイデンティティベースのポリシーに対するサービスコントロールポリシー (SCP) の影響をテストできます。

**Note**

ポリシーシミュレーターは、何らかの条件を持つ SCP の評価を行いません。

- ユーザー、ユーザーグループ、またはロールにまだアタッチされていない、新しいアイデンティティベースのポリシーを (Policy Simulator に入力またはコピーして) テストします。これらはシミュレーションでのみ使用され、保存されません。リソースベースのポリシーを、Policy Simulator に入力またはコピーすることはできません。
- 指定したサービス、アクション、リソースを含むアイデンティティベースのポリシーをテストします。例えば、ポリシーによってエンティティが Amazon S3 サービスの特定のバケットで ListAllMyBuckets、CreateBucket、および DeleteBucket アクションの実行を許可されていることをテストできます。
- テストするポリシー内の Condition 要素に含まれる IP アドレスや日付などのコンテキストキーを提供して、実際のシナリオをシミュレートします。

**Note**

シミュレーション内のアイデンティティベースのポリシーに、タグを明示的にチェックする Condition 要素が含まれていない場合、ポリシーシミュレーターは入力として提供されたタグをシミュレートしません。

- アイデンティティベースのポリシーにおいて、特定のリソースやアクションへのアクセスを許可または拒否しているのは、どのステートメントなのかを特定します。

## トピック

- [IAM Policy Simulator のしくみ](#)



- [IAM Policy Simulator の使用に必要なアクセス許可](#)
- [IAM Policy Simulator の使用方法 \(コンソール\)](#)
- [AWS CLI IAM Policy Simulator の使用 \( および AWS API\)](#)

## IAM Policy Simulator のしくみ

Policy Simulator は、アイデンティティベースのポリシー内のステートメントと、シミュレーション中に提供した入力の評価を行います。Policy Simulator は、AWS での実際の環境とは異なる結果を出力することがあります。Policy Simulator によるテスト後は、実際の AWS 環境とポリシーを照合して、得られた結果が目的にあっているか確認することをお勧めします。

Policy Simulator は、以下の点で実際の AWS 環境とは異なります。

- Policy Simulator では、現実の AWS サービスリクエストは生成しないため、実際の AWS 環境に不要な変更を加える可能性のあるリクエストのテストを安全に行えます。Policy Simulator は、本番環境での実際のコンテキストキー値を考慮しません。
- Policy Simulator では、選択されたアクションの実行はシミュレートされません。つまり、シミュレートされたリクエストに対するレスポンスをレポートしません。返される結果は、リクエストされたアクションが許可されるか拒否されるかだけです。
- Policy Simulator 内でポリシーを編集した場合、それらの変更は Policy Simulator にのみ影響を与えます。AWS アカウント 内の対応するポリシーが変更されることはありません。
- 何らかの条件を持つサービスコントロールポリシー (SCP) をテストすることはできません。
- Policy Simulator は、IAM ロールとユーザーのクロスアカウントアクセスのシミュレーションをサポートしていません。

### Note

IAM Policy Simulator は、認証用の[グローバル条件キー](#)をサポートしているサービスを識別しません。例えば、あるサービスが [aws:TagKeys](#) をサポートしていないことを、Policy Simulator は識別しません。

## IAM Policy Simulator の使用に必要なアクセス許可

Policy Simulator コンソールまたは Policy Simulator API を使用して、ポリシーをテストできます。コンソールを使用するユーザーは、デフォルトで、ユーザー、ユーザーグループ、またはロールにま

だアタッチされていないポリシーを (Policy Simulator コンソールに入力またはコピーすることで) テストできます。それらのポリシーはシミュレーションでのみ使用され、機密情報が公開されることはありません。API ユーザーには、アタッチされていないポリシーをテストするアクセス権限が必要です。コンソールまたは API ユーザーに、AWS アカウント の IAM ユーザー、ユーザーグループ、またはロールにアタッチされているポリシーをテストすることを許可できます。そのためには、それらのポリシーを取得するアクセス許可を付与する必要があります。リソースベースのポリシーをテストするには、リソースのポリシーを取得するための権限がユーザーに必要です。

ポリシーのシミュレートをユーザーに許可するコンソールまたは API ポリシーの例については、「[the section called “ポリシーの例: AWS Identity and Access Management \(IAM\)”](#)」を参照してください。

### Policy Simulator コンソールの使用に必要なアクセス許可

AWS アカウント の IAM ユーザー、ユーザーグループ、またはロールにアタッチされているポリシーをテストすることをユーザーに許可できます。そのためには、それらのポリシーを取得するアクセス許可をユーザーに付与する必要があります。リソースベースのポリシーをテストするには、リソースのポリシーを取得するための権限がユーザーに必要です。

ユーザー、ユーザーグループ、またはロールにアタッチされたポリシーのために Policy Simulator コンソールの使用を許可するポリシー例を確認するには、「[IAM: Policy Simulator のコンソールへのアクセス](#)」を参照してください。

特定のパスがあるユーザーにのみ Policy Simulator コンソールの使用を許可するポリシー例を確認するには、「[IAM: ユーザーパスに基づく Policy Simulator のコンソールへのアクセス](#)」を参照してください。

1 つのタイプのエンティティのみのために Policy Simulator コンソールの使用を許可するポリシーを作成するには、次の手順を実行します。

コンソールユーザーがユーザーのポリシーをシミュレートできるようにするには

次のアクションをポリシーに含めます。

- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetUser
- iam:GetUserPolicy

- iam:ListAttachedUserPolicies
- iam:ListGroupsWithUser
- iam:ListGroupPolicies
- iam:ListUserPolicies
- iam:ListUsers

コンソールユーザーがユーザーグループのポリシーをシミュレートできるようにするには

次のアクションをポリシーに含めます。

- iam:GetGroup
- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:ListAttachedGroupPolicies
- iam:ListGroupPolicies
- iam:ListGroups

コンソールユーザーがロールのポリシーをシミュレートできるようにするには

次のアクションをポリシーに含めます。

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies
- iam:ListRolePolicies
- iam:ListRoles

リソースベースのポリシーをテストするには、リソースのポリシーを取得するためのアクセス権限がユーザーに必要です。

コンソールユーザーが Amazon S3 バケットのリソースベースのポリシーをテストできるようにするには

次のアクションをポリシーに含めます。

- s3:GetBucketPolicy

例えば、次のポリシーではこのアクションを使用して、コンソールユーザーに特定の Amazon S3 バケット内のリソースベースのポリシーをシミュレートすることを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetBucketPolicy",
      "Resource": "arn:aws:s3:::bucket-name/*"
    }
  ]
}
```

Policy Simulator API の使用に必要なアクセス許可

Policy Simulator API オペレーション [GetContextKeyForCustomPolicy](#) および [SimulateCustomPolicy](#) を使用すると、ユーザー、ユーザーグループ、またはロールにまだアタッチされていないポリシーをテストできます。このようなポリシーをテストするには、ポリシーを文字列として API に渡します。それらのポリシーはシミュレーションでのみ使用され、機密情報が公開されることはありません。API を使用して、AWS アカウントの IAM ユーザー、ユーザーグループ、またはロールにアタッチされているポリシーをテストすることもできます。そのためには、[GetContextKeyForPrincipalPolicy](#) および [SimulatePrincipalPolicy](#) を呼び出すアクセス許可をユーザーに付与する必要があります。

現在の AWS アカウント でアタッチされているポリシーとアタッチされていないポリシーに対して Policy Simulator API を使用することを許可するポリシーの例を表示するには、「[IAM: Policy Simulator API へのアクセス](#)」を参照してください。

1 つのタイプのポリシーのみのために Policy Simulator API の使用を許可するポリシーを作成するには、次の手順を実行します。

API ユーザーが API に直接文字列として渡されるポリシーをシミュレートできるようにするには

次のアクションをポリシーに含めます。

- iam:GetContextKeysForCustomPolicy
- iam:SimulateCustomPolicy

API ユーザーが IAM ユーザー、ユーザーグループ、ロール、またはリソースにアタッチされたポリシーをシミュレートできるようにするには

次のアクションをポリシーに含めます。

- iam:GetContextKeysForPrincipalPolicy
- iam:SimulatePrincipalPolicy

例えば、Bob という名前のユーザーに、Alice という名前のユーザーに割り当てられたポリシーをシミュレートするアクセス許可を付与するには、Bob にリソース `arn:aws:iam::777788889999:user/alice` へのアクセス権を付与します。

特定のパスがあるユーザーにのみ Policy Simulator API の使用を許可するポリシー例を確認するには、[IAM: ユーザーパスに基づいた Policy Simulator API へのアクセス](#)を参照してください。

## IAM Policy Simulator の使用方法 (コンソール)

デフォルトでは、ユーザーは、ユーザー、ユーザーグループ、またはロールにまだアタッチされていないポリシーをテストできます。そのためには、それらのポリシーを Policy Simulator コンソールに入力またはコピーします。それらのポリシーはシミュレーションでのみ使用され、機密情報が公開されることはありません。


ユーザー、ユーザーグループ、またはロールにアタッチされていないポリシーをテストするには (コンソール)

1. IAM ポリシーシミュレーターコンソール (<https://policysim.aws.amazon.com/>) を開きます。
2. ページ上部にある [Mode: (モード:)] メニューで [New Policy (新しいポリシー)] を選択します。
3. [Policy Sandbox (ポリシーサンドボックス)] で [新規ポリシーの作成] を選択します。
4. ポリシーを Policy Simulator に入力またはコピーし、次のステップで説明されているようにシミュレーションを実行します。

IAM Policy Simulator コンソールを使用するアクセス許可を取得したら、この Simulator により、IAM ユーザー、ユーザーグループ、ロール、またはリソースポリシーをテストできます。

ユーザー、ユーザーグループ、またはロールにアタッチされているポリシーをテストするには (コンソール)

1. IAM ポリシーシミュレーターコンソール (<https://policysim.aws.amazon.com/>) を開きます。

 Note

IAM ユーザーとして Policy Simulator にサインインするには、割り当てられたサインイン URL を使用して AWS Management Console にサインインします。その後、<https://policysim.aws.amazon.com/> に移動します。IAM ユーザーとしてサインインする詳細については、「[IAM ユーザーが AWS にサインインする方法](#)」を参照してください。

Policy Simulator が [Existing Policies] (既存のポリシー) モードで開き、アカウントの IAM ユーザーを [Users, Groups, and Roles] (ユーザー、グループ、およびロール) の下に一覧表示します。

2. 目的のタスクに該当するオプションを選択します。

これをテストするには:	処理:
ユーザーにアタッチされているポリシー	ユーザー、グループ、ロールリストでユーザーを選択します。その後、ユーザーを選択します。
ユーザーグループにアタッチされているポリシー	ユーザー、グループ、ロールリストでグループを選択します。次に、ユーザーグループを選択します。
ロールにアタッチされているポリシー	ユーザー、グループ、ロールリストでロールを選択します。その後、ロールを選択します。
リソースにアタッチされているポリシー	「 <a href="#">Step 9</a> 」を参照してください。

これをテストするには:

処理:

ユーザー、ユーザーグループ、またはロールのカスタムポリシー

[新しいポリシーの作成] を選択します。新しい [ポリシー] ペインで、ポリシーを入力または貼り付け、[Apply (適用)] を選択します。

### ヒント

ユーザーグループにアタッチされたポリシーをテストするには、IAM Policy Simulator を [IAM コンソール](#) から直接起動します。ナビゲーションペインで、[Groups] を選択します。ポリシーをテストするグループの名前を選択し、次に [Permissions (アクセス許可)] タブを選択します。シミュレートを選択します。

ユーザーにアタッチされているカスタマー管理ポリシーをテストするには、ナビゲーションペインで [ユーザー] を選択します。ポリシーをテストするユーザーの名前を選択します。[Permissions (アクセス許可)] タブを選択し、テストするポリシーを展開します。右端にある [ポリシーのシミュレート] を選択します。新しいウィンドウで IAM Policy Simulator が開き、選択したポリシーが [ポリシー] ペインに表示されます。

3. (オプション) アカウントが [AWS Organizations](#) の組織のメンバーである場合は、AWS Organizations SCP の横にあるチェックボックスをオンにして、シミュレートされた評価に SCP を含めます。SCP は、組織または組織単位 (OU) のアクセス許可の上限を指定する JSON ポリシーです。SCP はメンバーアカウントのエンティティに対するアクセス許可を制限します。SCP によってサービスまたはアクションがブロックされる場合、そのアカウントには、そのサービスにアクセスできるエンティティ、またはそのアクションを実行できるエンティティがありません。管理者が IAM またはリソースポリシーを使用してそのサービスまたはアクションへのアクセス権限を明示的に付与したとしても同様です。

アカウントが組織のメンバーでない場合、チェックボックスは表示されません。

4. (オプション) IAM エンティティ (ユーザーまたはロール) の [アクセス許可境界](#) として設定されているポリシーをテストできますが、ユーザーグループに対してはテストできません。現在、エンティティにアクセス許可の境界ポリシーが設定されている場合、そのポリシーが [ポリシー] ペインに表示されます。1つのエンティティに設定できるアクセス許可の境界は1つのみです。別のアクセス許可の境界をテストするために、カスタムのアクセス許可の境界を作成できます。そのためには、[新しいポリシーの作成] を選択します。新しい [ポリシー] ペインが開きます。メ



メニューで、[Custom IAM Permissions Boundary Policy (カスタムの IAM アクセス許可の境界ポリシー)] を選択します。新しいポリシーの名前を入力し、下の領域にポリシーを入力またはコピーします。[Apply (適用)] を選択して、ポリシーを保存します。次に、[Back (戻る)] を選択して、元の [ポリシー] ペインに戻ります。その後、シミュレーションに使用するアクセス許可の境界の横にあるチェックボックスをオンにします。

5. (オプション) ユーザー、ユーザーグループ、またはロールにアタッチされているポリシーのサブセットのみをテストできます。そのためには、[ポリシー] ペインで、除外する各ポリシーの横にあるチェックボックスをオフにします。
6. Policy Simulator で、[Select service (サービスの選択)] を選択してから、テストするサービスを選択します。その後、[アクションの選択] を選択し、テストするアクションを 1 つ以上選択します。メニューでは、サービスごとに使用できる選択肢しか表示されませんが、[Action Settings and Results (アクションの設定と結果)] では選択したすべてのサービスとアクションが表示されます。
7. (オプション) [Step 2](#) および [Step 5](#) で選択したポリシーに、[AWS グローバル条件キー](#)を持つ条件が含まれている場合、それらのキーの値を入力します。これを行うには、[グローバル設定] セクションを展開して、そこに表示されるキー名の値を入力します。

**⚠ Warning**

条件キーの値を空にすると、そのキーはシミュレーション時に無視されます。これによってエラーが発生し、シミュレーションの実行が失敗する場合があります。また、シミュレーションが実行されても、結果が信頼できない場合があります。シミュレーションが、条件キーや変数の値を含む実際の状況と一致しない場合もあります。

8. (オプション) 選択した各アクションは [Action Settings and Results (アクションの設定と結果)] リストに表示され、実施にシミュレーションを実行するまで [Not simulated (未シミュレーション)] が [アクセス許可] 列に表示されます。シミュレーションを実行する前に、リソースで各アクションを設定できます。特定のシナリオ用に個別のアクションを設定するには、矢印を選択してアクションの行を展開します。アクションがリソースレベルのアクセス許可をサポートしている場合、アクセスをテストする特定のリソースの [Amazon リソースネーム \(ARN\)](#) を入力できます。デフォルトでは、各リソースはワイルドカード (\*) に設定されています。任意の [条件コンテキストキー](#) の値を指定することもできます。前述したように、値が空のキーは無視されるため、シミュレーションの失敗や信頼性の低い結果が生じる可能性があります。
  - a. アクション名の横にある矢印を選択して各行を拡張し、使用するシナリオでアクションを正確にシミュレートするために必要な追加の情報を設定します。アクションがリソースレベルのアクセス許可を必要とする場合、アクセスをシミュレートする特定のリソースの [Amazon](#)



[リソースネーム \(ARN\)](#) を入力できます。デフォルトでは、各リソースはワイルドカード (\*) に設定されています。

- b. アクションがリソースレベルのアクセス許可をサポートするものの、それを必要としない場合、[Add Resource (リソースの追加)] を選択して、シミュレーションに追加するリソースタイプを選択できます。
- c. 選択したポリシーのいずれかに、このアクションのサービスのコンテキストキーを参照する Condition 要素が含まれる場合、アクションの下にそのキー名が表示されます。そのアクションのシミュレーションで、指定したリソースに対して使用する値を指定できます。

### 異なるグループのリソースタイプが必要なアクション

アクションによって、異なる状況下で異なるリソースタイプが必要です。リソースタイプの各グループにはシナリオが関連付けられています。そのいずれかがシミュレーション内容に適合しているのであれば、それを選択すると、シミュレーターが対象のシナリオに適したリソースタイプを要求します。次の一覧では、サポートされるそれぞれのシナリオオプションと、シミュレーションを実行するために定義する必要のあるリソースを示しています。

次の各 Amazon EC2 シナリオでは、instance、image、および security-group リソースを指定する必要があります。シナリオに EBS ボリュームが含まれる場合、その volume をリソースとして指定する必要があります。Amazon EC2 シナリオに仮想プライベートクラウド (VPC) が含まれる場合、network-interface リソースを指定する必要があります。IP サブネットが含まれる場合、subnet リソースを指定する必要があります。Amazon EC2 シナリオオプションの詳細については、[Amazon EC2 ユーザーガイド](#)の「サポートされているプラットフォーム」を参照してください。

- EC2-VPC-InstanceStore

instance, image, security-group, network-interface

- EC2-VPC-InstanceStore-Subnet

instance, image, security-group, network-interface, subnet

- EC2-VPC-EBS

instance, image, security-group, network-interface, volume

- EC2-VPC-EBS-Subnet

instance, image, security-group, network-interface, subnet, volume

9. (任意) リソースベースのポリシーをシミュレーションに含める場合、まずそのリソースでシミュレートするアクションを [Step 6](#) で選択する必要があります。選択したアクションの行を展開し、シミュレートするポリシーのリソースの ARN を入力します。その後、[ARN] テキストボックスの横にある [Include Resource Policy] を選択します。IAM Policy Simulator では現在、次のサービスのリソースベースポリシーをサポートしています。Amazon S3 (リソースベースポリシーのみ。ACL は現在サポート対象外)、Amazon SQS、Amazon SNS、およびロック解除された S3 Glacier ボールト (ロックされたボールトは現在サポート対象外)。
10. 右上の [Run Simulation (シミュレーションの実行)] を選択します。

[アクションの設定と結果]の各行の[アクセス許可]列には、指定したリソースでのそのアクションのシミュレーション結果が表示されます。

11. ポリシーのどのステートメントがアクションを明示的に許可または拒否したかを調べるには、[Permissions] 列の **[N matching statement(s)]** リンクを選択して行を展開します。次に、[Show statement (ステートメントの表示)] リンクを選択します。[ポリシー] ペインに、関連するポリシーが表示され、シミュレーションの結果に影響を与えたステートメントが強調表示されます。

#### Note

アクションが暗黙的に拒否 (つまり、明示的に許可されていないためのみアクションが拒否された場合) リストおよびステートメントを表示オプションは表示されません。

## IAM Policy Simulator コンソールメッセージのトラブルシューティング

以下の表では、IAM Policy Simulator の使用時に表示される可能性のある通知および警告メッセージを示しています。それらのメッセージが表示されたときの問題の解決手順についても説明しています。

Message	解決手順
This policy has been edited. Changes will not be saved to your account.	<p>対処は必要ありません。</p> <p>これは通知メッセージです。IAM ポリシーシミュレーターで既存のポリシーを編集した場合、その変更は AWS アカウント に影響を与えません。Policy Simulator では、テスト目的のみポリシーに変更を加えられます。</p>

Message	解決手順
<p>リソースポリシーを取得できない。原因: #####</p> <p>One or more policies require values in the simulation settings. The simulation might fail without these values.</p>	<p>Policy Simulator は、リクエストされたリソースポリシーにアクセスすることはできません。指定されたリソース ARN が正しいこと、またシミュレーションを実行しているユーザーにリソースポリシーを見る権限があることを確認します。</p> <p>このメッセージが表示されるのは、テストするポリシーに条件キーや変数が含まれているが、これらのキーや変数の値を [Simulation Settings (シミュレーションの設定)] で指定しなかった場合です。</p> <p>このメッセージが表示されないようにするには、[Simulation Settings (シミュレーション設定)] を選択し、各条件キーまたは変数に値を入力します。</p>
<p>You have changed policies. These results are no longer valid.</p>	<p>このメッセージが表示されるのは、[結果] ペインに結果が表示されている間に、選択したポリシーを変更した場合です。[結果] ペインに表示されている結果は動的に更新されません。</p> <p>このメッセージが表示されないようにするには、[Run Simulation (シミュレーションの実行)] を再度選択します。[ポリシー] ペインで行った変更に基づいて、新しいシミュレーションの結果が表示されます。</p>

Message	解決手順
<p>The resource you typed for this simulation does not match this service.</p>	<p>このメッセージが表示されるのは、現在のシミュレーション用に選択したサービスと一致しない Amazon リソースネーム (ARN) を [Simulation Settings (シミュレーションの設定)] ペインで入力した場合です。例えば、Amazon DynamoDB リソースの ARN を指定し、シミュレートするサービスに Amazon Redshift を選択した場合などです。</p> <p>このメッセージが表示されないようにするには、以下のいずれかの操作を行います。</p> <ul style="list-style-type: none"><li>• [Simulation Settings (シミュレーションの設定)] ペインのボックスから ARN を削除する。</li><li>• [Simulation Settings (シミュレーションの設定)] で指定した ARN に一致するサービスを選択する。</li></ul>
<p>このアクションは、Amazon S3 ACL や S3 Glacier ポールトロックポリシーなどのリソースベースポリシーに加えて、特別なアクセスコントロールメカニズムをサポートするサービスに属します。Policy Simulator はこれらのメカニズムをサポートしないため、結果は本番環境と異なる場合があります。</p>	<p>対処は必要ありません。</p> <p>これは通知メッセージです。現在のバージョンの Policy Simulator は、ユーザーおよびユーザーグループにアタッチされたポリシーと、Amazon S3、Amazon SQS、Amazon SNS、S3 Glacier 用のリソースベースのポリシーを評価できます。Policy Simulator は、他の AWS サービスでサポートされるすべてのアクセスコントロールメカニズムをサポートするわけではありません。</p>

Message	解決手順
DynamoDB FGAC is currently not supported.	<p>対処は必要ありません。</p> <p>この情報メッセージは、きめ細かなアクセスコントロールに関するものです。きめ細かなアクセスコントロールは、IAM ポリシー条件を使用して、DynamoDB テーブルおよびインデックス内の個々のデータ項目および属性にだれがアクセスできるかを決定する機能です。また、これらのテーブルとインデックスに対して実行できるアクションも参照します。IAM Policy Simulator の現在のバージョンでは、このタイプのポリシー条件はサポートされていません。DynamoDB のきめ細かなアクセスコントロールの詳細については、「<a href="#">DynamoDB のきめ細かなアクセスコントロール</a>」を参照してください。</p>
ポリシー構文に準拠していないポリシーがあります。ポリシー検証を使用して、ポリシーの推奨される更新を確認できます。	IAM ポリシーの文法に準拠しないポリシーがある場合、このメッセージがポリシーのリストの先頭に表示されます。これらのポリシーをシミュレートするには、 <a href="#">IAM ポリシーの検証</a> でポリシー検証オプションを確認して、これらのポリシーを特定して修正します。
This policy must be updated to comply with the latest policy syntax rules.	このメッセージが表示されるのは、IAM ポリシーの文法に準拠しないポリシーがある場合です。これらのポリシーをシミュレートするには、 <a href="#">IAM ポリシーの検証</a> でポリシー検証オプションを確認して、これらのポリシーを特定して修正します。

## AWS CLI IAM Policy Simulator の使用 ( および AWS API )

通常、Policy Simulator コマンドでは、次の 2 つのことを実行するために、API オペレーションの呼び出しが必要となります。

1. ポリシーを評価し、ポリシーが参照するコンテキストキーのリストを返します。次のステップで、参照しているコンテキストキーの値を指定できるように、それらを確認しておく必要があります。
2. ポリシーをシミュレートし、シミュレーションで使用されるアクション、リソース、コンテキストキーの一覧を提供します。

セキュリティ上の理由から、API オペレーションは以下の 2 つのグループに分かれています。

- API に文字列として直接渡されるポリシーのみをシミュレートする API オペレーション。このセットには、[GetContextKeysForCustomPolicy](#) と [SimulateCustomPolicy](#) が含まれます。
- 特定の IAM ユーザー、ユーザーグループ、ロール、またはリソースにアタッチされたポリシーをシミュレートする API オペレーション。これらの API オペレーションは他の IAM エンティティに割り当てられたアクセス許可の詳細を表示できるため、これらの API オペレーションへのアクセスを制限することを検討してください。このセットには、[GetContextKeysForPrincipalPolicy](#) と [SimulatePrincipalPolicy](#) が含まれます。API オペレーションへのアクセス制限の詳細については、「[ポリシーの例: AWS Identity and Access Management \(IAM\)](#)」を参照してください。

いずれの場合でも、API オペレーションはアクションやリソースのリストに対する 1 つ以上のポリシーの影響をシミュレートします。各アクションは各リソースと組み合わせられており、シミュレーションによって、ポリシーがリソースのそのアクションを許可するか、または拒否するかを特定できます。また、ポリシーが参照するコンテキストキーの値を提供することもできます。ポリシーが参照するコンテキストキーのリストを取得するには、まず [GetContextKeysForCustomPolicy](#) または [GetContextKeysForPrincipalPolicy](#) を呼び出します。コンテキストキーの値を指定しない場合、シミュレーションは引き続き実行されます。ただし、Policy Simulator が、そのコンテキストキーを評価に含めることができないため、信頼性の低い結果が得られることがあります。

コンテキストキーのリストを取得するには (AWS CLI、AWS API)

以下のコマンドでは、ポリシーのリストを評価し、ポリシーで使用されているコンテキストキーのリストを返します。

- AWS CLI: [aws iam get-context-keys-for-custom-policy](#) および [aws iam get-context-keys-for-principal-policy](#)
- AWS API: [GetContextKeysForCustomPolicy](#) および [GetContextKeysForPrincipalPolicy](#)

IAM ポリシーをシミュレートするには (AWS CLI、AWS API)

以下のコマンドでは、IAM ポリシーをシミュレートしてユーザーの有効なアクセス許可を確認します。

- AWS CLI: [aws iam simulate-custom-policy](#) および [aws iam simulate-principal-policy](#)
- AWS API: [SimulateCustomPolicy](#) および [SimulatePrincipalPolicy](#)

## IAM ID のアクセス許可の追加および削除

ID (ユーザー、ユーザーグループ、またはロール) のアクセス許可を定義するにはポリシーを使用します。アクセス許可を追加および削除するには、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用して、ID の IAM ポリシーをアタッチおよびデタッチします。また、ポリシーを使用して、同じ方法を使用しているエンティティ (ユーザーまたはロール) のみに [許可の境界](#) を設定することもできます。アクセス許可の境界は、エンティティが持つことができる最大のアクセス許可を制御する AWS のアドバンスド機能です。

トピック

- [用語](#)
- [ID アクティビティの表示](#)
- [IAM ID アクセス許可の追加 \(コンソール\)](#)
- [IAM ID アクセス許可の削除 \(コンソール\)](#)
- [IAM ポリシーの追加 \(AWS CLI\)](#)
- [IAM ポリシーの削除 \(AWS CLI\)](#)
- [IAM ポリシーの追加 \(AWS API\)](#)
- [IAM ポリシーの削除 \(AWS API\)](#)

## 用語

アクセス許可ポリシーをアイデンティティ (ユーザー、ユーザーグループ、およびロール) に関連付ける場合、管理ポリシーとインラインポリシーのどちらを使用するかで、用語や手順が異なることがあります。



- **アタッチ** – 管理ポリシーで使用します。管理ポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチします。ポリシーをアタッチすると、そのポリシー内のアクセス許可が ID に適用されます。
- **デタッチ** – 管理ポリシーで使用します。管理ポリシーを IAM アイデンティティ (ユーザー、ユーザーグループ、またはロール) からデタッチします。ポリシーをデタッチすると、そのアクセス許可がアイデンティティから削除されます。
- **埋め込み** – インラインポリシーで使用します。インラインポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) に埋め込みます。ポリシーを埋め込むと、そのポリシー内のアクセス許可が ID に適用されます。インラインポリシーはアイデンティティに保存されるため、結果は似ていますが、アタッチされるのではなく埋め込まれます。

**Note**

[サービスにリンクされたロール](#)のインラインポリシーは、ロールに依存するサービスにのみ組み込むことができます。サービスでこの機能がサポートされているかどうかについては、そのサービスの [AWS ドキュメント](#) を参照してください。

- **削除** – インラインポリシーで使用します。インラインポリシーを IAM アイデンティティ (ユーザー、ユーザーグループ、またはロール) から削除します。ポリシーを削除すると、そのアクセス許可がアイデンティティから削除されます。

**Note**

[サービスにリンクされたロール](#)のインラインポリシーを削除できるのは、そのロールに依存するサービスに限ります。サービスでこの機能がサポートされているかどうかについては、そのサービスの [AWS ドキュメント](#) を参照してください。

これらのアクションはいずれも、コンソール、AWS CLI、または AWS API を使用して実行できます。

### 詳細情報

- 管理ポリシーとインラインポリシーの違いの詳細については、「[管理ポリシーとインラインポリシー](#)」を参照してください。
- アクセス許可の境界の詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。



- IAM ポリシーの一般情報については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。
- IAM ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。
- AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

## ID アクティビティの表示

アイデンティティ (ユーザー、ユーザーグループ、ロール) のアクセス許可を変更する前に、サービスレベルの直近アクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## IAM ID アクセス許可の追加 (コンソール)

アクセス許可をアイデンティティ (ユーザー、ユーザーグループ、またはロール) に追加するには、AWS Management Console を使用します。そのためには、アクセス許可を制御する管理ポリシーをアタッチするか、[アクセス許可の境界](#)としてポリシーを指定します。また、インラインポリシーを埋め込むこともできます。

管理ポリシーをアイデンティティのアクセス許可ポリシーとして使用するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーのリストで、アタッチするポリシーの名前の横にあるラジオボックスをオンにします。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [Actions (アクション)] を選択し、[Attach (アタッチ)] を選択します。
5. ポリシーを添付する ID を 1 つ以上選択します。検索ボックスを使用して、プリンシパルエンティティのリストをフィルタリングできます。ID を選択したら、[ポリシーのアタッチ] を選択します。

## 管理ポリシーを使用してアクセス許可の境界を設定するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、設定するポリシーの名前を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [ポリシー詳細] ページで、[アタッチされたエンティティ] タブを選択し、必要に応じて [アクセス許可の境界としてアタッチ] セクションを開き、[このポリシーをアクセス許可の境界として設定] を選択します。
5. ポリシーをアクセス許可の境界として使用する対象のユーザーまたはロールを 1 つ以上選択します。検索ボックスを使用して、プリンシパルエンティティのリストをフィルタリングできます。プリンシパルを選択したら、[アクセス許可の境界を設定] を選択します。

## ユーザーまたはロールのインラインポリシーを埋め込むには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM ナビゲーションペインで、[Users] (ユーザー) または [Roles] (ロール) を選択します。
3. 一覧で、ポリシーを埋め込むユーザーまたはロールの名前を選択します。
4. [アクセス許可] タブを選択します。
5. [アクセス許可を追加]、[インラインポリシーを作成] の順に選択します。

### Note

IAM の「[サービスにリンクされたロール](#)」にインラインポリシーを埋め込むことはできません。リンクされたサービスは、ロールの許可を変更できるかどうかを定義するため、サービスコンソール、API、または AWS CLI からポリシーを追加できる場合があります。サービスにリンクされたロールのドキュメントをサービスで表示するには、「[IAM と連携する AWS のサービス](#)」を参照の上、お使いのサービスの [Service-Linked Role] 列で [Yes] を選択します。

6. 以下の方法のいずれかを選択してポリシーの作成に必要な手順を表示します。

- [既存の管理ポリシーのインポート](#) – アカウント内で管理ポリシーをインポートし、ポリシーを編集して特定の要件に合わせてカスタマイズすることができます。管理ポリシーは、AWS 管理ポリシーまたは以前に作成したカスタマー管理ポリシーにすることができます。
  - [ビジュアルエディタでのポリシーの作成](#) – ビジュアルエディタで最初から新しいポリシーを構築することができます。ビジュアルエディタを使用する場合は、JSON 構文を理解する必要はありません。
  - [JSON エディターを使用したポリシーの作成](#) – [JSON] エディタオプションで、JSON 構文を使用してポリシーを作成することができます。新しい JSON ポリシードキュメントを入力するか、[ポリシー例](#)を貼り付けることができます。
7. インラインポリシーを作成した後は、自動的にユーザーまたはロールに埋め込まれます。

ユーザーグループのインラインポリシーを埋め込むには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ] を選択します。
3. 一覧で、ポリシーを埋め込むユーザーグループの名前を選択します。
4. [アクセス許可] タブで、[アクセス許可の追加] を選択してから、インラインポリシーの作成。
5. 以下のいずれかを実行します。
  - [ビジュアル] オプションを選択して、ポリシーを作成します。詳細については、「[ビジュアルエディタでのポリシーの作成](#)」を参照してください。
  - [JSON] オプションを選択して、ポリシーを作成します。詳細については、「[JSON エディターを使用したポリシーの作成](#)」を参照してください。
6. ポリシーが完成したら、[Create policy] (ポリシーの作成) を選択します。

1 つ以上のエンティティのアクセス許可の境界を変更するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、設定するポリシーの名前を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。

4. [ポリシー詳細] ページで、[アタッチされたエンティティ] タブを選択し、必要に応じて [アクセス許可の境界としてアタッチ] セクションを開きます。境界を変更するユーザーまたはロールの横にあるチェックボックスを選択し、[変更] を選択します。
5. アクセス許可の境界として使用する新しいポリシーを選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。ポリシーを選択したら、[アクセス許可の境界を設定] を選択します。

## IAM ID アクセス許可の削除 (コンソール)

アクセス許可をアイデンティティ (ユーザー、ユーザーグループ、またはロール) から削除するには、AWS Management Console を使用します。そのためには、アクセス許可を制御する管理ポリシーをデタッチするか、[アクセス許可の境界](#)として指定されているポリシーを削除します。また、インラインポリシーを削除することもできます。

アクセス許可ポリシーとして使用されている管理ポリシーをデタッチするには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーのリストで、デタッチするポリシーの名前の横にあるラジオボックスを選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [アクション] を選択して、[削除] を選択します。
5. ポリシーをデタッチする ID を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。ID を選択したら、[ポリシーのデタッチ] を選択します。

アクセス許可の境界を削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、設定するポリシーの名前を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [ポリシーの概要] ページで、[アタッチされたエンティティ] タブを選択し、必要に応じて [アクセス許可の境界としてアタッチ] セクションを開き、アクセス許可の境界を削除するエンティティを選択します。次に、[境界を削除] を選択します。

## 5. 境界を削除することを確認し、[境界を削除] を選択します。

インラインポリシーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ]、[ユーザー]、または [ロール] を選択します。
3. リストで、削除するポリシーを持つユーザーグループ、ユーザー、またはロールの名前を選択します。
4. [アクセス許可] タブを選択します。
5. ポリシーの横にあるチェックボックスを選択し、[削除] を選択します。
6. 確認ボックスで、[削除] を選択します。

## IAM ポリシーの追加 ( AWS CLI)

アクセス許可をアイデンティティ (ユーザー、ユーザーグループ、またはロール) に追加するには、AWS CLI を使用します。そのためには、アクセス許可を制御する管理ポリシーをアタッチするか、[アクセス許可の境界](#)としてポリシーを指定します。また、インラインポリシーを埋め込むこともできます。

管理ポリシーをエンティティのアクセス許可ポリシーとして使用するには (AWS CLI)

1. (オプション) 管理ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [aws iam list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [get-policy](#)
2. 管理ポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチするには、以下のいずれかのコマンドを使用します。
  - [aws iam attach-user-policy](#)
  - [aws iam attach-group-policy](#)
  - [aws iam attach-role-policy](#)

管理ポリシーを使用してアクセス許可の境界を設定するには (AWS CLI)

1. (オプション) 管理ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [aws iam list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [aws iam get-policy](#)
2. 管理ポリシーを使用してエンティティ (ユーザーまたはロール) のアクセス許可の境界を設定するには、以下のいずれかのコマンドを使用します。
  - [aws iam put-user-permissions-boundary](#)
  - [aws iam put-role-permissions-boundary](#)

インラインポリシーを埋め込むには (AWS CLI)

インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) に埋め込むには、以下のいずれかのコマンドを使用します。

- [aws iam put-user-policy](#)
- [aws iam put-group-policy](#)
- [aws iam put-role-policy](#)

IAM ポリシーの削除 (AWS CLI)

AWS CLI を使用して、アクセス許可を制御する管理ポリシーをデタッチするか、[アクセス許可の境界](#)として指定されているポリシーを削除します。また、インラインポリシーを削除することもできます。

アクセス許可ポリシーとして使用されている管理ポリシーをデタッチするには (AWS CLI)

1. (オプション) ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [aws iam list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [aws iam get-policy](#)
2. (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のコマンドを実行します。
  - 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示する場合:

- [aws iam list-entities-for-policy](#)
- アイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチされている管理ポリシーを一覧表示するには、以下のいずれかのコマンドを使用します。
  - [aws iam list-attached-user-policies](#)
  - [aws iam list-attached-group-policies](#)
  - [aws iam list-attached-role-policies](#)
- 3. 管理ポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) からデタッチするには、以下のいずれかのコマンドを使用します。
  - [aws iam detach-user-policy](#)
  - [aws iam detach-group-policy](#)
  - [aws iam detach-role-policy](#)

アクセス許可の境界を削除するには (AWS CLI)

1. (オプション) アクセス許可の境界を設定するために現在使用されている管理ポリシーを確認するには、以下のコマンドを実行します。
  - [aws iam get-user](#)
  - [aws iam get-role](#)
2. (オプション) 管理ポリシーがアクセス許可の境界として使用されているユーザーまたはロールを確認するには、次のコマンドを実行します。
  - [aws iam list-entities-for-policy](#)
3. (オプション) 管理ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [aws iam list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [aws iam get-policy](#)
4. ユーザーまたはロールからアクセス許可の境界を削除するには、以下のいずれかのコマンドを使用します。
  - [aws iam delete-user-permissions-boundary](#)
  - [aws iam delete-role-permissions-boundary](#)



## インラインポリシーを削除するには (AWS CLI)

1. (オプション) アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされたすべてのインラインポリシーを一覧表示するには、以下のいずれかのコマンドを使用します。
  - [aws iam list-user-policies](#)
  - [aws iam list-group-policies](#)
  - [aws iam list-role-policies](#)
2. (オプション) アイデンティティ (ユーザー、ユーザーグループ、またはロール) に埋め込まれたインラインポリシードキュメントを取得するには、以下のいずれかのコマンドを使用します。
  - [aws iam get-user-policy](#)
  - [aws iam get-group-policy](#)
  - [aws iam get-role-policy](#)
3. インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) から削除するには、以下のいずれかのコマンドを使用します。
  - [aws iam delete-user-policy](#)
  - [aws iam delete-group-policy](#)
  - [aws iam delete-role-policy](#)

## IAM ポリシーの追加 (AWS API)

AWS API を使用して、アクセス許可を制御する管理ポリシーをアタッチするか、[アクセス許可の境界](#)としてポリシーを指定します。また、インラインポリシーを埋め込むこともできます。

管理ポリシーをエンティティのアクセス許可ポリシーとして使用するには (AWS API)

1. (オプション) ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
2. 管理ポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチするには、以下のいずれかのオペレーションを呼び出します。
  - [AttachUserPolicy](#)
  - [AttachGroupPolicy](#)



- [AttachRolePolicy](#)

管理ポリシーを使用してアクセス許可の境界を設定するには (AWS API)

1. (オプション) 管理ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
2. 管理ポリシーを使用してエンティティ (ユーザーまたはロール) のアクセス許可の境界を設定するには、以下のいずれかのオペレーションを呼び出します。
  - [PutUserPermissionsBoundary](#)
  - [PutRolePermissionsBoundary](#)

インラインポリシーを埋め込むには (AWS API)

インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) に埋め込むには、以下のいずれかのオペレーションを呼び出します。

- [PutUserPolicy](#)
- [PutGroupPolicy](#)
- [PutRolePolicy](#)

IAM ポリシーの削除 (AWS API)

AWS API を使用して、アクセス許可を制御する管理ポリシーをデタッチするか、[アクセス許可の境界](#)として指定されているポリシーを削除します。また、インラインポリシーを削除することもできます。

アクセス許可ポリシーとして使用されている管理ポリシーをデタッチするには (AWS API)

1. (オプション) ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
2. (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のオペレーションを呼び出します。

- 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示する場合:
    - [ListEntitiesForPolicy](#)
  - アイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチされている管理ポリシーを一覧表示するには、以下のいずれかのオペレーションを呼び出します。
    - [ListAttachedUserPolicies](#)
    - [ListAttachedGroupPolicies](#)
    - [ListAttachedRolePolicies](#)
3. 管理ポリシーをアイデンティティ (ユーザー、ユーザーグループ、またはロール) からデタッチするには、以下のいずれかのオペレーションを呼び出します。
- [DetachUserPolicy](#)
  - [DetachGroupPolicy](#)
  - [DetachRolePolicy](#)

アクセス許可の境界を削除するには (AWS API)

1. (オプション) アクセス許可の境界を設定するために現在使用されている管理ポリシーを確認するには、以下のオペレーションを呼び出します。
  - [GetUser](#)
  - [GetRole](#)
2. (オプション) 管理ポリシーがアクセス許可の境界として使用されているユーザーまたはロールを確認するには、次のオペレーションを呼び出します。
  - [ListEntitiesForPolicy](#)
3. (オプション) 管理ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
4. ユーザーまたはロールからアクセス許可の境界を削除するには、以下のいずれかのオペレーションを呼び出します。
  - [DeleteUserPermissionsBoundary](#)
  - [DeleteRolePermissionsBoundary](#)

## インラインポリシーを削除するには (AWS API)

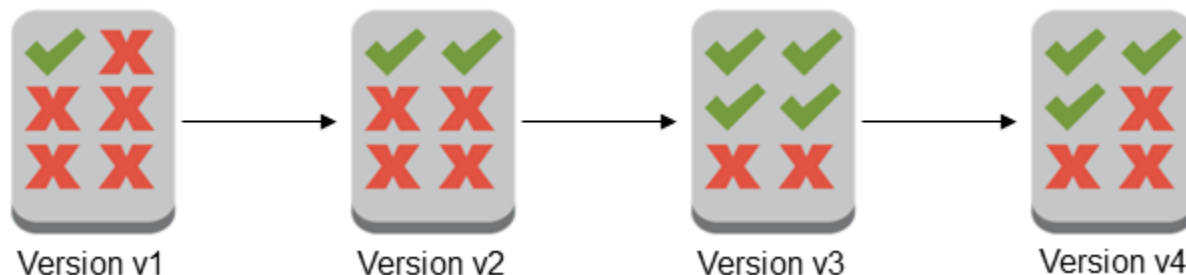
1. (オプション) アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされたすべてのインラインポリシーを一覧表示するには、以下のいずれかのオペレーションを呼び出します。
  - [ListUserPolicies](#)
  - [ListGroupPolicies](#)
  - [ListRolePolicies](#)
2. (オプション) アイデンティティ (ユーザー、ユーザーグループ、またはロール) に埋め込まれたインラインポリシードキュメントを取得するには、以下のいずれかのオペレーションを呼び出します。
  - [GetUserPolicy](#)
  - [GetGroupPolicy](#)
  - [GetRolePolicy](#)
3. インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) から削除するには、以下のいずれかのオペレーションを呼び出します。
  - [DeleteUserPolicy](#)
  - [DeleteGroupPolicy](#)
  - [DeleteRolePolicy](#)

## IAM ポリシーのバージョニング

お客様が IAM カスタマー管理ポリシーを変更した場合、または AWS が AWS 管理ポリシーを変更した場合、変更されたポリシーで既存のポリシーは上書きされません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。IAM は、最大 5 つのバージョンのカスタマー管理ポリシーを保存します。IAM はインラインポリシーのバージョニングをサポートしていません。

次の図は、カスタマー管理ポリシーのバージョニングを示しています。この例では、バージョン 1 ~ 4 が保存されます。IAM には、最大 5 つのマネージドポリシーバージョンを保存できます。6 番目の保存バージョンを作成することになるポリシーを編集する場合、以後保存しないこととする古いバージョンを選択できます。いつでも他の 4 つの保存済みバージョンのいずれかに戻すことができます。

## Multiple versions of a single managed policy



ポリシーのバージョンは、Version ポリシーの要素とは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。Version ポリシー要素の詳細については、「[IAM JSON ポリシー要素Version](#)」を参照してください。

バージョンを使用して、管理ポリシーの変更を追跡できます。たとえば、管理ポリシーに変更を加えた後で、その変更が意図しない結果をもたらしたことに気付く場合があります。この場合、以前のバージョンをデフォルトのバージョンとして設定することで、管理ポリシーを以前のバージョンにロールバックできます。

以下のトピックでは、管理ポリシーにバージョニングを使用する方法を説明しています。

### トピック

- [ポリシーのデフォルトバージョンの設定におけるアクセス権限](#)
- [カスタマー管理ポリシーのデフォルトバージョンの設定](#)
- [バージョンを使用して変更をロールバックする](#)
- [バージョンの制限](#)

### ポリシーのデフォルトバージョンの設定におけるアクセス権限

ポリシーのデフォルトバージョンを設定するために必要なアクセス権限は、このタスクの AWS API オペレーションに対応します。ポリシーのデフォルトバージョンを設定するには、CreatePolicyVersion または SetDefaultPolicyVersion API オペレーションを使用できます。ユーザーが既存のポリシーにデフォルトバージョンのポリシーを設定できるようにするには、iam:CreatePolicyVersion アクションあるいは iam:SetDefaultPolicyVersion アクションのいずれかへのアクセスを許可できます。iam:CreatePolicyVersion アクションは、ポリシーの新しいバージョンを作成して、デフォルトとしてこのバージョンを設定できるようにします。iam:SetDefaultPolicyVersion アクションは、既存のすべてのポリシーをデフォルトとして設定できるようにします。

**⚠ Important**

ユーザーポリシーで `iam:SetDefaultPolicyVersion` アクションを拒否することは、ユーザーが新しいポリシーバージョンを作成して、それをデフォルトとして設定することを阻止しません。

ユーザーが既存のカスタマー管理ポリシーの変更にアクセスすることを拒否するには、次のポリシーを使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "iam:CreatePolicyVersion",
        "iam:SetDefaultPolicyVersion"
      ],
      "Resource": "arn:aws:iam::*:policy/POLICY-NAME"
    }
  ]
}
```

## カスタマー管理ポリシーのデフォルトバージョンの設定

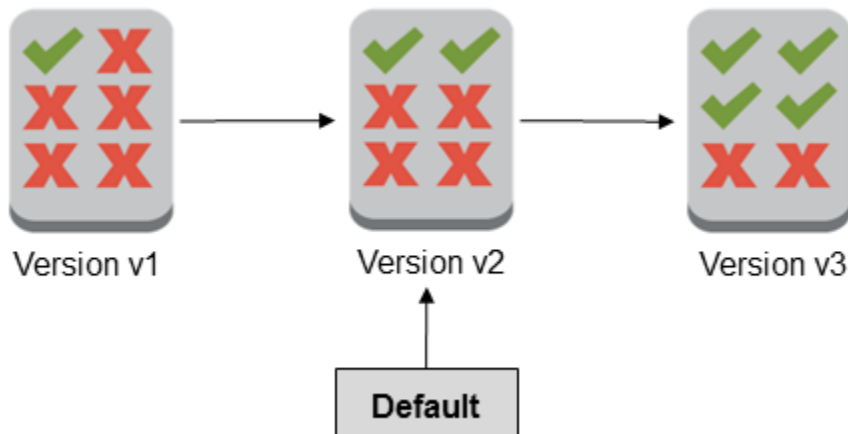
管理ポリシーのバージョンのうちの 1 つはデフォルトバージョンに設定されています。ポリシーのデフォルトバージョンは動作バージョン、つまり管理ポリシーをアタッチするすべてのプリンシパルエンティティ (ユーザー、ユーザーグループ、ロール) に対して有効なバージョンです。

カスタマー管理ポリシーを作成すると、ポリシーは v1 として識別される単一バージョンから始まります。管理ポリシーのバージョンが 1 つのみの場合は、そのバージョンが自動的にデフォルトとして設定されます。カスタマー管理ポリシーに複数のバージョンがある場合は、デフォルトとして設定するバージョンを選択します。AWS 管理ポリシーのデフォルトバージョンは AWS によって設定されます。次の図にその概念を示します。

## Managed policy with one version



## Managed policy with multiple versions



カスタマー管理ポリシーのデフォルトバージョンを設定すると、そのバージョンは、ポリシーがアタッチされているすべての IAM アイデンティティ (ユーザー、ユーザーグループ、ロール) に対してそのバージョンを適用できます。AWS 管理ポリシーまたはインラインポリシーのデフォルトバージョンを設定することはできません。

カスタマー管理ポリシーのデフォルトバージョンを設定するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーのリストで、デフォルトバージョンを設定するポリシーのポリシー名を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。

4. [ポリシーのバージョン] タブを選択します。デフォルトのバージョンとして設定するバージョンの横にあるチェックボックスをオンにし、[デフォルトとして設定] を選択します。

AWS Command Line Interface または AWS API からカスタマー管理ポリシーのデフォルトバージョンを設定するには、「[カスタマー管理ポリシーの編集 \(AWS CLI\)](#)」を参照してください。

## バージョンを使用して変更をロールバックする

カスタマー管理ポリシーのデフォルトバージョンを設定して、変更をロールバックすることができます。たとえば、次のシナリオを考えてみます。

AWS Management Console を使用して特定の Amazon S3 バケットの管理をユーザーに許可するカスタマー管理ポリシーを作成するとします。作成した時点では、存在するバージョンは v1 として識別される 1 つのバージョンのみであり、そのバージョンが自動的にデフォルトとして設定されます。ポリシーは目的どおりに機能します。

後から、ポリシーを更新して 2 つ目の Amazon S3 バケットを管理するアクセス許可を追加します。IAM は、変更内容を含む v2 ポリシーの新しいバージョンを作成します。v2 をデフォルトのバージョンとして設定してからしばらくして、Amazon S3 コンソールを使用する権限がないという報告をユーザーから受けたとします。この場合は、ポリシーのバージョンを、目的どおりに機能することがわかっている v1 にロールバックできます。これを行うには、デフォルトのバージョンとして v1 を設定します。ユーザーは Amazon S3 コンソールを使用して元のバケットを管理できるようになりました。

後で、ポリシーのバージョン v2 のエラーを確認してから、再度ポリシーを更新して 2 つ目の Amazon S3 バケットを管理するアクセス許可を追加します。IAM は、v3 として識別される別の新しいバージョンのポリシーを作成します。デフォルトのバージョンとして v3 を設定すると、このバージョンは目的どおりに機能します。この時点で、ポリシーのバージョン v2 を削除します。

## バージョンの制限

マネージド ポリシーは最大 5 つのバージョンを持つことができます。AWS Command Line Interface または AWS API を使用して、管理ポリシーのバージョンを 5 つ以上にすると必要な場合は、まず既存のバージョンを 1 つ以上削除する必要があります。AWS Management Console を使用している場合は、ポリシーを編集する前にバージョンを削除する必要はありません。6 番目のバージョンを保存する際、デフォルト以外のポリシーのバージョンを 1 つ以上削除することを求めるダイアログボックスが表示されます。確認しやすいように、各バージョンの JSON ポリシードキュメントを表示します。このダイアログボックスについては、「[the section called "IAM ポリシーの編集"](#)」を参照してください。

管理ポリシーのデフォルトバージョン以外の任意のバージョンを削除できます。バージョンを削除しても、残りのバージョンのバージョン ID は変更されません。その結果、バージョン ID はシーケンシャルではなくなる場合があります。たとえば、管理ポリシーのバージョン v2 と v4 を削除して 2 つの新しいバージョンを追加すると、残ったバージョンの ID は v1、v3、v5、v6、v7 となる可能性があります。

## IAM ポリシーの編集

[ポリシー](#) は、ID またはリソースにアタッチされたときに、そのアクセス許可を定義するエンティティです。ポリシーは JSON ドキュメントとして AWS に格納され、IAM のアイデンティティベースのポリシーとしてプリンシパルに添付されます。IAM ユーザーグループ、ユーザー、ロールなどのプリンシパル (またはアイデンティティ) に、アイデンティティベースのポリシーをアタッチできます。アイデンティティベースのポリシーには、AWS 管理ポリシー、カスタマー管理ポリシー、および [インラインポリシー](#) があります。IAM でカスタマー管理ポリシーとインラインポリシーを編集できます。AWS 管理ポリシーは編集できません。AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

### トピック

- [ポリシーアクセスの確認](#)
- [カスタマー管理ポリシーの編集 \(コンソール\)](#)
- [インラインポリシーの編集 \(コンソール\)](#)
- [カスタマー管理ポリシーの編集 \(AWS CLI\)](#)
- [カスタマー管理ポリシーの編集 \(AWS API\)](#)

### ポリシーアクセスの確認

ポリシーのアクセス許可を変更する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

### カスタマー管理ポリシーの編集 (コンソール)

カスタマー管理ポリシーを編集して、ポリシーで定義されたアクセス許可を変更することができます。カスタマー管理ポリシーは最大 5 つのバージョンを持つことができます。これは、管理ポリシーのバージョンを 5 つ以上にする必要のある場合は、AWS Management Console より、削除するバージョンを指定するよう求められるため、重要です。ポリシーを編集する前に、デフォルトのバージョ




ンを変更したり、ポリシーのバージョンを削除したりすることもできます。バージョンの詳細については、「[IAM ポリシーのバージョニング](#)」を参照してください。

カスタマー管理ポリシーを編集するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、編集するポリシーの名前を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [アクセス許可] タブを選択し、[編集] を選択します。
5. 以下のいずれかを実行します。
  - [ビジュアル] オプションを選択し、JSON 構文を理解することなくポリシーを変更します。ポリシー内の各権限ブロックのサービス、アクション、リソース、またはオプションの条件を変更することができます。また、ポリシーをインポートして、ポリシーの最下部に権限を追加することもできます。変更が完了したら、[次へ] を選択して続行します。
  - [JSON] オプションを選択し、JSON テキストボックスにテキストを入力または貼り付けてポリシーを変更します。また、ポリシーをインポートして、ポリシーの最下部に権限を追加することもできます。[ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。
6. [確認して保存] ページで、このポリシーで定義されているアクセス許可を確認し、[変更を保存] を選択して作業を保存します。
7. 最大 5 つのバージョンの管理ポリシーがすでにある場合は、[変更を保存] を選択すると、ダイアログボックスが表示されます。新しいバージョンを保存するには、ポリシーの最も古い非デフォルトバージョンが削除され、この新しいバージョンに置き換えられます。オプションで、新しいバージョンをポリシーのデフォルトバージョンとして設定できます。

[ポリシーを保存] を選択して、新しいバージョンのポリシーを保存します。

 Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

## カスタマー管理ポリシーのデフォルトバージョンを設定するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーのリストで、デフォルトバージョンを設定するポリシーのポリシー名を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [ポリシーのバージョン] タブを選択します。デフォルトのバージョンとして設定するバージョンの横にあるチェックボックスをオンにし、[デフォルトとして設定] を選択します。

## カスタマー管理ポリシーのバージョンを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. 削除するバージョンのカスタマー管理ポリシーの名前を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [ポリシーのバージョン] タブを選択します。削除するバージョンの横にあるチェックボックスをオンにします。その後、[Delete] (削除) をクリックします。
5. バージョンを削除することを確認し、[Delete (削除)] を選択します。

## インラインポリシーの編集 (コンソール)

インラインポリシーは AWS Management Console から編集できます。

### ユーザー、ユーザーグループ、ロールのインラインポリシーを編集するには (コンソール)

1. ナビゲーションペインで、[ユーザー]、[ユーザーグループ]、または [ロール] を選択します。
2. ポリシーを修正するユーザー、ユーザーグループ、またはロールの名前を選択します。次に [Permissions (アクセス許可)] タブを選択し、ポリシーを展開します。
3. インラインポリシーを編集するには、[ポリシーの編集] を選択します。
4. 以下のいずれかを実行します。
  - [ビジュアル] オプションを選択し、JSON 構文を理解することなくポリシーを変更します。ポリシー内の各権限ブロックのサービス、アクション、リソース、またはオプションの条件を変

更することができます。また、ポリシーをインポートして、ポリシーの最下部に権限を追加することもできます。変更が完了したら、[次へ] を選択して続行します。

- [JSON] オプションを選択し、JSON テキストボックスにテキストを入力または貼り付けてポリシーを変更します。また、ポリシーをインポートして、ポリシーの最下部に権限を追加することもできます。[ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。現在アタッチされているエンティティには影響を与えず変更を保存するには、[デフォルトのバージョンとして設定] チェックボックスをオフにします。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

5. [確認] ページで、ポリシーの概要を確認してから、[変更を保存] を選択して作業を保存します。

## カスタマー管理ポリシーの編集 (AWS CLI)

カスタマー管理ポリシーは AWS Command Line Interface (AWS CLI) で編集できます。

#### Note

マネージド ポリシーは最大 5 つのバージョンを持つことができます。変更するカスタマー管理ポリシーのバージョンが 5 つを超える場合は、最初に 1 つ以上の既存のバージョンを削除する必要があります。

カスタマー管理ポリシーを編集するには (AWS CLI)

1. (オプション) ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [get-policy](#)
2. (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のコマンドを実行します。

- 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示する場合:
    - [list-entities-for-policy](#)
  - アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされた管理ポリシーを一覧表示する場合:
    - [list-attached-user-policies](#)
    - [list-attached-group-policies](#)
    - [list-attached-role-policies](#)
3. カスタマー管理ポリシーを編集するには、次のコマンドを実行します。
    - [create-policy-version](#)
  4. (オプション) カスタマー管理ポリシーを検証するには、次の IAM Access Analyzer コマンドを実行します。
    - [ポリシーの検証](#)

カスタマー管理ポリシーのデフォルトバージョンを設定するには (AWS CLI)

1. (オプション) 管理ポリシーを一覧表示するには、次のコマンドを実行します。
  - [list-policies](#)
2. カスタマー管理ポリシーのデフォルトバージョンを設定するには、次のコマンドを実行します。
  - [set-default-policy-version](#)

カスタマー管理ポリシーのバージョンを削除するには (AWS CLI)

1. (オプション) 管理ポリシーを一覧表示するには、次のコマンドを実行します。
  - [list-policies](#)
2. カスタマー管理ポリシーを削除するには、次のコマンドを実行します。
  - [delete-policy-version](#)

## カスタマー管理ポリシーの編集 (AWS API)

カスタマー管理ポリシーは AWS API で編集できます。

### Note

マネージド ポリシーは最大 5 つのバージョンを持つことができます。変更するカスタマー管理ポリシーのバージョンが 5 つを超える場合は、最初に 1 つ以上の既存のバージョンを削除する必要があります。

カスタマー管理ポリシーを編集するには (AWS API)

1. (オプション) ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
2. (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のオペレーションを呼び出します。
  - 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示する場合:
    - [ListEntitiesForPolicy](#)
  - アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされた管理ポリシーを一覧表示する場合:
    - [ListAttachedUserPolicies](#)
    - [ListAttachedGroupPolicies](#)
    - [ListAttachedRolePolicies](#)
3. カスタマー管理ポリシーを編集するには、次のオペレーションを呼び出します。
  - [CreatePolicyVersion](#)
4. (オプション) カスタマー管理ポリシーを検証するには、次の IAM Access Analyzer オペレーションを呼び出します。
  - [検証ポリシー](#)

カスタマー管理ポリシーのデフォルトバージョンを設定するには (AWS API)

1. (オプション) 管理ポリシーを一覧表示するには、次のオペレーションを呼び出します。
  - [ListPolicies](#)
2. カスタマー管理ポリシーのデフォルトバージョンを設定するには、次のオペレーションを呼び出します。
  - [SetDefaultPolicyVersion](#)

カスタマー管理ポリシーのバージョンを削除するには (AWS API)

1. (オプション) 管理ポリシーを一覧表示するには、次のオペレーションを呼び出します。
  - [ListPolicies](#)
2. カスタマー管理ポリシーを削除するには、次のオペレーションを呼び出します。
  - [DeletePolicyVersion](#)

## IAM ポリシーを削除する

AWS Management Console、AWS Command Line Interface ( AWS CLI )、または IAM API を使用して、IAM ポリシーを削除できます。

### Note

IAM ポリシーの削除は永久的です。ポリシーが削除された後、復元できません。

管理ポリシーとインラインポリシーの違いの詳細については、「[管理ポリシーとインラインポリシー](#)」を参照してください。

IAM ポリシーの一般情報については、「[IAM でのポリシーとアクセス許可](#)」を参照してください。

AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STSクォータ](#)」を参照してください。

トピック

- [ポリシーアクセスの確認](#)

- [IAM ポリシーの削除 \(コンソール\)](#)
- [IAM ポリシーの削除 \(AWS CLI\)](#)
- [IAM ポリシーの削除 \(AWS API\)](#)

## ポリシーアクセスの確認

ポリシーを削除する前に、サービスレベルの最近のアクティビティを確認する必要があります。これは、アクセス権を使用しているプリンシパル (ユーザーまたはアプリケーション) から削除しないようにするために重要です。最後にアクセスした情報を表示する方法の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## IAM ポリシーの削除 (コンソール)

カスタマー管理ポリシーを削除して、AWS アカウント からこのポリシーを削除することができます。AWS 管理ポリシーを削除することはできません。

カスタマー管理ポリシーを削除するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. 削除するカスタマー管理ポリシーの横にあるラジオボックスを選択します。検索ボックスを使用して、グループのリストをフィルタリングできます。
4. [Actions] (アクション) を選択してから、[Delete] (削除) をクリックします。
5. 指示に従ってポリシーを削除することを確認し、[削除] を選択します。

ユーザーグループ、ユーザー、ロールのインラインポリシーを削除するには (コンソール)

1. ナビゲーションペインで、[ユーザーグループ]、[ユーザー]、または [ロール] を選択します。
2. ポリシーを削除するユーザーグループ、ユーザー、またはロールの名前を選択します。次に、[Permissions (アクセス許可)] タブを選択します。
3. 削除するポリシーの横にあるチェックボックスを選択して、[削除] を選択します。ユーザーまたはロールのインラインポリシーを削除するには、[削除] を選択して削除を確定します。1つのインラインポリシーを削除する場合は、[ユーザーグループ] にポリシーの名前を入力し、[削除] を選択します。複数のインラインポリシーを選択する場合は、[ユーザーグループ] に削除するポリ

シーの数を入力し、**inline policies** に続いて [削除] を選択します。たとえば、3つのインラインポリシーを削除する場合は、**3 inline policies** と入力します。

## IAM ポリシーの削除 (AWS CLI)

カスタマー管理ポリシーは AWS Command Line Interface から削除できます。

カスタマー管理ポリシーを削除するには (AWS CLI)

- (オプション) ポリシーの情報を表示するには、以下のコマンドを実行します。
  - 管理ポリシーを一覧表示するには: [list-policies](#)
  - 管理ポリシーの詳細情報を取得するには: [get-policy](#)
- (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のコマンドを実行します。
  - 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示するには、次のコマンドを実行します。
    - [list-entities-for-policy](#)
  - アイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチされている管理ポリシーを一覧表示するには、以下のいずれかのコマンドを実行します。
    - [list-attached-user-policies](#)
    - [list-attached-group-policies](#)
    - [list-attached-role-policies](#)
- カスタマー管理ポリシーを削除するには、次のコマンドを実行します。
  - [delete-policy](#)

インラインポリシーを削除するには (AWS CLI)

- (オプション) アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされたすべてのインラインポリシーを一覧表示するには、以下のいずれかのコマンドを使用します。
  - [aws iam list-user-policies](#)
  - [aws iam list-group-policies](#)
  - [aws iam list-role-policies](#)



2. (オプション) アイデンティティ (ユーザー、ユーザーグループ、またはロール) に埋め込まれたインラインポリシードキュメントを取得するには、以下のいずれかのコマンドを使用します。
  - [aws iam get-user-policy](#)
  - [aws iam get-group-policy](#)
  - [aws iam get-role-policy](#)
3. インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) から削除するには、以下のいずれかのコマンドを使用します。
  - [aws iam delete-user-policy](#)
  - [aws iam delete-group-policy](#)
  - [aws iam delete-role-policy](#)

## IAM ポリシーの削除 (AWS API)

カスタマー管理ポリシーは AWS API で削除できます。

カスタマー管理ポリシーを削除するには (AWS API)

1. (オプション) ポリシーの情報を表示するには、以下のオペレーションを呼び出します。
  - 管理ポリシーを一覧表示するには: [ListPolicies](#)
  - 管理ポリシーの詳細情報を取得するには: [GetPolicy](#)
2. (オプション) ポリシーとアイデンティティとの関係を確認するには、以下のオペレーションを呼び出します。
  - 管理ポリシーがアタッチされたアイデンティティ (ユーザー、ユーザーグループ、およびロール) を一覧表示するには、次のオペレーションを呼び出します。
    - [ListEntitiesForPolicy](#)
  - アイデンティティ (ユーザー、ユーザーグループ、またはロール) にアタッチされている管理ポリシーを一覧表示するには、以下のいずれかのオペレーションを呼び出します。
    - [ListAttachedUserPolicies](#)
    - [ListAttachedGroupPolicies](#)
    - [ListAttachedRolePolicies](#)
3. カスタマー管理ポリシーを削除するには、次のオペレーションを呼び出します。

- [DeletePolicy](#)

インラインポリシーを削除するには (AWS API)

1. (オプション) アイデンティティ (ユーザー、ユーザーグループ、ロール) にアタッチされたすべてのインラインポリシーを一覧表示するには、以下のいずれかのオペレーションを呼び出します。
  - [ListUserPolicies](#)
  - [ListGroupPolicies](#)
  - [ListRolePolicies](#)
2. (オプション) アイデンティティ (ユーザー、ユーザーグループ、またはロール) に埋め込まれたインラインポリシードキュメントを取得するには、以下のいずれかのオペレーションを呼び出します。
  - [GetUserPolicy](#)
  - [GetGroupPolicy](#)
  - [GetRolePolicy](#)
3. インラインポリシーを ID (ユーザー、ユーザーグループ、または [サービスにリンクされたロール](#) 以外のロール) から削除するには、以下のいずれかのオペレーションを呼び出します。
  - [DeleteUserPolicy](#)
  - [DeleteGroupPolicy](#)
  - [DeleteRolePolicy](#)

## 最終アクセス情報を使用した AWS のアクセス許可の調整

管理者として、IAM リソース (ロール、ユーザー、ユーザーグループまたはポリシー) に必要以上のアクセス許可を付与する可能性があります。IAM は、未使用のアクセス許可を特定して削除できるようにするために、最終アクセス情報を提供します。サービスの最終アクセス情報を使用して、ポリシーを調整し、IAM ID およびポリシーで使用されるサービスとアクションにのみアクセスを許可することができます。これにより、[最小権限のベストプラクティス](#)に準拠できるようになります。IAM または AWS Organizations に存在する ID またはポリシーの最終アクセス情報を表示できます。

未使用のアクセスアナライザーを使用して、最終アクセス時間情報を継続的に監視できます。詳しくは、「[外部アクセスと未使用のアクセスに関する検出結果](#)」を参照してください。

## トピック

- [IAM の最終アクセス情報タイプ](#)
- [AWS Organizations の最終アクセス情報](#)
- [最終アクセス情報についての主要事項](#)
- [必要な許可](#)
- [IAM および Organizations エンティティのトラブルシューティングアクティビティ](#)
- [AWS が最終アクセス情報を追跡する場所](#)
- [IAM の最終アクセス情報の表示](#)
- [の最終アクセス情報の表示](#)
- [最終アクセス情報を使用するシナリオ例](#)
- [IAM アクションの最終アクセス情報サービスとアクション](#)

## IAM の最終アクセス情報タイプ

IAM ID の最終アクセス情報には、許可された AWS サービス情報と許可されたアクション情報の 2 つのタイプが表示されます。この情報には、AWS API へのアクセスの試行が行われた日時が含まれます。アクションについては、最終アクセス情報によってサービス管理アクションが報告されます。管理アクションには、作成、削除、および変更アクションが含まれます。IAM の最終アクセス情報の表示方法の詳細については、「[IAM の最終アクセス情報の表示](#)」を参照してください。

IAM ID に付与するアクセス許可に関する意思決定を行うために最終アクセス情報を使用するシナリオ例については、「[最終アクセス情報を使用するシナリオ例](#)」を参照してください。

管理アクションの情報の提供方法の詳細については、「[最終アクセス情報についての主要事項](#)」を参照してください。

## AWS Organizations の最終アクセス情報

管理アカウント認証情報を使用してサインインした場合、組織の IAM エンティティ AWS Organizations またはポリシーに関してサービスの最終アクセス情報を表示することもできます。AWS Organizations エンティティには、組織のルート、組織単位 (OU)、またはアカウントが含まれています。AWS Organizations の最終アクセス情報には、サービスコントロールポリシー (SCP) によって許可されるサービスに関する情報が含まれます。この情報は、組織またはアカウント内のど

のプリンシパル (ルートユーザー、IAM ユーザーまたはロール) が最後にサービスにアクセスしようとしたかを示しています。レポートの詳細と AWS Organizations の最終アクセス情報の表示方法については、「[の最終アクセス情報の表示](#)」を参照してください。

Organizations エンティティに付与するアクセス許可に関する意思決定を行うために最終アクセス情報を使用するシナリオ例については、「[最終アクセス情報を使用するシナリオ例](#)」を参照してください。

## 最終アクセス情報についての主要事項

レポートの最終アクセス情報を使用して IAM アイデンティティまたは組織エンティティのアクセス許可を変更する前に、その情報に関する次の詳細を確認してください。

- 追跡期間 – 最近のアクティビティは、4 時間以内に IAM コンソールに表示されます。サービス情報の追跡期間は、サービスがアクション情報の追跡を開始した時期に応じて、400 日以上です。Amazon S3 アクション情報の追跡期間は、2020 年 4 月 12 日に始まりました。Amazon EC2、IAM、および Lambda アクションの追跡期間は、2021 年 4 月 7 日から始まりました。その他すべてのサービスの追跡期間は 2023 年 5 月 23 日に開始されました。アクションの最終アクセス情報があるサービスのリストについては、「[IAM アクションの最終アクセス情報サービスとアクション](#)」を参照してください。どのリージョンのアクションの最終アクセス情報が参照できるかについての詳細は、「[AWS が最終アクセス情報を追跡する場所](#)」を参照してください
- 報告された試行 – サービスの最終アクセス時間データには、成功したものだけでなく、AWS API へのすべてのアクセスの試行が含まれます。これには、AWS Management Console、いずれかの SDK を通じた AWS API、またはその他のコマンドラインツールを使用して行われたすべての試行が含まれます。サービスの最終アクセス時間データで予期しないエントリが表示されても、リクエストが拒否された可能性があるため、アカウントが侵害されたことを意味するわけではありません。すべての API 呼び出しと、それらが成功したか、アクセスが拒否されたかに関する情報については、権威あるソースとして CloudTrail ログを参照してください。
- PassRole – iam:PassRole アクションは追跡されず、IAM アクションの最終アクセス情報には含まれません。
- アクションの最終アクセス情報 — アクションの最終アクセス情報は、IAM ID がアクセスするサービス管理アクションで使用できます。どのアクションの最終アクセス情報が報告されるかについては、「[サービスの一覧とアクション](#)」を参照してください。

### Note

アクセスの最終時間情報は、Amazon S3 データイベントでは使用できません。

- 管理イベント — IAM は、CloudTrail によってログ記録されるサービス管理イベントのアクション情報を提供します。CloudTrail 管理イベントは、コントロールプレーンオペレーションまたはコントロールプレーンイベントと呼ばれることもあります。管理イベントでは、AWS アカウントのリソースで実行される管理オペレーションについて知ることができます。CloudTrail の管理イベントの詳細については、AWS CloudTrail ユーザーガイドの「[Cloudtrail を使用した管理イベントのログ記録](#)」を参照してください。
- レポート所有者 — レポートを生成するプリンシパルのみがレポートの詳細を表示できます。つまり、AWS Management Console で情報を表示するときに、情報が生成されてロードされるのを待たなければならない場合があります。AWS CLI または AWS API を使用してレポートの詳細を取得する場合、認証情報はレポートを生成したプリンシパルの認証情報と一致する必要があります。ロールまたはフェデレーテッドユーザーに対して一時的な認証情報を使用する場合は、同じセッション中にレポートを生成および取得する必要があります。引き受けたロールセッションのプリンシパルの詳細については、「[AWS JSON ポリシーの要素: Principal](#)」を参照してください。
- IAM リソース — IAM の最終アクセス情報には、アカウントの IAM リソース (ロール、ユーザー、ユーザーグループ、ポリシー) が含まれます。Organizations への最終アクセスに関する情報には、指定された Organizations エンティティのプリンシパル (IAM ユーザー、IAM ロール、または AWS アカウントのルートユーザー) が含まれます。最終アクセス情報には、認証されていない試行は含まれません。
- IAM ポリシータイプ — IAM の最終アクセス情報には、IAM アイデンティティポリシーで許可されているサービスが含まれます。以下は、ロールにアタッチされたポリシーか、ユーザーに直接、またはグループ経由でアタッチされたポリシーを示します。他のポリシータイプで許可されているアクセスはレポートに含まれていません。除外されたポリシータイプには、リソースベースのポリシー、アクセスコントロールリスト、AWS Organizations SCP、IAM アクセス許可の境界、およびセッションポリシーなどがあります。サービスにリンクされたロールによって提供されるアクセス許可は、リンク先のサービスによって定義され、IAM で変更することはできません。サービスにリンクされたロールの詳細については、「[サービスリンクロールの使用](#)」を参照してください。さまざまなポリシータイプを評価してアクセスを許可または拒否する方法については、「[ポリシーの評価論理](#)」を参照してください。
- Organizations ポリシータイプ — AWS Organizations の情報には、Organizations エンティティの継承されたサービスコントロールポリシー (SCP) で許可されているサービスが含まれます。SCP は、Root、OU、またはアカウントに接続されているポリシーです。他のポリシータイプで許可されているアクセスはレポートに含まれていません。除外されたポリシータイプには、アイデンティティベースのポリシー、リソースベースのポリシー、アクセスコントロールリスト、IAM アクセス許可の境界、およびセッションポリシーなどがあります。アクセスを許可または拒否するために各ポリシータイプを評価する方法については、「[ポリシーの評価論理](#)」を参照してください。

- ポリシー ID の指定 – AWS CLI または AWS API を使用して Organizations で最終アクセス情報のレポートを生成する際、オプションでポリシー ID を指定できます。生成されたレポートには、そのポリシーのみで許可されているサービスの情報が含まれます。この情報には、指定した IAM エンティティまたはエンティティの子の最新のアカウントアクティビティが含まれます。詳細については、「[aws iam generate-organizations-access-report](#)」または「[GenerateOrganizationsAccessReport](#)」を参照してください。
- Organizations 管理アカウント – サービスの最終アクセス情報を表示するには、組織の管理アカウントにサインインする必要があります。IAM コンソール、AWS CLI、または AWS API を使用して管理アカウントのデータを表示するように選択できます。管理アカウントは SCP によって制限されないため、結果として得られるレポートにすべての AWS サービスがリストされます。CLI または API でポリシー ID を指定した場合、ポリシーは無視されます。サービスごとに、管理アカウントのみの情報がレポートに含まれます。ただし、他の IAM エンティティのレポートによって、管理アカウントのアクティビティに対する情報は返されません。
- Organizations 設定 – 管理者は、Organizations のデータを生成する前に[組織のルートで SCP を有効にする](#)必要があります。

## 必要な許可

AWS Management Console で最終アクセス情報を表示するには、必要なアクセス許可を付与するポリシーが必要です。

### IAM 情報のアクセス許可

IAM コンソールを使用して、ユーザー、ロール、またはポリシーの最終アクセス情報を表示するには、以下のアクションを含むポリシーが必要です。

- iam:GenerateServiceLastAccessedDetails
- iam:Get\*
- iam:List\*

これらのアクセス許可では、ユーザーは、以下を表示することができます。

- [管理ポリシー](#) にアタッチされているユーザー、グループ、またはロール
- ユーザーまたはロールがアクセスできるサービス
- 最後にサービスにアクセスした時間

- 最後に特定の Amazon EC2、IAM、Lambda、または Amazon S3 アクションを使用しようとした時間

AWS CLI または AWS API を使用して IAM の最終アクセス情報を表示するには、使用するオペレーションに一致するアクセス許可も必要です。

- iam:GenerateServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetailsWithEntities
- iam:ListPoliciesGrantingServiceAccess

この例では、IAM の最終アクセス情報の表示を許可する ID ベースのポリシーを作成する方法を示します。さらに、IAM のすべてのへの読み取り専用アクセスが許可されます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GenerateServiceLastAccessedDetails",
      "iam:Get*",
      "iam:List*"
    ],
    "Resource": "*"
  }
}
```

## AWS Organizations 情報のアクセス許可

IAM コンソールを使用して Organizations のルート、OU、またはアカウントエンティティのレポートを表示するには、以下のアクションを含むポリシーが必要です。

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribeAccount
- organizations:DescribeOrganization
- organizations:DescribeOrganizationalUnit



- `organizations:DescribePolicy`
- `organizations:ListChildren`
- `organizations:ListParents`
- `organizations:ListPoliciesForTarget`
- `organizations:ListRoots`
- `organizations:ListTargetsForPolicy`

AWS CLI または AWS API を使用して Organizations のサービスの最終アクセス情報を表示するには、以下のアクションを含むポリシーが必要です。

- `iam:GenerateOrganizationsAccessReport`
- `iam:GetOrganizationsAccessReport`
- `organizations:DescribePolicy`
- `organizations:ListChildren`
- `organizations:ListParents`
- `organizations:ListPoliciesForTarget`
- `organizations:ListRoots`
- `organizations:ListTargetsForPolicy`

この例では、組織のサービスの最終アクセス情報の閲覧を許可する ID ベースのポリシーを作成する方法を示します。さらに、Organizations のすべてのへの読み取り専用アクセスが許可されます。このポリシーは、プログラムおよびコンソールアクセスのアクセス許可を定義します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GenerateOrganizationsAccessReport",
      "iam:GetOrganizationsAccessReport",
      "organizations:Describe*",
      "organizations:List*"
    ],
    "Resource": "*"
  }
}
```



```
}
```

また、[iam:OrganizationsPolicyId](#) 条件キーを使用して、特定の Organizations ポリシーのみのレポートの生成を許可することもできます。ポリシーの例については、「[IAM: Organizations ポリシーのサービスの最終アクセス情報を表示](#)」を参照してください。

## IAM および Organizations エンティティのトラブルシューティングアクティビティ

場合によっては、AWS Management Consoleの最終アクセス情報のテーブルが空になる可能性もあります。または、AWS CLI または AWS API リクエストにより、空の情報セットまたは null フィールドが返される可能性があります。このような場合は、次の点を確認してください。

- アクションの最終アクセス情報については、表示される予定のアクションがリストに返されないことがあります。これは、IAM アイデンティティがアクションのアクセス許可を持っていないか、最終アクセス情報のアクションを AWS がまだ追跡していないために発生します。
- IAM ユーザーの場合は、直接またはグループメンバーシップで、インラインポリシーまたは管理ポリシーが 1 つ以上アタッチされていることを確認します。
- IAM グループの場合は、グループにインラインポリシーまたは管理ポリシーが 1 つ以上アタッチされていることを確認します。
- IAM グループの場合は、サービスにアクセスするためにグループのポリシーを使用したメンバーのサービスの最終アクセス情報のみ、レポートにより返されます。メンバーが他のポリシーを使用していたかどうかを確認するには、そのユーザーの最終アクセス情報を確認します。
- IAM ロールの場合は、ロールにインラインポリシーまたは管理ポリシーが 1 つ以上アタッチされていることを確認します。
- IAM エンティティ (ユーザーまたはロール) の場合は、そのエンティティのアクセス許可に影響する可能性のある他のポリシータイプを確認します。このポリシータイプには、リソースベースのポリシー、アクセスコントロールリスト、AWS Organizations ポリシー、IAM アクセス許可の境界、またはセッションポリシーなどがあります。詳細については、「[ポリシータイプ](#)」または「[単一アカウント内のポリシーを評価する](#)」を参照してください。
- IAM ポリシーで、指定した管理ポリシーが、1 人以上のユーザー、メンバーを持つグループ、またはロールに関連付けられていることを確認します。
- Organizations エンティティ (ルート、OU、またはアカウント) の場合は、Organizations 管理アカウント認証情報を使用してサインインしていることを確認します。
- [SCP が組織のルートで有効になっている](#)ことを確認します。
- アクション最終アクセス時間情報は、[IAM アクションの最終アクセス情報サービスとアクション](#) にリストされているアクションでのみ使用できます。

変更したら、アクティビティが IAM コンソールレポートに表示されるまで 4 時間以上かかります。AWS CLI または AWS API を使用する場合は、新しいレポートを生成して更新データを表示する必要があります。

## AWS が最終アクセス情報を追跡する場所

AWS は、標準の AWS リージョンの最終アクセス情報を収集します。AWS でさらにリージョンを追加すると、AWS の各リージョンにおける情報の追跡開始日とあわせて、これらのリージョンが以下のテーブルに追加されます。

- サービス情報 – サービスの追跡期間は過去 400 日以上です。リージョンが過去 400 日以内にこの機能を追跡し始めた場合は、それよりも短くなります。
- アクション情報 – Amazon S3 管理アクションの追跡期間は、2020 年 4 月 12 日に始まりました。Amazon EC2、IAM、および Lambda の管理アクションの追跡期間は、2021 年 4 月 7 日に始まりました。その他すべてのサービスの管理アクションの追跡期間は、2023 年 5 月 23 日に開始します。リージョンの追跡日が 2023 年 5 月 23 日より後の場合、そのリージョンのアクション最終アクセス日情報は後の日付から開始されます。

リージョン名	リージョン	追跡開始日
米国東部 ( オハイオ )	us-east-2	2017 年 10 月 27 日
米国東部 ( バージニア北部 )	us-east-1	2015 年 10 月 1 日
米国西部 ( 北カリフォルニア )	us-west-1	2015 年 10 月 1 日
米国西部 ( オレゴン )	us-west-2	2015 年 10 月 1 日
アフリカ ( ケープタウン )	af-south-1	2020 年 4 月 22 日
アジアパシフィック ( 香港 )	ap-east-1	2019 年 4 月 24 日
アジアパシフィック ( ハイデラバード )	ap-south-2	2022 年 11 月 22 日
アジアパシフィック ( ジャカルタ )	ap-southeast-3	2021 年 12 月 13 日

リージョン名	リージョン	追跡開始日
アジアパシフィック (メルボルン)	ap-southeast-4	2023 年 1 月 23 日
アジアパシフィック (ムンバイ)	ap-south-1	2016 年 6 月 27 日
アジアパシフィック (大阪)	ap-northeast-3	2018 年 2 月 11 日
アジアパシフィック (ソウル)	ap-northeast-2	2016 年 1 月 6 日
アジアパシフィック (シンガポール)	ap-southeast-1	2015 年 10 月 1 日
アジアパシフィック (シドニー)	ap-southeast-2	2015 年 10 月 1 日
アジアパシフィック (東京)	ap-northeast-1	2015 年 10 月 1 日
カナダ (中部)	ca-central-1	2017 年 10 月 28 日
欧州 (フランクフルト)	eu-central-1	2015 年 10 月 1 日
欧州 (アイルランド)	eu-west-1	2015 年 10 月 1 日
欧州 (ロンドン)	eu-west-2	2017 年 10 月 28 日
欧州 (ミラノ)	eu-south-1	2020 年 4 月 28 日
欧州 (パリ)	eu-west-3	2017 年 12 月 18 日
欧州 (スペイン)	eu-south-2	2022 年 11 月 15 日
欧州 (ストックホルム)	eu-north-1	2018 年 12 月 12 日
欧州 (チューリッヒ)	eu-central-2	2022 年 11 月 8 日
イスラエル (テルアビブ)	il-central-1	2023 年 8 月 1 日

リージョン名	リージョン	追跡開始日
中東 (バーレーン)	me-south-1	2019 年 7 月 29 日
中東 (アラブ首長国連邦)	me-central-1	2022 年 8 月 30 日
南米 (サンパウロ)	sa-east-1	2015 年 12 月 11 日
AWS GovCloud (米国東部)	us-gov-east-1	2023 年 7 月 1 日
AWS GovCloud (米国西部)	us-gov-west-1	2023 年 7 月 1 日

前の表にリージョンが記載されていない場合、そのリージョンは、まだ最終アクセス情報を提供していません。

AWS リージョンは地理的な領域内の AWS リソースのコレクションです。リージョンは、パーティションにグループ化されます。標準リージョンは、aws パーティションに属するリージョンです。各パーティションの詳細については、『AWS 全般のリファレンス』の「[Amazon Resource Names \(ARNs\) Format](#)」を参照してください。リージョンの詳細については、『AWS 全般のリファレンス』の「[About AWS Regions](#)」も参照してください。

## IAM の最終アクセス情報の表示

AWS Management Console、AWS CLI、または AWS API を使用して、IAM の最終アクセス情報を表示できます。最終アクセス情報が表示される[サービスとそのアクションのリスト](#)を参照してください。最終アクセス情報の詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

IAM では、次の各タイプのリソースに関する情報を表示できます。いずれの場合も、情報には、指定された報告期間に許可されたサービスが含まれます。

- ユーザー – ユーザーが許可された各サービスへのアクセスを最後に試みた時間を表示します。
- ユーザーグループ – グループメンバーが許可された各サービスへのアクセスを最後に試みた時間に関する情報を表示します。また、このレポートには、アクセスを試みたメンバーの合計数も表示されます。
- ロール – 許可された各サービスへのアクセスにおいて、ユーザーが最後にロールを使用した時間を表示します。

- ポリシー – ユーザーまたはロールが許可された各サービスへのアクセスを最後に試みた時間に関する情報を表示します。また、このレポートには、アクセスを試みたエンティティの合計数も表示されます。

#### Note

IAM のリソースに関するアクセス情報を表示する前に、情報のレポート期間、レポート対象のエンティティ、評価対象のポリシータイプをご確認ください。詳細については、「[the section called “最終アクセス情報についての主要事項”](#)」を参照してください。

## IAM の情報の表示 (コンソール)

IAM コンソールの [アクセスアドバイザー] タブで、IAM の最終アクセス情報を表示できます。

### IAM の情報を表示するには (コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ユーザーグループ]、[ユーザー]、[ロール]、または [ポリシー] を選択します。
3. 任意のユーザー、ユーザーグループ、ロール、またはポリシーの名前をクリックして、その[概要] ページを開き、[アクセスアドバイザー] タブを選択します。選択したリソースに基づき、次の情報を表示します。
  - ユーザーグループ – ユーザーグループメンバー (ユーザー) がアクセスできるサービスのリストを表示します。また、メンバーが最後にサービスにアクセスした日時、使用したユーザーグループポリシー、リクエストを行ったユーザーグループメンバーも表示できます。ポリシーの名前を選択して、ポリシーが管理ポリシーかインラインユーザーグループポリシーかを確認します。ユーザーグループメンバーの名前を選択して、ユーザーグループのすべてのメンバーと、サービスへの最終アクセス時間を選択します。
  - ユーザー – ユーザーがアクセスできるサービスのリストを表示します。また、最後にサービスにアクセスした日時と現在ユーザーに関連付けられているポリシーも表示できます。ポリシーの名前を選択して、そのポリシーが管理ポリシー、インラインユーザーポリシー、ユーザーグループのインラインポリシーのいずれであるかを確認します。

- ロール – ロールでアクセスできるサービス、サービスへのロールの最終アクセス時間、および使用したポリシーのリストを表示します。ポリシーの名前を選択して、ポリシーが管理ポリシーかインラインロールポリシーかを確認します。
  - ポリシー – ポリシーで許可されたアクションを含むサービスのリストを表示します。また、サービスにアクセスするために最後にポリシーが使用された日時と、ポリシーを使用したエンティティ (ユーザーまたはロール) も表示できます。最終アクセス日には、別のポリシーを通じてこのポリシーへのアクセスが許可された日付も含まれます。エンティティの名前を選択して、このポリシーがアタッチされているエンティティや、サービスへの最終アクセス時間を確認します。
4. テーブルの [サービス] 列で、[アクションの最終アクセス情報を含むサービス](#) の名前のいずれかを選択し、IAM エンティティがアクセスしようとした管理アクションのリストを表示します。AWS リージョン と、ユーザーが最後にアクションを実行しようとした日時を示すタイムスタンプを表示できます。
  5. [最終アクセス日] の列は、[アクションの最終アクセス情報を含むサービスとサービスの管理アクションに表示されます](#)。この列で返される可能性のある次の結果を確認します。これらの結果は、サービスやアクションが許可されているかどうか、アクセスされたかどうか、および最終アクセス情報が AWS によって追跡されているかどうかによって異なります。

#### <数値> 日前

追跡期間中にサービスまたはアクションが使用されてから経過した日数。サービスの追跡期間は、過去 400 日間です。Amazon S3 アクションの追跡期間は、2020 年 4 月 12 日に開始しました。Amazon EC2 IAM および Lambda アクションの追跡期間は、2021 年 4 月 7 日に開始しました。その他すべてのサービスの追跡期間は 2023 年 5 月 23 日に開始されました。各 AWS リージョン の追跡開始日の詳細については、「[AWS が最終アクセス情報を追跡する場所](#)」を参照してください。

#### 追跡期間内にアクセスがない

追跡対象のサービスまたはアクションが、追跡期間中にエンティティによって使用されていません。

リストに表示されないアクションに対するアクセス許可を持っている可能性があります。これは、アクションの追跡情報が現在 AWS に含まれていない場合に発生することがあります。追跡情報がないことだけに基づいてアクセス許可を決定しないでください。代わりに、この情報を使用して、最小限の権限を付与するという全体的な戦略を通知し、サポートすることをお勧めします。ポリシーをチェックして、アクセスレベルが適切であることを確認します。

## IAM の情報の表示 ( AWS CLI)

AWS CLI を使用して、IAMリソースが AWS サービスおよび Amazon S3、Amazon EC2、IAM、Lambda アクションへのアクセスを試行するために最後に使用された時刻に関する情報を取得できます。IAM リソースには、ユーザー、ユーザーグループ、ロール、またはポリシーを選択できます。

IAM の情報を表示するには (AWS CLI API)

1. レポートを生成します。レポートで必要な IAM がリクエストに含まれている必要があります。リソース (ユーザー、ユーザーグループ、ロール、またはポリシー) の ARN がリクエストに含まれている必要があります。レポートで生成する粒度レベルを指定して、いずれかのサービス、またはサービスとアクションの両方のアクセスの詳細を表示できます。リクエストにより `job-id` が返されます。これを使用して、`get-service-last-accessed-details` および `get-service-last-accessed-details-with-entities` オペレーションを使用して、ジョブが完了するまで、`job-status` をモニタリングできます。
  - [aws iam generate-service-last-accessed-details](#)
2. 前のステップの `job-id` パラメータを使用して、レポートに関する詳細を取得します。
  - [aws iam get-service-last-accessed-details](#)

このオペレーションでは、`generate-service-last-accessed-details` オペレーションでリクエストしたリソースのタイプと粒度レベルに基づき、次の情報が返されます。

- ユーザー – 指定したユーザーがアクセスできるサービスのリストを返します。サービスごとに、オペレーションは、ユーザーが最後に試行した日時とユーザーの ARN を返します。
- ユーザーグループ – 指定したユーザーグループのメンバーがアクセスできるサービスのリストを返します。このユーザーグループにアタッチされたポリシーを使用して、サービスのリストを返します。サービスごとに、オペレーションは、ユーザーグループメンバーが最後に試行した日時を返します。また、そのユーザーの ARN と、サービスにアクセスしようとしたユーザーグループメンバーの合計数も返ります。[GetServiceLastAccessedDetailsWithEntities](#) オペレーションを使用して、すべてのメンバーのリストを取得します。
- ロール – 指定したロールがアクセスできるサービスのリストを返します。サービスごとに、オペレーションは、ロールが最後に試行した日時とロールの ARN を返します。
- ポリシー – 指定されたポリシーがアクセスを許可するサービスのリストを返します。各サービスについて、オペレーションは、エンティティ (ユーザーまたはロール) が最後にポリシー



を使用してサービスにアクセスしようとした日時を返します。また、そのエンティティの ARN とアクセスを試みたエンティティの合計数も返ります。

3. 特定のサービスにアクセスするためにユーザーグループまたはポリシーのアクセス許可を使用したエンティティについて説明します。このオペレーションは、各エンティティの ARN、ID、名前、パス、タイプ (ユーザーまたはロール)、および最後にサービスにアクセスしようとしたエンティティのリストを返します。このオペレーションは、ユーザーとロールに対しても使用できませんが、そのエンティティに関する情報のみが返ります。

- [aws iam get-service-last-accessed-details-with-entities](#)

4. 特定のサービスにアクセスする際にアイデンティティ (ユーザー、ユーザーグループ、またはロール) が使用したアイデンティティベースのポリシーについて説明します。アイデンティティとサービスを指定すると、このオペレーションは、アイデンティティが指定されたサービスにアクセスするために使用できるアクセス許可ポリシーのリストを返します。このオペレーションは、ポリシーの現在の状態を示し、生成されたレポートには依存しません。また、他のポリシータイプ (例: リソースベースのポリシー、アクセスコントロールリスト、AWS Organizations ポリシー、IAM アクセス許可の境界、セッションポリシー) は返されません。詳細については、「[ポリシータイプ](#)」または「[単一アカウント内のポリシーを評価する](#)」を参照してください。

- [aws iam list-policies-granting-service-access](#)

## IAM の情報の表示 (AWS API)

AWS APIを使用して、IAMリソースが AWS サービスおよび Amazon S3、Amazon EC2、IAM、Lambda アクションへのアクセスを試行するために最後に使用された時刻に関する情報を取得できます。IAM リソースには、ユーザー、ユーザーグループ、ロール、またはポリシーを選択できます。レポートで生成する粒度レベルを指定して、いずれかのサービス、またはサービスとアクションの両方の詳細を表示できます。

## IAM の情報を表示するには (AWS API)

1. レポートを生成します。レポートに必要な IAM リソース (ユーザー、グループ、ロール、またはポリシー) の ARN がリクエストに含まれている必要があります。JobId が返ります。これを使用して、GetServiceLastAccessedDetails および GetServiceLastAccessedDetailsWithEntities オペレーションを使用して、ジョブが完了するまで、JobStatus をモニタリングできます。

- [GenerateServiceLastAccessedDetails](#)

2. 前のステップの JobId パラメータを使用して、レポートに関する詳細を取得します。



- [GetServiceLastAccessedDetails](#)

このオペレーションでは、GenerateServiceLastAccessedDetails オペレーションでリクエストしたリソースのタイプと粒度レベルに基づき、次の情報が返されます。

- ユーザー – 指定したユーザーがアクセスできるサービスのリストを返します。サービスごとに、オペレーションは、ユーザーが最後に試行した日時とユーザーの ARN を返します。
  - ユーザーグループ – 指定したユーザーグループのメンバーがアクセスできるサービスのリストを返します。このユーザーグループにアタッチされたポリシーを使用して、サービスのリストを返します。サービスごとに、オペレーションは、ユーザーグループメンバーが最後に試行した日時を返します。また、そのユーザーの ARN と、サービスにアクセスしようとしたユーザーグループメンバーの合計数も返ります。[GetServiceLastAccessedDetailsWithEntities](#) オペレーションを使用して、すべてのメンバーのリストを取得します。
  - ロール – 指定したロールがアクセスできるサービスのリストを返します。サービスごとに、オペレーションは、ロールが最後に試行した日時とロールの ARN を返します。
  - ポリシー – 指定されたポリシーがアクセスを許可するサービスのリストを返します。各サービスについて、オペレーションは、エンティティ (ユーザーまたはロール) が最後にポリシーを使用してサービスにアクセスしようとした日時を返します。また、そのエンティティの ARN とアクセスを試みたエンティティの合計数も返ります。
3. 特定のサービスにアクセスするためにユーザーグループまたはポリシーのアクセス許可を使用したエンティティについて説明します。このオペレーションは、各エンティティの ARN、ID、名前、パス、タイプ (ユーザーまたはロール)、および最後にサービスにアクセスしようとしたエンティティのリストを返します。このオペレーションは、ユーザーとロールに対しても使用できませんが、そのエンティティに関する情報のみが返ります。

- [GetServiceLastAccessedDetailsWithEntities](#)

4. 特定のサービスにアクセスする際にアイデンティティ (ユーザー、ユーザーグループ、またはロール) が使用したアイデンティティベースのポリシーについて説明します。アイデンティティとサービスを指定すると、このオペレーションは、アイデンティティが指定されたサービスにアクセスするために使用できるアクセス許可ポリシーのリストを返します。このオペレーションは、ポリシーの現在の状態を示し、生成されたレポートには依存しません。また、他のポリシータイプ (例: リソースベースのポリシー、アクセスコントロールリスト、AWS Organizations ポリシー、IAM アクセス許可の境界、セッションポリシー) は返されません。詳細については、「[ポリシータイプ](#)」または「[単一アカウント内のポリシーを評価する](#)」を参照してください。

- [ListPoliciesGrantingServiceAccess](#)

## の最終アクセス情報の表示

AWS Organizations のサービスの最終アクセス情報を表示するには、IAM コンソール、AWS CLI、または AWS API を使用します。データ、必要なアクセス許可、トラブルシューティング、およびサポートされているリージョンに関する重要な情報については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

IAM コンソールにサインインする場合、AWS Organizations 管理アカウントの認証情報を使用して、組織内の任意のエンティティの情報を表示できます。Organizations エンティティには、組織ルート、組織ユニット (OU)、およびアカウントが含まれます。IAM コンソールを使用して、組織のサービスコントロールポリシー (SCP) の情報を表示することもできます。IAM は、エンティティに適用される SCP によって許可されているサービスのリストを表示します。サービスごとに、選択した IAM エンティティまたはエンティティの子の最新のアカウントアクティビティ情報を表示できます。

AWS CLI または AWS API を管理アカウント認証情報とともに使用した場合、組織のすべてのエンティティまたはポリシーのレポートを生成できます。エンティティに関するプログラムによるレポートには、エンティティに適用される SCP によって許可されているサービスのリストが含まれます。サービスごとに、このレポートには、指定した IAM エンティティまたはエンティティのサブツリーでのアカウントの最新のアクティビティが含まれます。

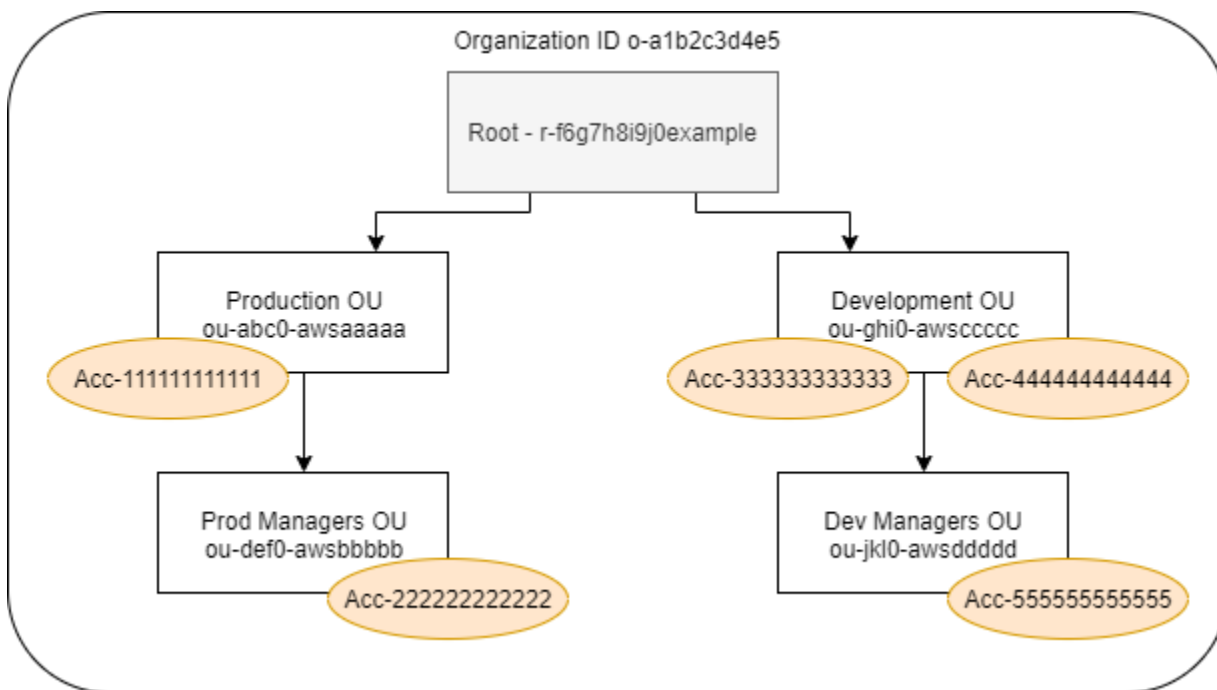
ポリシーに関するプログラムによるレポートを生成する場合、Organizations エンティティを指定する必要があります。このレポートには、指定した SCP によって許可されているサービスのリストが含まれます。サービスごとに、このレポートには、そのポリシーによってアクセス権限が付与されているエンティティまたはエンティティの子での最新のアカウントアクティビティが含まれます。詳細については、「[aws iam generate-organizations-access-report](#)」または「[GenerateOrganizationsAccessReport](#)」を参照してください。

レポートを表示する前に、管理アカウントの要件および情報、レポート期間、レポート対象のエンティティ、評価対象のポリシータイプをご確認ください。詳細については、「[the section called “最終アクセス情報についての主要事項”](#)」を参照してください。

## AWS Organizations エンティティパスを理解する

AWS CLI または AWS API を使用して AWS Organizations アクセスレポートを生成する場合は、エンティティパスを指定する必要があります。パスとは、Organizations エンティティの構造をテキストで表記したものです。

組織の既知の構造を使用して、エンティティパスを構築できます。たとえば、AWS Organizations に次のような組織構造があるとします。



[Dev Managers (開発マネージャー)] の OU のパスは、組織の ID、ルート、および OU までのパスにあるすべての OU を使用して構築されます。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-ghi0-awsccccc/ou-jkl0-awsdddd/
```

[Production (本番稼働用)] OU のアカウントのパスは、組織の ID、ルート、OU、およびアカウント番号を使用して構築されます。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-abc0-awsaaaa/111111111111/
```

### Note

組織 ID はグローバルに一意ですが、OU ID とルート ID は組織内でのみ一意です。これは、2 つの組織が同じ組織 ID を共有しないことを意味します。ただし、別の組織が自分と同じ ID を持つ OU またはルートを持っている可能性があります。OU またはルートを指定するときは、必ず組織 ID を含めることをお勧めします。

## Organizations 情報の表示 (コンソール)

IAM コンソールを使用して、ルート、OU、アカウント、またはポリシーのサービスの最終アクセス情報を表示できます。

### ルートの情報を表示するには (コンソール)

1. AWS Management Console Organizations 管理アカウントの認証情報を使用して にサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access Reports] (レポートへのアクセス) セクションの下にあるナビゲーションペインで、[Organization activity] (組織アクティビティ) を選択します。
3. [Organization activity (組織アクティビティ)] ページで、[ルート] を選択します。
4. [Details and activity] (詳細およびアクティビティ) タブで、[Service access report] (サービスアクセスレポート) セクションを表示します。情報には、ルートに直接アタッチされているポリシーによって許可されているサービスのリストが含まれます。この情報は、サービスに最後にアクセスしたアカウントとその時間が示されます。サービスにアクセスしたプリンシパルの詳細については、そのアカウントの管理者としてサインインし、[IAM サービスの最終アクセス情報を表示します](#)。
5. [Attached SCPs (アタッチされた SCP)] タブを選択し、ルートにアタッチされているサービスコントロールポリシー (SCP) のリストを表示します。に、各ポリシーがアタッチされているターゲットエンティティの数が示されます。IAM は各ポリシーがアタッチされているターゲットエンティティの数を表示できます。この情報を使用して確認する SCP を決定します。
6. SCP の名前を選択し、ポリシーで許可されているすべてのサービスを表示します。サービスごとに、サービスに最後にアクセスしたアカウントとその時間を表示します。
7. [Edit in AWS Organizations] を選択して、追加の詳細を標示し、Organizations コンソールで SCP を編集します。詳細については、AWS Organizations ユーザーガイドの「[SCP の更新](#)」を参照してください。

### OU またはアカウントの情報を表示するには (コンソール)

1. Organizations 管理アカウントの認証情報を使用して AWS Management Console にサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access Reports] (レポートへのアクセス) セクションの下にあるナビゲーションペインで、[Organization activity] (組織アクティビティ) を選択します。
3. [Organization activity (組織アクティビティ)] ページで、組織の構造を展開します。管理アカウントを除き、表示する OU またはアカウントの名前を選択します。

4. [Details and activity] (詳細およびアクティビティ) タブで、[Service access report] (サービスアクセスレポート) セクションを表示します。情報には、OU またはアカウントと そのすべての親にアタッチされている SCP によって許可されているサービスのリストが含まれます。この情報は、サービスに最後にアクセスしたアカウントとその時間が示されます。サービスにアクセスしたプリンシパルの詳細については、そのアカウントの管理者としてサインインし、[IAM サービスの最終アクセス情報](#)を表示します。
5. [Attached SCPs (アタッチされた SCP)] タブを選択し、OU またはアカウントに直接アタッチされているサービスコントロールポリシー (SCP) のリストを表示します。IAM は各ポリシーがアタッチされているターゲットエンティティの数を表示します。この情報を使用して確認する SCP を決定します。
6. SCP の名前を選択し、ポリシーで許可されているすべてのサービスを表示します。サービスごとに、サービスに最後にアクセスしたアカウントとその時間を表示します。
7. [Edit in AWS Organizations] を選択して、追加の詳細を標示し、Organizations コンソールで SCP を編集します。詳細については、AWS Organizations ユーザーガイドの「[SCP の更新](#)」を参照してください。

#### 管理アカウントの情報を表示するには (コンソール)

1. AWS Management Console Organizations 管理アカウントの認証情報を使用して にサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access Reports] (レポートへのアクセス) セクションの下にあるナビゲーションペインで、[Organization activity] (組織アクティビティ) を選択します。
3. [Organization activity (組織アクティビティ)] ページで、組織の構造を展開して管理アカウントの名前を選択します。
4. [Details and activity] (詳細およびアクティビティ) タブで、[Service access report] (サービスアクセスレポート) セクションを表示します。情報には、すべての AWS サービスのリストが含まれます。管理アカウントは SCP によって制限されません。情報に、アカウントがサービスに最後にアクセスしたかどうかとその時間が示されます。サービスにアクセスしたプリンシパルの詳細については、そのアカウントの管理者としてサインインし、[IAM サービスの最終アクセス情報](#)を表示します。
5. アカウントが管理アカウントであるため、[Attached SCPs (アタッチされた SCP)] タブを選択して、アタッチされた SCP がないことを確認します。

## ポリシーの情報を表示するには (コンソール)

1. AWS Management Console Organizations 管理アカウントの認証情報を使用してサインインし、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access Reports (レポートへのアクセス)] セクションの下にあるナビゲーションペインで、[Service control policies (SCPs) (サービスコントロールポリシー (SCP))] を選択します。
3. [Service control policies (SCPs) (サービスコントロールポリシー (SCP))] ページで、組織内のポリシーのリストを表示します。各ポリシーがアタッチされているターゲットエンティティの数を表示できます。
4. SCP の名前を選択し、ポリシーで許可されているすべてのサービスを表示します。サービスごとに、サービスに最後にアクセスしたアカウントとその時間を表示します。
5. [Edit in AWS Organizations] を選択して、追加の詳細を標示し、Organizations コンソールで SCP を編集します。詳細については、AWS Organizations ユーザーガイドの「[SCP の更新](#)」を参照してください。

## Organizations 情報の表示 (AWS CLI)

AWS CLI を使用して、Organizations ルート、OU、アカウント、またはポリシーのサービスの最終アクセス情報を取得できます。

## Organizations サービスの最終アクセス情報を表示するには (AWS CLI)

1. 必要な IAM および Organizations のアクセス権限がある Organizations 管理アカウント認証情報を使用して、ルートに対して SCP が有効になっていることを確認します。詳細については、「[最終アクセス情報についての主要事項](#)」を参照してください。
2. レポートを生成します。リクエストには、レポートが必要な IAM エンティティ (ルート、OU、またはアカウント) のパスを含める必要があります。必要に応じて、`organization-policy-id` パラメータを含めて、特定のポリシーのレポートを表示できます。コマンドにより、`job-id` が返されます。これを `get-organizations-access-report` コマンドで使用してジョブが完了するまで `job-status` をモニタリングできます。
  - [aws iam generate-organizations-access-report](#)
3. 前のステップの `job-id` パラメータを使用して、レポートに関する詳細を取得します。
  - [aws iam get-organizations-access-report](#)



このコマンドにより、エンティティメンバーがアクセスできるサービスのリストが返されます。サービスごとに、このコマンドにより、エンティティメンバーが最後に試行した日時とアカウントのエンティティパスが返されます。また、アクセス可能なサービスの総数とアクセスされなかったサービスの数も返されます。オプションの `organizations-policy-id` パラメータを指定した場合、アクセス可能なサービスは指定したポリシーによって許可されたものです。

## Organizations 情報の表示 (AWS API)

AWS API を使用して、Organizations ルート、OU、アカウント、またはポリシーのサービスの最終アクセス情報を取得できます。

## Organizations サービスの最終アクセス情報を表示するには (AWS API)

1. 必要な および のアクセス権限がある 管理アカウント認証情報を使用して、ルートに対して SCP が有効になっていることを確認します。詳細については、「[最終アクセス情報についての主要事項](#)」を参照してください。
2. レポートを生成します。リクエストには、レポートが必要な IAM エンティティ (ルート、OU、またはアカウント) のパスを含める必要があります。必要に応じて、`OrganizationsPolicyId` パラメータを含めて、特定のポリシーのレポートを表示できます。オペレーションにより、`JobId` が返されます。これを `GetOrganizationsAccessReport` オペレーションで使用して、ジョブが完了するまで、`JobStatus` をモニタリングできます。
  - [GenerateOrganizationsAccessReport](#)
3. 前のステップの `JobId` パラメータを使用して、レポートに関する詳細を取得します。
  - [GetOrganizationsAccessReport](#)

このオペレーションにより、エンティティメンバーがアクセスできるサービスのリストが返されます。サービスごとに、このオペレーションにより、エンティティメンバーが最後に試行した日時とアカウントのエンティティパスが返されます。また、アクセス可能なサービスの総数とアクセスされなかったサービスの数も返されます。オプションの `OrganizationsPolicyId` パラメータを指定した場合、アクセス可能なサービスは指定したポリシーによって許可されたものです。

## 最終アクセス情報を使用するシナリオ例

最終アクセス情報を使用して、IAM エンティティまたは AWS Organizations エンティティに付与するアクセス許可に関する決定を行うことができます。詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

### Note

IAM または AWS Organizations のエンティティあるいはポリシーに関するアクセス情報を表示する前に、データのレポート期間、レポート対象のエンティティ、評価対象のポリシータイプをご確認ください。詳細については、「[the section called “最終アクセス情報についての主要事項”](#)」を参照してください。

お客様は、管理者として、会社に適したアクセシビリティと最小限のアクセス権限のバランスを取る必要があります。

### 情報を使用した IAM グループのアクセス許可の制限

最終アクセス情報を使用して、ユーザーが必要とするサービスのみを含むように IAM グループのアクセス許可を制限することができます。この方法は、サービスレベルで[最小限の権限を付与する](#)上で重要なステップです。

たとえば、Paulo Santos は、Example Corp のAWS ユーザー権限の定義を担当する管理者です。この会社は AWS の使用を開始したばかりであり、ソフトウェア開発チームは使用する AWS のサービスをまだ定義していません。Paulo は必要なサービスにのみアクセス可能なアクセス許可をチームに付与しようと考えていますが、サービスが定義されていないため、パワーユーザーのアクセス許可を一時的に付与します。その後、最終アクセス情報を使用してグループのアクセス許可を制限します。

Paulo は、次の JSON テキストを使用して、ExampleDevelopment という名前の管理ポリシーを作成します。次に、Development という名前のグループにそのポリシーをアタッチし、開発者全員をグループに追加します。

### Note

Paulo のパワーユーザーは、一部のサービスや機能を使用するために、iam:CreateServiceLinkedRole アクセス許可が必要な場合があります。彼は、このアクセス許可を追加すると、ユーザーがサービスにリンクされたロールを作成することを



許可されることを理解しています。彼は、パワーユーザーのためにこのリスクを受け入れません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessToAllServicesExceptPeopleManagement",
      "Effect": "Allow",
      "NotAction": [
        "iam:*",
        "organizations:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredIamAndOrgsActions",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:ListRoles",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

90 日後、Paulo は Development を使用して、AWS Management Console グループの[最終アクセス情報を表示](#)します。グループメンバーがアクセスしたサービスのリストを表示します。先週、このユーザーがアクセスしたサービスは、AWS CloudTrail、Amazon CloudWatch Logs、Amazon EC2、AWS KMS、Amazon S3 の 5 つであることが分かりました。AWS の使用開始時は他のいくつかのサービスにもアクセスしていましたが、それ以降はアクセスしていません。

Paulo は、これらの 5 つのサービスと、IAM および Organizations の必要なアクションのみ含まれるようにポリシーのアクセス許可を変更することにしました。また、次の JSON テキストを使用して、ExampleDevelopment ポリシーを編集します。

**Note**

Paulo のパワーユーザーは、一部のサービスや機能を使用するために、iam:CreateServiceLinkedRole アクセス許可が必要な場合があります。彼は、このアクセス許可を追加すると、ユーザーがサービスにリンクされたロールを作成することを許可されることを理解しています。彼は、パワーユーザーのためにこのリスクを受け入れます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessToListedServices",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "kms:*",
        "cloudtrail:*",
        "logs:*",
        "ec2:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredIamAndOrgsActions",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:ListRoles",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

アクセス許可をさらに制限するために、Paulo は、AWS CloudTrail の [イベント履歴] でアカウントのイベントを表示します。ここでは、詳細なイベント情報を表示して、開発者が必要とするアクションとリソースのみが含まれるように、ポリシーのアクセス許可を制限することができます。詳細に

については、AWS CloudTrail CloudTrail ユーザーガイドの [CloudTrail イベント履歴でのイベントの表示](#)を参照してください。

## 情報を使用した IAM ユーザーのアクセス許可の制限

最終アクセス情報を使用して、各 IAM ユーザーのアクセス許可を制限することができます。

たとえば、Martha Rivera は、アクセス許可を管理する IT 管理者であり、AWS アクセス許可を組織内のユーザーに過剰に付与しないように管理する必要があります。定期的なセキュリティチェックの一部として、彼女はすべての IAM ユーザーのアクセス許可を見直します。これらのユーザーの中には、以前にセキュリティエンジニアのロールを担っていた、Nikhil Jayashankar という名前のアプリケーション開発者がいます。ジョブ要件が変更され、Nikhil は現在 app-dev グループと security-team グループの両方のメンバーです。app-dev グループでは、Amazon EC2、Amazon EBS、Auto Scaling、Amazon S3、Route 53、Elastic Transcoder などの複数のサービスへのアクセス許可が付与されています。以前のジョブの security-team グループでは、IAM および CloudTrail へのアクセス許可が付与されています。

管理者である Martha が IAM コンソールにサインインしてユーザーを選択後、nikhilj という名前を選択し、[アクセスアドバイザー] タブを選択します。

Martha は [最終アクセス] 列を確認し、Nikhil が最近、IAM、CloudTrail、Route 53、Amazon Elastic Transcoder、および多数の AWS サービスにアクセスしていないことに気づきました。ニヒルは Amazon S3 にアクセスしました。Martha は、サービスのリストから [S3] を選択し、Nikhil が過去 2 週間以内にいくつかの S3 List アクションを実行したことを知ります。Martha は、Nikhil が社内のセキュリティチームのメンバーではなくなったため、IAM と CloudTrail にアクセスするビジネス上のニーズがなくなったことを確認します。

Martha は、サービスおよびアクションの最終アクセス情報に基づいて作業を行う準備ができました。ただし、前の例のグループとは異なり、nikhilj のような IAM ユーザーには、複数のポリシーが適用され、複数のグループのメンバーになることがあります。Martha は、nikhilj やその他のグループメンバーのアクセスを誤って中断しないように慎重に進める必要があります。彼女は、Nikhil に付与する必要があるアクセス権だけでなく、これらのアクセス許可が付与されている背景を確認する必要があります。

Martha は、[アクセス許可] タブを選択します。ここでは、nikhilj に直接アタッチされているポリシーと、グループからアタッチされているポリシーを確認します。各ポリシーを展開して、ポリシーの概要を確認し、Nikhil が現在使用していないサービスへのアクセスを許可しているポリシーを確認します。

- IAM – IAMFullAccess AWS 管理ポリシーは、nikhilj に直接アタッチされているほか、security-team グループにもアタッチされています。
- CloudTrail - AWSCloudTrailReadOnlyAccess AWS 管理ポリシー は、security-team グループにアタッチされています。
- Route 53 - App-Dev-Route53 カスタマー管理ポリシーは、app-dev グループにアタッチされています。
- Elastic Transcoder - App-Dev-ElasticTranscoder カスタマー管理ポリシーは、app-dev グループにアタッチされています。

Martha は、nikhilj に直接アタッチされている AWS 管理ポリシー IAMFullAccess を削除することにしました。また、Nikhil の security-team グループのメンバーシップも削除します。これらの 2 つのアクションによって、IAM および CloudTrail への不要なアクセスが削除されます。

Route 53 および Elastic Transcoder への Nikhil のアクセス許可は、app-dev グループによって付与されています。Nikhil は、これらのサービスを使用していませんが、グループの他のメンバーは使用する可能性があります。Martha は、app-dev グループの最終アクセス情報を確認し、複数のメンバーが最近 Route 53 と Amazon S3 にアクセスしたことを知ります。しかし、昨年 Elastic Transcoder にアクセスしたグループメンバーはいませんでした。彼女は、カスタマー管理ポリシー App-Dev-ElasticTranscoder をグループから削除します。

次に、カスタマー管理ポリシー App-Dev-ElasticTranscoder の最終アクセス情報を確認します。このポリシーは、その他の IAM アイデンティティのいずれにもアタッチされていないことがわかりました。彼女は社内で調査を行い、このポリシーは今後不要であることを確認して削除しました。

#### IAM リソースを削除する前に情報を使用する

IAM リソースを削除する前に、最終アクセス情報を使用して、最後にリソースを使用してから一定の時間が経過したことを確認できます。これは、ユーザー、グループ、ロール、ポリシーに適用されます。これらのアクションの詳細については、以下のトピックを参照してください。

- ユーザー – [IAM ユーザーの削除](#)
- グループ – [グループの削除](#)
- ロール – [ロールの削除](#)
- ポリシー – [管理ポリシーを削除する \(これにより、ポリシーは ID から切り離されます\)](#)

## IAM ポリシーを編集する前に情報を使用する

IAM アイデンティティ (ユーザー、グループ、ロール)、またはそのリソースに影響するポリシーを編集する前の IAM ポリシーの最終アクセス情報を確認できます。使用しているユーザーのアクセスを削除しないように、このステップは重要です。

たとえば、Arnav Desai は Example Corp の開発者および AWS 管理者です。彼のチームが AWS を使い始めたとき、IAM と組織を除くすべてのサービスへのフルアクセスを許可するパワーユーザーアクセスをすべての開発者に与えました。[最小限の権限の付与](#)の最初のステップとして、Arnav は AWS CLI を使用して、アカウントの管理ポリシーを確認しようとしています。

そのために、Arnav はまず、アイデンティティにアタッチされているアカウントのカスタマー管理ポリシーを表示します。次のコマンドを使用します。

```
aws iam list-policies --scope Local --only-attached --policy-usage-filter
PermissionsPolicy
```

レスポンスから、各ポリシーの ARN をキャプチャします。Arnav は次に、次のコマンドを使用して、各ポリシーの最終アクセス情報のレポートを生成します。

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

レスポンスから、JobId フィールドから生成したレポートの ID をキャプチャします。続いて、Arnav は、JobStatus フィールドより COMPLETED または FAILED の値が返るまで、次のコマンドをポーリングします。ジョブが失敗した場合は、そのエラーをキャプチャします。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

ジョブが COMPLETED のステータスになったら、Arnav は JSON 形式の ServicesLastAccessed 配列の内容を解析します。

```
"ServicesLastAccessed": [
  {
    "TotalAuthenticatedEntities": 1,
    "LastAuthenticated": 2018-11-01T21:24:33.222Z,
    "ServiceNamespace": "dynamodb",
    "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/IAMExampleUser",
    "ServiceName": "Amazon DynamoDB"
  },
```

```
{
  "TotalAuthenticatedEntities": 0,
  "ServiceNamespace": "ec2",
  "ServiceName": "Amazon EC2"
},
{
  "TotalAuthenticatedEntities": 3,
  "LastAuthenticated": 2018-08-25T15:29:51.156Z,
  "ServiceNamespace": "s3",
  "LastAuthenticatedEntity": "arn:aws:iam::123456789012:role/IAMExampleRole",
  "ServiceName": "Amazon S3"
}
]
```

Arnav は、この情報から、ExamplePolicy1 ポリシーによって、Amazon DynamoDB、Amazon S3、Amazon EC2 の 3 つのサービスへのアクセスが許可されていることを理解します。IAMExampleUser という名前の IAM ユーザーが、11 月 1 日に DynamoDB に最後にアクセスを試み、8 月 25 日には IAMExampleRole ロールを使用して、Amazon S3 にアクセスしようとしたユーザーがいました。昨年 Amazon S3 にアクセスを試みたエンティティは他にも 2 つあります。ただし、昨年 Amazon EC2 にアクセスしようとしたユーザーはいませんでした。

つまり、Arnav は、ポリシーから Amazon EC2 アクションを安全に削除することができます。Arnav は、このポリシーの最新の JSON ドキュメントを確認しようとしています。まず、次のコマンドを使用して、ポリシーのバージョン番号を確認する必要があります。

```
aws iam list-policy-versions --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1
```

レスポンスから、Arnav は Versions 配列の現在のデフォルトバージョン番号を収集します。続いて、次のコマンドでそのバージョン番号 (v2) を使用して、JSON ポリシードキュメントをリクエストします。

```
aws iam get-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1 --version-id v2
```

Arnav は、PolicyVersion 配列の Document フィールドで返る JSON ポリシードキュメントを保存します。Arnav は、ポリシードキュメント内で、ec2 名前空間内のアクションを検索します。ポリシーに残っている他の名前空間からのアクションがない場合、影響を受けるアイデンティティ (ユー

ザー、グループ、およびロール) からポリシーを切り離します。その後、ポリシーを削除します。この場合、ポリシーには Amazon DynamoDB サービスと Amazon S3 サービスが含まれます。したがって、Arnav は、ドキュメントから Amazon EC2 アクションを削除し、変更内容を保存します。また、次のコマンドを使用して、ポリシーをドキュメントの新しいバージョンで更新し、そのバージョンをデフォルトのポリシーバージョンとして設定します。

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1 --policy-document file://UpdatedPolicy.json --set-as-default
```

これで、ExamplePolicy1 ポリシーは更新され、不要な Amazon EC2 サービスへのアクセスは削除されました。

## その他の IAM のシナリオ

IAM リソース (ユーザー、グループ、ロール、ポリシー) によって、サービスに最後にアクセスを試みた際に関する情報は、次のタスクを完了する際に役立ちます。

- [ポリシー – 既存のカスタマー管理ポリシーまたはインラインポリシーを編集してアクセス許可を削除する](#)
- [ポリシー – インラインポリシーを削除して、管理ポリシーに変換し、削除する](#)
- [ポリシー – 既存のポリシーに明示的な拒否を追加する](#)
- [ポリシー – 管理ポリシーを ID \(ユーザー、グループ、ロール\) からデタッチする](#)
- [エンティティ – IAM エンティティ \(ユーザーまたはロール\) が持つことができる最大のアクセス許可を制御するためのアクセス許可の境界を設定する](#)
- [グループ – グループからユーザーを削除する](#)

## 情報を使用した組織単位のアクセス許可の調整

最終アクセス情報を使用して、AWS Organizations で組織単位 (OU) のアクセス許可を制限することができます。

たとえば、John Stiles は AWS Organizations 管理者です。彼は、会社の AWS アカウント でユーザーに過剰なアクセス許可を与えないようにする責任があります。定期的なセキュリティ監査の一部として、彼は自分の組織のアクセス権限を見直します。彼の Development OU には、新しい AWS サービスのテストによく使用されるアカウントが含まれています。John は、180 日間以上アクセスされていないサービスに関するレポートを定期的に見直すことに決定します。その後、彼は OU のメンバーがそれらのサービスにアクセスするためのアクセス許可を削除します。



John は自分の管理アカウント認証情報を使用して IAM コンソールにサインインします。IAM コンソールで、彼は Development OU の Organizations データを見つけます。彼は、サービスアクセスレポートテーブルを見直し、希望期間の 180 日間以上アクセスされていない 2 つの AWS サービスがあることを確認します。彼は、開発チームが Amazon Lex および AWS Database Migration Service にアクセスするためのアクセス権限を追加したことを覚えています。John は、開発チームに連絡し、これらのサービスをテストするビジネスニーズがないことを確認します。

John は、最終アクセス情報に基づいて作業を行う準備ができました。彼は、[AWS Organizations での編集] を選択し、SCP が複数のエンティティにアタッチされていることが通知されます。彼は [Continue (続行)] を選択します。AWS Organizations で、彼は SCP がアタッチされている Organizations エンティティを確認するためにターゲットを見直します。すべてのエンティティは Development OU 内にあります。

John は、AWS Database Migration Service SCP の Amazon Lex および NewServiceTest アクションへのアクセスを拒否することに決定します。このアクションにより、サービスへの不要なアクセスが削除されます。

## IAM アクションの最終アクセス情報サービスとアクション

次の表は、[IAM アクションの最終アクセス情報](#)が表示される AWS サービスの一覧です。各サービスのアクションのリストについては、「サービス認証リファレンス」の「[AWS サービスのアクション、リソース、条件キー](#)」を参照してください。

サービス	サービスプレフィックス
<a href="#">AWS Identity and Access Management Access Analyzer</a>	access-analyzer
<a href="#">AWS Account Management</a>	アカウント
<a href="#">AWS Certificate Manager</a>	acm
<a href="#">Amazon Managed Workflows for Apache Airflow</a>	airflow
<a href="#">AWS Amplify</a>	amplify
<a href="#">AWS Amplify UI Builder</a>	amplifyuibuilder
<a href="#">Amazon アプリインテグレーション</a>	app-integrations



サービス	サービスプレフィックス
<a href="#">AWS AppConfig</a>	appconfig
<a href="#">Amazon AppFlow</a>	appflow
<a href="#">AWS Application Cost Profiler</a>	application-cost-profiler
<a href="#">Amazon CloudWatch Application Insights</a>	applicationinsights
<a href="#">AWS App Mesh</a>	appmesh
<a href="#">Amazon AppStream 2.0</a>	appstream
<a href="#">AWS AppSync</a>	appsync
<a href="#">Amazon Managed Service for Prometheus</a>	aps
<a href="#">Amazon Athena</a>	athena
<a href="#">AWS Audit Manager</a>	auditmanager
<a href="#">AWS Auto Scaling</a>	オートスケーリング
<a href="#">AWS Marketplace</a>	aws-marketplace
<a href="#">AWS Backup</a>	バックアップ
<a href="#">AWS Batch</a>	バッチ
<a href="#">Amazon Braket</a>	braket
<a href="#">AWS Budgets</a>	予算
<a href="#">AWS Cloud9</a>	cloud9
<a href="#">AWS CloudFormation</a>	cloudformation
<a href="#">Amazon CloudFront</a>	cloudfront

サービス	サービスプレフィックス
<a href="#">AWS CloudHSM</a>	cloudhsm
<a href="#">Amazon CloudSearch</a>	cloudsearch
<a href="#">AWS CloudTrail</a>	CloudTrail
<a href="#">Amazon CloudWatch</a>	cloudwatch
<a href="#">AWS CodeArtifact</a>	codeartifact
<a href="#">AWS CodeDeploy</a>	codedeploy
<a href="#">Amazon CodeGuru Profiler</a>	codeguru-profiler
<a href="#">Amazon CodeGuru Reviewer</a>	codeguru-reviewer
<a href="#">AWS CodePipeline</a>	codepipeline
<a href="#">AWS CodeStar</a>	codestar
<a href="#">AWS CodeStar 通知</a>	codestar-notifications
<a href="#">Amazon Cognito ID</a>	cognito-identity
<a href="#">Amazon Cognito ユーザープール</a>	cognito-idp
<a href="#">Amazon Cognito Sync</a>	cognito-sync
<a href="#">Amazon Comprehend Medical</a>	comprehendmedical
<a href="#">AWS Compute Optimizer</a>	compute-optimizer
<a href="#">AWS Config</a>	config
<a href="#">Amazon Connect</a>	接続
<a href="#">AWS Cost and Usage Report</a>	cur

サービス	サービスプレフィックス
<a href="#">AWS Glue DataBrew</a>	databrew
<a href="#">AWS Data Exchange</a>	dataexchange
<a href="#">AWS Data Pipeline</a>	datapipeline
<a href="#">DynamoDB Accelerator</a>	dax
<a href="#">AWS Device Farm</a>	devicefarm
<a href="#">Amazon DevOps Guru</a>	devops-guru
<a href="#">AWS Direct Connect</a>	directconnect
<a href="#">Amazon Data Lifecycle Manager</a>	dlm
<a href="#">AWS Database Migration Service</a>	dms
<a href="#">Amazon DocumentDB Elastic Clusters</a>	docdb-elastic
<a href="#">AWS Directory Service</a>	ds
<a href="#">Amazon DynamoDB</a>	dynamodb
<a href="#">Amazon Elastic Block Store</a>	ebs
<a href="#">Amazon Elastic Compute Cloud</a>	Ec2
<a href="#">Amazon Elastic Container Registry</a>	ecr
<a href="#">Amazon Elastic Container Registry Public</a>	ecr-public
<a href="#">Amazon Elastic Container Service</a>	Ecs
<a href="#">Amazon Elastic Kubernetes Service</a>	eks
<a href="#">Amazon Elastic Inference</a>	elastic-inference
<a href="#">Amazon ElastiCache</a>	elasticache

サービス	サービスプレフィックス
<a href="#">AWS Elastic Beanstalk</a>	elasticbeanstalk
<a href="#">Amazon Elastic File System</a>	elasticfilesystem
<a href="#">Elastic Load Balancing</a>	elasticloadbalancing
<a href="#">Amazon Elastic Transcoder</a>	elastictranscoder
<a href="#">Amazon EMR on EKS (EMR コンテナ)</a>	emr-containers
<a href="#">Amazon EMR Serverless</a>	emr-serverless
<a href="#">Amazon OpenSearch Service</a>	es
<a href="#">Amazon EventBridge</a>	events
<a href="#">Amazon CloudWatch Evidently</a>	evidently
<a href="#">Amazon FinSpace</a>	finspace
<a href="#">Amazon Data Firehose</a>	firehose
<a href="#">AWS Fault Injection Service</a>	fis
<a href="#">AWS Firewall Manager</a>	fms
<a href="#">Amazon Fraud Detector</a>	frauddetector
<a href="#">Amazon FSx</a>	fsx
<a href="#">Amazon GameLift</a>	gamelift
<a href="#">Amazon Location Service</a>	geo
<a href="#">Amazon S3 Glacier</a>	glacier
<a href="#">Amazon Managed Grafana</a>	grafana

サービス	サービスプレフィックス
<a href="#">AWS IoT Greengrass</a>	greengrass
<a href="#">AWS Ground Station</a>	groundstation
<a href="#">Amazon GuardDuty</a>	guardduty
<a href="#">AWS HealthLake</a>	healthlake
<a href="#">Amazon Honeycode</a>	honeycode
<a href="#">AWS Identity and Access Management</a>	iam
<a href="#">AWS Identity Store</a>	identitystore
<a href="#">EC2 Image Builder</a>	imagebuilder
<a href="#">Amazon Inspector Classic</a>	inspector
<a href="#">Amazon Inspector</a>	inspector2
<a href="#">AWS IoT</a>	iot
<a href="#">AWS IoT Analytics</a>	iotanalytics
<a href="#">AWS IoT Core Device Advisor</a>	iotdeviceadvisor
<a href="#">AWS IoT Events</a>	iotevents
<a href="#">AWS IoT Fleet Hub</a>	iotfleethub
<a href="#">AWS IoT SiteWise</a>	iotsitewise
<a href="#">AWS IoT TwinMaker</a>	iottwinmaker
<a href="#">AWS IoT Wireless</a>	iotwireless
<a href="#">Amazon Interactive Video Service</a>	ivs
<a href="#">Amazon Interactive Video Service Chat</a>	ivschat

サービス	サービスプレフィックス
<a href="#">Amazon Managed Streaming for Apache Kafka</a>	kafka
<a href="#">Amazon Managed Streaming for Kafka Connect</a>	kafkaconnect
<a href="#">Amazon Kendra</a>	kendra
<a href="#">Amazon Kinesis</a>	kinesis
<a href="#">Amazon Kinesis Analytics V2</a>	kinesisanalytics
<a href="#">AWS Key Management Service</a>	kms
<a href="#">AWS Lambda</a>	lambda
<a href="#">Amazon Lex</a>	lex
<a href="#">AWS License Manager Linux サブスクリプションマネージャー</a>	license-manager-linux-subscriptions
<a href="#">Amazon Lightsail</a>	lightsail
<a href="#">Amazon CloudWatch Logs</a>	ログ
<a href="#">Amazon Lookout for Equipment</a>	lookoutequipment
<a href="#">Amazon Lookout for Metrics</a>	lookoutmetrics
<a href="#">Amazon Lookout for Vision</a>	lookoutvision
<a href="#">AWS Mainframe Modernization</a>	m2
<a href="#">Amazon Managed Blockchain</a>	managedblockchain
<a href="#">AWS Elemental MediaConnect</a>	mediaconnect
<a href="#">AWS Elemental MediaConvert</a>	mediaconvert
<a href="#">AWS Elemental MediaLive</a>	medialive

サービス	サービスプレフィックス
<a href="#">AWS Elemental MediaStore</a>	mediastore
<a href="#">AWS Elemental MediaTailor</a>	mediatailor
<a href="#">Amazon MemoryDB for Redis</a>	memorydb
<a href="#">AWS Application Migration Service</a>	mgn
<a href="#">AWS Migration Hub</a>	mgh
<a href="#">AWS Migration Hub 戦略レコメンデーション</a>	migration hub-strategy
<a href="#">Amazon Pinpoint</a>	mobiletargeting
<a href="#">Amazon MQ</a>	mq
<a href="#">AWS Network Manager</a>	networkmanager
<a href="#">Amazon Nimble Studio</a>	nimble
<a href="#">AWS HealthOmics</a>	omics
<a href="#">AWS OpsWorks</a>	opsworks
<a href="#">AWS OpsWorks CM</a>	opsworks-cm
<a href="#">AWS Outposts</a>	outposts
<a href="#">AWS Organizations</a>	組織
<a href="#">AWS Panorama</a>	panorama
<a href="#">AWS Performance Insights</a>	pi
<a href="#">Amazon EventBridge パイプ</a>	pipes
<a href="#">Amazon Polly</a>	polly

サービス	サービスプレフィックス
<a href="#">Amazon Connect Customer Profiles</a>	profile
<a href="#">Amazon QLDB</a>	qldb
<a href="#">AWS Resource Access Manager</a>	ram
<a href="#">AWS ごみ箱</a>	rbin
<a href="#">Amazon Relational Database Service</a>	rds
<a href="#">Amazon Redshift</a>	redshift
<a href="#">Amazon Redshift Data API</a>	redshift-data
<a href="#">AWS Migration Hub Refactor Spaces</a>	refactor-spaces
<a href="#">Amazon Rekognition</a>	rekognition
<a href="#">AWS Resilience Hub</a>	resiliencehub
<a href="#">AWS Resource Explorer</a>	resource-explorer-2
<a href="#">AWS Resource Groups</a>	resource-groups
<a href="#">AWS RoboMaker</a>	robomaker
<a href="#">AWS Identity and Access Management Roles Anywhere</a>	rolesanywhere
<a href="#">Amazon Route 53</a>	route53
<a href="#">Amazon Route 53 Recovery コントロール</a>	route53-recovery-control-config
<a href="#">Amazon Route 53 Recovery 準備状況</a>	route53-recovery-readiness
<a href="#">Amazon Route 53 Resolver</a>	route53resolver



サービス	サービスプレフィックス
<a href="#">AWS CloudWatch RUM</a>	rum
<a href="#">Amazon Simple Storage Service</a>	s3
<a href="#">Amazon S3 on Outposts</a>	s3-outposts
<a href="#">Amazon SageMaker 地理空間機能</a>	sagemaker-geospatial
<a href="#">Savings Plans</a>	savingsplans
<a href="#">Amazon EventBridge スキーマ</a>	schemas
<a href="#">Amazon SimpleDB</a>	sdb
<a href="#">AWS Secrets Manager</a>	secretsmanager
<a href="#">AWS Security Hub</a>	securityhub
<a href="#">Amazon Security Lake</a>	securitylake
<a href="#">AWS Serverless Application Repository</a>	serverlessrepo
<a href="#">AWS Service Catalog</a>	servicecatalog
<a href="#">AWS Cloud Map</a>	servicediscovery
<a href="#">Service Quotas</a>	servicequotas
<a href="#">Amazon Simple Email Service</a>	ses
<a href="#">AWS Shield</a>	shield
<a href="#">AWS Signer</a>	signer
<a href="#">AWS SimSpace Weaver</a>	simspaceweaver
<a href="#">AWS Server Migration Service</a>	sms

サービス	サービスプレフィックス
<a href="#">Amazon Pinpoint SMS および音声サービス</a>	sms-voice
<a href="#">AWS Snowball</a>	snowball
<a href="#">Amazon Simple Queue Service</a>	sq
<a href="#">AWS Systems Manager</a>	ssm
<a href="#">AWS Systems Manager Incident Manager</a>	ssm-incidents
<a href="#">AWS Systems Manager for SAP</a>	ssm-sap
<a href="#">AWS Step Functions</a>	states
<a href="#">AWS Security Token Service</a>	sts
<a href="#">Amazon Simple Workflow Service</a>	swf
<a href="#">Amazon CloudWatch Synthetics</a>	synthetics
<a href="#">AWS Resource Groups Tagging API</a>	タグ
<a href="#">Amazon Textract</a>	textract
<a href="#">Amazon Timestream</a>	timestream
<a href="#">AWS 通信ネットワークビルダー</a>	tnb
<a href="#">Amazon Transcribe</a>	transcribe
<a href="#">AWS Transfer Family</a>	移管
<a href="#">Amazon Translate</a>	translate
<a href="#">Amazon Connect Voice ID</a>	voiceid
<a href="#">Amazon VPC Lattice</a>	vpc-lattice
<a href="#">AWS WAFV2</a>	wafv2

サービス	サービスプレフィックス
<a href="#">AWS Well-Architected Tool</a>	wellarchitected
<a href="#">Amazon Connect Wisdom</a>	wisdom
<a href="#">Amazon WorkLink</a>	worklink
<a href="#">Amazon WorkSpaces</a>	ワークスペース
<a href="#">AWS X-Ray</a>	xray

アクションの最終アクセス情報をサポートするアクション

次の表は、アクションの最終アクセス情報が入手できるアクションの一覧です。

サービスプレフィックス	アクション
access-analyzer	access-analyzer:ApplyArchiveRule
	access-analyzer:CancelPolicyGeneration
	access-analyzer:CheckAccessNotGranted
	access-analyzer:CheckNoNewAccess
	access-analyzer:CreateAccessPreview
	access-analyzer:CreateAnalyzer
	access-analyzer:CreateArchiveRule
	access-analyzer>DeleteAnalyzer
	access-analyzer>DeleteArchiveRule
	access-analyzer:GetAccessPreview
	access-analyzer:GetAnalyzedResource

サービスプレフィックス	アクション
	access-analyzer:GetAnalyzer
	access-analyzer:GetArchiveRule
	access-analyzer:GetFinding
	access-analyzer:GetGeneratedPolicy
	access-analyzer:ListAccessPreviewFindings
	access-analyzer:ListAccessPreviews
	access-analyzer:ListAnalyzedResources
	access-analyzer:ListAnalyzers
	access-analyzer:ListArchiveRules
	access-analyzer:ListFindings
	access-analyzer:ListPolicyGenerations
	access-analyzer:StartPolicyGeneration
	access-analyzer:StartResourceScan
	access-analyzer:UpdateArchiveRule
	access-analyzer:UpdateFindings
	access-analyzer:ValidatePolicy

サービスプレフィックス	アクション
アカウント	account>DeleteAlternateContact account:DisableRegion account:EnableRegion account:GetAlternateContact account:GetContactInformation account:GetRegionOptStatus account:ListRegions account:PutAlternateContact account:PutContactInformation
acm	acm>DeleteCertificate acm:DescribeCertificate acm:ExportCertificate acm:GetAccountConfiguration acm:GetCertificate acm:ImportCertificate acm:ListCertificates acm:PutAccountConfiguration acm:RenewCertificate acm:RequestCertificate acm:ResendValidationEmail acm:UpdateCertificateOptions

サービスプレフィックス	アクション
airflow	airflow:CreateCliToken airflow:CreateEnvironment airflow:CreateWebLoginToken airflow>DeleteEnvironment airflow:GetEnvironment airflow:ListEnvironments airflow:PublishMetrics airflow:UpdateEnvironment

サービスプレフィックス	アクション
amplify	amplify:CreateApp
	amplify:CreateBackendEnvironment
	amplify:CreateBranch
	amplify:CreateDeployment
	amplify:CreateDomainAssociation
	amplify:CreateWebHook
	amplify>DeleteApp
	amplify>DeleteBackendEnvironment
	amplify>DeleteBranch
	amplify>DeleteDomainAssociation
	amplify>DeleteJob
	amplify>DeleteWebHook
	amplify:GenerateAccessLogs
	amplify:GetApp
	amplify:GetArtifactUrl
	amplify:GetBackendEnvironment
	amplify:GetBranch
	amplify:GetDomainAssociation
	amplify:GetJob
	amplify:GetWebHook
	amplify:ListApps

サービスプレフィックス	アクション
	amplify:ListArtifacts
	amplify:ListBackendEnvironments
	amplify:ListBranches
	amplify:ListDomainAssociations
	amplify:ListJobs
	amplify:ListWebHooks
	amplify:StartDeployment
	amplify:StartJob
	amplify:StopJob
	amplify:UpdateApp
	amplify:UpdateBranch
	amplify:UpdateDomainAssociation
	amplify:UpdateWebHook



サービスプレフィックス	アクション
amplifyuibuilder	amplifyuibuilder:CreateComponent
	amplifyuibuilder:CreateForm
	amplifyuibuilder:CreateTheme
	amplifyuibuilder>DeleteComponent
	amplifyuibuilder>DeleteForm
	amplifyuibuilder>DeleteTheme
	amplifyuibuilder:ExportComponents
	amplifyuibuilder:ExportThemes
	amplifyuibuilder:GetCodegenJob
	amplifyuibuilder:ListCodegenJobs
	amplifyuibuilder:ListComponents
	amplifyuibuilder:ListForms
	amplifyuibuilder:ListThemes
	amplifyuibuilder:ResetMetadataFlag
	amplifyuibuilder:StartCodegenJob
	amplifyuibuilder:UpdateComponent
	amplifyuibuilder:UpdateForm
	amplifyuibuilder:UpdateTheme

サービスプレフィックス	アクション
app-integrations	app-integrations:CreateApplication
	app-integrations:CreateDataIntegration
	app-integrations:CreateEventIntegration
	app-integrations>DeleteApplication
	app-integrations>DeleteDataIntegration
	app-integrations>DeleteEventIntegration
	app-integrations:GetApplication
	app-integrations:GetDataIntegration
	app-integrations:GetEventIntegration
	app-integrations:ListApplicationAssociations
	app-integrations:ListApplications
	app-integrations:ListDataIntegrationAssociations
	app-integrations:ListDataIntegrations
	app-integrations:ListEventIntegrationAssociations
	app-integrations:ListEventIntegrations
	app-integrations:UpdateApplication
	app-integrations:UpdateDataIntegration
	app-integrations:UpdateEventIntegration

サービスプレフィックス	アクション
appconfig	appconfig:CreateApplication
	appconfig:CreateConfigurationProfile
	appconfig:CreateDeploymentStrategy
	appconfig:CreateEnvironment
	appconfig:CreateExtension
	appconfig:CreateExtensionAssociation
	appconfig:CreateHostedConfigurationVersion
	appconfig>DeleteApplication
	appconfig>DeleteConfigurationProfile
	appconfig>DeleteDeploymentStrategy
	appconfig>DeleteEnvironment
	appconfig>DeleteExtension
	appconfig>DeleteExtensionAssociation
	appconfig>DeleteHostedConfigurationVersion
	appconfig:GetApplication
	appconfig:GetConfiguration
	appconfig:GetConfigurationProfile
	appconfig:GetDeployment
	appconfig:GetDeploymentStrategy
	appconfig:GetEnvironment
appconfig:GetExtension	

サービスプレフィックス	アクション
	appconfig:GetExtensionAssociation
	appconfig:GetHostedConfigurationVersion
	appconfig:ListApplications
	appconfig:ListConfigurationProfiles
	appconfig:ListDeployments
	appconfig:ListDeploymentStrategies
	appconfig:ListEnvironments
	appconfig:ListExtensionAssociations
	appconfig:ListExtensions
	appconfig:ListHostedConfigurationVersions
	appconfig:StartDeployment
	appconfig:StopDeployment
	appconfig:UpdateApplication
	appconfig:UpdateConfigurationProfile
	appconfig:UpdateDeploymentStrategy
	appconfig:UpdateEnvironment
	appconfig:UpdateExtension
	appconfig:UpdateExtensionAssociation
	appconfig:ValidateConfiguration

サービスプレフィックス	アクション
appflow	appflow:CancelFlowExecutions
	appflow:CreateConnectorProfile
	appflow:CreateFlow
	appflow>DeleteConnectorProfile
	appflow>DeleteFlow
	appflow:DescribeConnector
	appflow:DescribeConnectorEntity
	appflow:DescribeConnectorProfiles
	appflow:DescribeConnectors
	appflow:DescribeFlow
	appflow:DescribeFlowExecutionRecords
	appflow:ListConnectorEntities
	appflow:ListConnectors
	appflow:ListFlows
	appflow:RegisterConnector
	appflow:ResetConnectorMetadataCache
	appflow:StartFlow
	appflow:StopFlow
	appflow:UnRegisterConnector
	appflow:UpdateConnectorProfile
	appflow:UpdateConnectorRegistration

サービスプレフィックス	アクション
	appflow:UpdateFlow
application-cost-profiler	application-cost-profiler:DeleteReportDefinition application-cost-profiler:GetReportDefinition application-cost-profiler:ImportApplicationUsage application-cost-profiler:ListReportDefinitions application-cost-profiler:PutReportDefinition application-cost-profiler:UpdateReportDefinition

サービスプレフィックス	アクション
applicationinsights	applicationinsights:AddWorkload
	applicationinsights:CreateApplication
	applicationinsights:CreateComponent
	applicationinsights:CreateLogPattern
	applicationinsights>DeleteApplication
	applicationinsights>DeleteComponent
	applicationinsights>DeleteLogPattern
	applicationinsights:DescribeApplication
	applicationinsights:DescribeComponent
	applicationinsights:DescribeComponentConfiguration
	applicationinsights:DescribeComponentConfigurationRecommendation
	applicationinsights:DescribeLogPattern
	applicationinsights:DescribeObservation
	applicationinsights:DescribeProblem
	applicationinsights:DescribeProblemObservations
	applicationinsights:DescribeWorkload
	applicationinsights>ListApplications
	applicationinsights>ListComponents
	applicationinsights>ListConfigurationHistory
	applicationinsights>ListLogPatterns

サービスプレフィックス	アクション
	applicationinsights:ListLogPatternSets
	applicationinsights:ListProblems
	applicationinsights:ListWorkloads
	applicationinsights:RemoveWorkload
	applicationinsights:UpdateApplication
	applicationinsights:UpdateComponent
	applicationinsights:UpdateComponentConfiguration
	applicationinsights:UpdateLogPattern
	applicationinsights:UpdateWorkload



サービスプレフィックス	アクション
appmesh	appmesh:CreateGatewayRoute
	appmesh:CreateMesh
	appmesh:CreateRoute
	appmesh:CreateVirtualGateway
	appmesh:CreateVirtualNode
	appmesh:CreateVirtualRouter
	appmesh:CreateVirtualService
	appmesh>DeleteGatewayRoute
	appmesh>DeleteMesh
	appmesh>DeleteRoute
	appmesh>DeleteVirtualGateway
	appmesh>DeleteVirtualNode
	appmesh>DeleteVirtualRouter
	appmesh>DeleteVirtualService
	appmesh:DescribeGatewayRoute
	appmesh:DescribeMesh
	appmesh:DescribeRoute
	appmesh:DescribeVirtualGateway
	appmesh:DescribeVirtualNode
	appmesh:DescribeVirtualRouter
appmesh:DescribeVirtualService	

サービスプレフィックス	アクション
	appmesh:ListGatewayRoutes
	appmesh:ListMeshes
	appmesh:ListRoutes
	appmesh:ListVirtualGateways
	appmesh:ListVirtualNodes
	appmesh:ListVirtualRouters
	appmesh:ListVirtualServices
	appmesh:StreamAggregatedResources
	appmesh:UpdateGatewayRoute
	appmesh:UpdateMesh
	appmesh:UpdateRoute
	appmesh:UpdateVirtualGateway
	appmesh:UpdateVirtualNode
	appmesh:UpdateVirtualRouter
	appmesh:UpdateVirtualService

サービスプレフィックス	アクション
appstream	appstream:AssociateAppBlockBuilderAppBlock
	appstream:AssociateApplicationFleet
	appstream:AssociateApplicationToEntitlement
	appstream:AssociateFleet
	appstream:BatchAssociateUserStack
	appstream:BatchDisassociateUserStack
	appstream:CopyImage
	appstream:CreateAppBlock
	appstream:CreateAppBlockBuilder
	appstream:CreateAppBlockBuilderStreamingURL
	appstream:CreateApplication
	appstream:CreateDirectoryConfig
	appstream:CreateEntitlement
	appstream:CreateFleet
	appstream:CreateImageBuilder
	appstream:CreateImageBuilderStreamingURL
	appstream:CreateStack
	appstream:CreateStreamingURL
	appstream:CreateUpdatedImage
	appstream:CreateUsageReportSubscription
	appstream:CreateUser

サービスプレフィックス	アクション
	appstream:DeleteAppBlock
	appstream:DeleteAppBlockBuilder
	appstream:DeleteApplication
	appstream:DeleteDirectoryConfig
	appstream:DeleteEntitlement
	appstream:DeleteFleet
	appstream:DeleteImage
	appstream:DeleteImageBuilder
	appstream:DeleteImagePermissions
	appstream:DeleteStack
	appstream:DeleteUsageReportSubscription
	appstream:DeleteUser
	appstream:DescribeAppBlockBuilderAppBlockAssociations
	appstream:DescribeAppBlockBuilders
	appstream:DescribeAppBlocks
	appstream:DescribeApplicationFleetAssociations
	appstream:DescribeApplications
	appstream:DescribeDirectoryConfigs
	appstream:DescribeEntitlements
	appstream:DescribeFleets
	appstream:DescribeImageBuilders

サービスプレフィックス	アクション
	appstream:DescribeImagePermissions
	appstream:DescribeImages
	appstream:DescribeSessions
	appstream:DescribeStacks
	appstream:DescribeUsageReportSubscriptions
	appstream:DescribeUsers
	appstream:DescribeUserStackAssociations
	appstream:DisableUser
	appstream:DisassociateAppBlockBuilderAppBlock
	appstream:DisassociateApplicationFleet
	appstream:DisassociateApplicationFromEntitlement
	appstream:DisassociateFleet
	appstream:EnableUser
	appstream:ExpireSession
	appstream:ListAssociatedFleets
	appstream:ListAssociatedStacks
	appstream:ListEntitledApplications
	appstream:StartAppBlockBuilder
	appstream:StartFleet
	appstream:StartImageBuilder
	appstream:StopAppBlockBuilder

サービスプレフィックス	アクション
	appstream:StopFleet
	appstream:StopImageBuilder
	appstream:UpdateAppBlockBuilder
	appstream:UpdateApplication
	appstream:UpdateDirectoryConfig
	appstream:UpdateEntitlement
	appstream:UpdateFleet
	appstream:UpdateImagePermissions
	appstream:UpdateStack

サービスプレフィックス	アクション
appsync	appsync:AssociateApi
	appsync:AssociateMergedGraphQLApi
	appsync:AssociateSourceGraphQLApi
	appsync:CreateApiCache
	appsync:CreateApiKey
	appsync:CreateDataSource
	appsync:CreateDomainName
	appsync:CreateFunction
	appsync:CreateGraphQLApi
	appsync:CreateResolver
	appsync:CreateType
	appsync>DeleteApiCache
	appsync>DeleteApiKey
	appsync>DeleteDataSource
	appsync>DeleteDomainName
	appsync>DeleteFunction
	appsync>DeleteGraphQLApi
	appsync>DeleteResolver
	appsync>DeleteType
	appsync:DisassociateApi
appsync:DisassociateMergedGraphQLApi	

サービスプレフィックス	アクション
	appsync:DisassociateSourceGraphQLApi
	appsync:EvaluateCode
	appsync:EvaluateMappingTemplate
	appsync:FlushApiCache
	appsync:GetApiAssociation
	appsync:GetApiCache
	appsync:GetDataSource
	appsync:GetDataSourceIntrospection
	appsync:GetDomainName
	appsync:GetFunction
	appsync:GetGraphQLApi
	appsync:GetGraphQLApiEnvironmentVariables
	appsync:GetIntrospectionSchema
	appsync:GetResolver
	appsync:GetSchemaCreationStatus
	appsync:GetSourceApiAssociation
	appsync:GetType
	appsync:ListApiKeys
	appsync:ListDataSources
	appsync:ListDomainNames
	appsync:ListFunctions



サービスプレフィックス	アクション
	appsync:ListGraphQLApis
	appsync:ListResolvers
	appsync:ListResolversByFunction
	appsync:ListSourceApiAssociations
	appsync:ListTypes
	appsync:ListTypesByAssociation
	appsync:PutGraphQLApiEnvironmentVariables
	appsync:StartDataSourceIntrospection
	appsync:StartSchemaCreation
	appsync:StartSchemaMerge
	appsync:UpdateApiCache
	appsync:UpdateApiKey
	appsync:UpdateDataSource
	appsync:UpdateDomainName
	appsync:UpdateFunction
	appsync:UpdateGraphQLApi
	appsync:UpdateResolver
	appsync:UpdateSourceApiAssociation
	appsync:UpdateType

サービスプレフィックス	アクション
aps	aps:CreateAlertManagerDefinition
	aps:CreateLoggingConfiguration
	aps:CreateRuleGroupsNamespace
	aps:CreateScraper
	aps:CreateWorkspace
	aps>DeleteAlertManagerDefinition
	aps>DeleteLoggingConfiguration
	aps>DeleteRuleGroupsNamespace
	aps>DeleteScraper
	aps>DeleteWorkspace
	aps:DescribeAlertManagerDefinition
	aps:DescribeLoggingConfiguration
	aps:DescribeRuleGroupsNamespace
	aps:DescribeScraper
	aps:DescribeWorkspace
	aps:GetDefaultScraperConfiguration
	aps:ListRuleGroupsNamespaces
	aps:ListScrapers
	aps:ListWorkspaces
	aps:PutAlertManagerDefinition
	aps:PutRuleGroupsNamespace

サービスプレフィックス	アクション
	aps:UpdateLoggingConfiguration
	aps:UpdateWorkspaceAlias

サービスプレフィックス	アクション
athena	athena:BatchGetNamedQuery
	athena:BatchGetPreparedStatement
	athena:BatchGetQueryExecution
	athena:CancelCapacityReservation
	athena:CreateCapacityReservation
	athena:CreateDataCatalog
	athena:CreateNamedQuery
	athena:CreateNotebook
	athena:CreatePreparedStatement
	athena:CreatePresignedNotebookUrl
	athena:CreateWorkGroup
	athena>DeleteCapacityReservation
	athena>DeleteDataCatalog
	athena>DeleteNamedQuery
	athena>DeleteNotebook
	athena>DeletePreparedStatement
	athena>DeleteWorkGroup
	athena:ExportNotebook
	athena:GetCalculationExecution
	athena:GetCalculationExecutionCode
	athena:GetCalculationExecutionStatus

サービスプレフィックス	アクション
	athena:GetCapacityAssignmentConfiguration
	athena:GetCapacityReservation
	athena:GetDatabase
	athena:GetDataCatalog
	athena:GetNamedQuery
	athena:GetNotebookMetadata
	athena:GetPreparedStatement
	athena:GetQueryExecution
	athena:GetQueryResults
	athena:GetQueryResultsStream
	athena:GetQueryRuntimeStatistics
	athena:GetSession
	athena:GetSessionStatus
	athena:GetTableMetadata
	athena:GetWorkGroup
	athena:ImportNotebook
	athena:ListApplicationDPUSizes
	athena:ListCalculationExecutions
	athena:ListCapacityReservations
	athena:ListDatabases
	athena:ListDataCatalogs

サービスプレフィックス	アクション
	athena:ListEngineVersions
	athena:ListExecutors
	athena:ListNamedQueries
	athena:ListNotebookMetadata
	athena:ListNotebookSessions
	athena:ListPreparedStatements
	athena:ListQueryExecutions
	athena:ListSessions
	athena:ListTableMetadata
	athena:ListWorkGroups
	athena:PutCapacityAssignmentConfiguration
	athena:StartCalculationExecution
	athena:StartQueryExecution
	athena:StartSession
	athena:StopCalculationExecution
	athena:StopQueryExecution
	athena:TerminateSession
	athena:UpdateCapacityReservation
	athena:UpdateDataCatalog
	athena:UpdateNamedQuery
	athena:UpdateNotebook

サービスプレフィックス	アクション
	athena:UpdateNotebookMetadata
	athena:UpdatePreparedStatement
	athena:UpdateWorkGroup

サービスプレフィックス	アクション
auditmanager	auditmanager:AssociateAssessmentReportEvidenceFolder
	auditmanager:BatchAssociateAssessmentReportEvidence
	auditmanager:BatchCreateDelegationByAssessment
	auditmanager:BatchDeleteDelegationByAssessment
	auditmanager:BatchDisassociateAssessmentReportEvidence
	auditmanager:BatchImportEvidenceToAssessmentControl
	auditmanager:CreateAssessment
	auditmanager:CreateAssessmentFramework
	auditmanager:CreateAssessmentReport
	auditmanager:CreateControl
	auditmanager>DeleteAssessment
	auditmanager>DeleteAssessmentFramework
	auditmanager>DeleteAssessmentFrameworkShare
	auditmanager>DeleteAssessmentReport
	auditmanager>DeleteControl
	auditmanager:DeregisterAccount
	auditmanager:DeregisterOrganizationAdminAccount
	auditmanager:DisassociateAssessmentReportEvidenceFolder
	auditmanager:GetAccountStatus
	auditmanager:GetAssessment
	auditmanager:GetAssessmentFramework



サービスプレフィックス	アクション
	<code>auditmanager:GetAssessmentReportUrl</code>
	<code>auditmanager:GetChangeLogs</code>
	<code>auditmanager:GetControl</code>
	<code>auditmanager:GetDelegations</code>
	<code>auditmanager:GetEvidence</code>
	<code>auditmanager:GetEvidenceByEvidenceFolder</code>
	<code>auditmanager:GetEvidenceFileUploadUrl</code>
	<code>auditmanager:GetEvidenceFolder</code>
	<code>auditmanager:GetEvidenceFoldersByAssessment</code>
	<code>auditmanager:GetEvidenceFoldersByAssessmentControl</code>
	<code>auditmanager:GetInsights</code>
	<code>auditmanager:GetInsightsByAssessment</code>
	<code>auditmanager:GetOrganizationAdminAccount</code>
	<code>auditmanager:GetServicesInScope</code>
	<code>auditmanager:GetSettings</code>
	<code>auditmanager:ListAssessmentControlInsightsByControlDomain</code>
	<code>auditmanager:ListAssessmentFrameworks</code>
	<code>auditmanager:ListAssessmentFrameworkShareRequests</code>
	<code>auditmanager:ListAssessmentReports</code>
	<code>auditmanager:ListAssessments</code>
	<code>auditmanager:ListControlDomainInsights</code>

サービスプレフィックス	アクション
	<code>auditmanager:ListControlDomainInsightsByAssessment</code>
	<code>auditmanager:ListControlInsightsByControlDomain</code>
	<code>auditmanager:ListControls</code>
	<code>auditmanager:ListKeywordsForDataSource</code>
	<code>auditmanager:ListNotifications</code>
	<code>auditmanager:RegisterAccount</code>
	<code>auditmanager:RegisterOrganizationAdminAccount</code>
	<code>auditmanager:StartAssessmentFrameworkShare</code>
	<code>auditmanager:UpdateAssessment</code>
	<code>auditmanager:UpdateAssessmentControl</code>
	<code>auditmanager:UpdateAssessmentControlSetStatus</code>
	<code>auditmanager:UpdateAssessmentFramework</code>
	<code>auditmanager:UpdateAssessmentFrameworkShare</code>
	<code>auditmanager:UpdateAssessmentStatus</code>
	<code>auditmanager:UpdateControl</code>
	<code>auditmanager:UpdateSettings</code>
	<code>auditmanager:ValidateAssessmentReportIntegrity</code>

サービスプレフィックス	アクション
オートスケーリング	autoscaling:AttachInstances
	autoscaling:AttachLoadBalancers
	autoscaling:AttachLoadBalancerTargetGroups
	autoscaling:AttachTrafficSources
	autoscaling:BatchDeleteScheduledAction
	autoscaling:BatchPutScheduledUpdateGroupAction
	autoscaling:CancelInstanceRefresh
	autoscaling:CompleteLifecycleAction
	autoscaling>CreateAutoScalingGroup
	autoscaling>CreateLaunchConfiguration
	autoscaling>DeleteAutoScalingGroup
	autoscaling>DeleteLaunchConfiguration
	autoscaling>DeleteLifecycleHook
	autoscaling>DeleteNotificationConfiguration
	autoscaling>DeletePolicy
	autoscaling>DeleteScheduledAction
	autoscaling>DeleteWarmPool
	autoscaling:DescribeAccountLimits
	autoscaling:DescribeAdjustmentTypes
	autoscaling:DescribeAutoScalingGroups
	autoscaling:DescribeAutoScalingInstances

サービスプレフィックス	アクション
	autoscaling:DescribeAutoScalingNotificationTypes
	autoscaling:DescribeInstanceRefreshes
	autoscaling:DescribeLaunchConfigurations
	autoscaling:DescribeLifecycleHooks
	autoscaling:DescribeLifecycleHookTypes
	autoscaling:DescribeLoadBalancers
	autoscaling:DescribeLoadBalancerTargetGroups
	autoscaling:DescribeMetricCollectionTypes
	autoscaling:DescribeNotificationConfigurations
	autoscaling:DescribePolicies
	autoscaling:DescribeScalingActivities
	autoscaling:DescribeScalingProcessTypes
	autoscaling:DescribeScheduledActions
	autoscaling:DescribeTerminationPolicyTypes
	autoscaling:DescribeTrafficSources
	autoscaling:DescribeWarmPool
	autoscaling:DetachInstances
	autoscaling:DetachLoadBalancers
	autoscaling:DetachLoadBalancerTargetGroups
	autoscaling:DetachTrafficSources
	autoscaling:DisableMetricsCollection

サービスプレフィックス	アクション
	autoscaling:EnableMetricsCollection
	autoscaling:EnterStandby
	autoscaling:ExecutePolicy
	autoscaling:ExitStandby
	autoscaling:GetPredictiveScalingForecast
	autoscaling:PutLifecycleHook
	autoscaling:PutNotificationConfiguration
	autoscaling:PutScalingPolicy
	autoscaling:PutScheduledUpdateGroupAction
	autoscaling:PutWarmPool
	autoscaling:RecordLifecycleActionHeartbeat
	autoscaling:ResumeProcesses
	autoscaling:RollbackInstanceRefresh
	autoscaling:SetDesiredCapacity
	autoscaling:SetInstanceHealth
	autoscaling:SetInstanceProtection
	autoscaling:StartInstanceRefresh
	autoscaling:SuspendProcesses
	autoscaling:TerminateInstanceInAutoScalingGroup
	autoscaling:UpdateAutoScalingGroup
aws-marketplace	aws-marketplace:GetEntitlements

サービスプレフィックス	アクション
バックアップ	backup:CancelLegalHold
	backup:CreateBackupPlan
	backup:CreateBackupSelection
	backup:CreateBackupVault
	backup:CreateFramework
	backup:CreateLegalHold
	backup:CreateLogicallyAirGappedBackupVault
	backup:CreateReportPlan
	backup:CreateRestoreTestingPlan
	backup:CreateRestoreTestingSelection
	backup>DeleteBackupPlan
	backup>DeleteBackupSelection
	backup>DeleteBackupVault
	backup>DeleteBackupVaultAccessPolicy
	backup>DeleteBackupVaultLockConfiguration
	backup>DeleteBackupVaultNotifications
	backup>DeleteFramework
	backup>DeleteRecoveryPoint
	backup>DeleteReportPlan
	backup>DeleteRestoreTestingPlan
	backup>DeleteRestoreTestingSelection

サービスプレフィックス	アクション
	backup:DescribeBackupJob
	backup:DescribeBackupVault
	backup:DescribeCopyJob
	backup:DescribeFramework
	backup:DescribeGlobalSettings
	backup:DescribeProtectedResource
	backup:DescribeRecoveryPoint
	backup:DescribeRegionSettings
	backup:DescribeReportJob
	backup:DescribeReportPlan
	backup:DescribeRestoreJob
	backup:DisassociateRecoveryPoint
	backup:DisassociateRecoveryPointFromParent
	backup:ExportBackupPlanTemplate
	backup:GetBackupPlan
	backup:GetBackupPlanFromJSON
	backup:GetBackupPlanFromTemplate
	backup:GetBackupSelection
	backup:GetBackupVaultAccessPolicy
	backup:GetBackupVaultNotifications
	backup:GetLegalHold

サービスプレフィックス	アクション
	backup:GetRecoveryPointRestoreMetadata
	backup:GetRestoreJobMetadata
	backup:GetRestoreTestingInferredMetadata
	backup:GetRestoreTestingPlan
	backup:GetRestoreTestingSelection
	backup:GetSupportedResourceTypes
	backup:ListBackupJobs
	backup:ListBackupJobSummaries
	backup:ListBackupPlans
	backup:ListBackupPlanTemplates
	backup:ListBackupPlanVersions
	backup:ListBackupSelections
	backup:ListBackupVaults
	backup:ListCopyJobs
	backup:ListCopyJobSummaries
	backup:ListFrameworks
	backup:ListLegalHolds
	backup:ListProtectedResources
	backup:ListRecoveryPointsByBackupVault
	backup:ListRecoveryPointsByLegalHold
	backup:ListRecoveryPointsByResource



サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>backup:ListReportJobs</li><li>backup:ListReportPlans</li><li>backup:ListRestoreJobs</li><li>backup:ListRestoreJobsByProtectedResource</li><li>backup:ListRestoreJobSummaries</li><li>backup:ListRestoreTestingPlans</li><li>backup:ListRestoreTestingSelections</li><li>backup:PutBackupVaultAccessPolicy</li><li>backup:PutBackupVaultLockConfiguration</li><li>backup:PutBackupVaultNotifications</li><li>backup:PutRestoreValidationResult</li><li>backup:StartBackupJob</li><li>backup:StartCopyJob</li><li>backup:StartReportJob</li><li>backup:StartRestoreJob</li><li>backup:StopBackupJob</li><li>backup:UpdateBackupPlan</li><li>backup:UpdateFramework</li><li>backup:UpdateGlobalSettings</li><li>backup:UpdateRecoveryPointLifecycle</li><li>backup:UpdateRegionSettings</li></ul>

サービスプレフィックス	アクション
	backup:UpdateReportPlan backup:UpdateRestoreTestingPlan backup:UpdateRestoreTestingSelection

サービスプレフィックス	アクション
バッチ	batch:CancelJob
	batch:CreateComputeEnvironment
	batch:CreateJobQueue
	batch:CreateSchedulingPolicy
	batch>DeleteComputeEnvironment
	batch>DeleteJobQueue
	batch>DeleteSchedulingPolicy
	batch:DeregisterJobDefinition
	batch:DescribeComputeEnvironments
	batch:DescribeJobDefinitions
	batch:DescribeJobQueues
	batch:DescribeJobs
	batch:DescribeSchedulingPolicies
	batch>ListJobs
	batch>ListSchedulingPolicies
	batch:RegisterJobDefinition
	batch:SubmitJob
	batch:TerminateJob
	batch:UpdateComputeEnvironment
	batch:UpdateJobQueue
	batch:UpdateSchedulingPolicy

サービスプレフィックス	アクション
braket	braket:AcceptUserAgreement
	braket:AccessBraketFeature
	braket:CancelJob
	braket:CancelQuantumTask
	braket:CreateJob
	braket:CreateQuantumTask
	braket:GetDevice
	braket:GetJob
	braket:GetQuantumTask
	braket:GetServiceLinkedRoleStatus
	braket:GetUserAgreementStatus
	braket:SearchDevices
	braket:SearchJobs
	braket:SearchQuantumTasks

サービスプレフィックス	アクション
予算	budgets:ModifyBudget
	budgets:CreateBudgetAction
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets>DeleteBudgetAction
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets:ViewBudget
	budgets:DescribeBudgetAction
	budgets:DescribeBudgetActionHistories
	budgets:DescribeBudgetActionsForAccount
	budgets:DescribeBudgetActionsForBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ExecuteBudgetAction
	budgets:ModifyBudget
	budgets:UpdateBudgetAction

サービスプレフィックス	アクション
	budgets:ModifyBudget budgets:ModifyBudget
cloud9	cloud9:CreateEnvironmentEC2 cloud9:CreateEnvironmentMembership cloud9>DeleteEnvironment cloud9>DeleteEnvironmentMembership cloud9:DescribeEnvironmentMemberships cloud9:DescribeEnvironments cloud9:DescribeEnvironmentStatus cloud9:ListEnvironments cloud9:UpdateEnvironment cloud9:UpdateEnvironmentMembership

サービスプレフィックス	アクション
cloudformation	cloudformation:BatchDescribeTypeConfigurations
	cloudformation:CancelUpdateStack
	cloudformation:ContinueUpdateRollback
	cloudformation>CreateChangeSet
	cloudformation>CreateGeneratedTemplate
	cloudformation>CreateStack
	cloudformation>CreateStackInstances
	cloudformation>CreateStackSet
	cloudformation:DeactivateType
	cloudformation>DeleteChangeSet
	cloudformation>DeleteGeneratedTemplate
	cloudformation>DeleteStack
	cloudformation>DeleteStackInstances
	cloudformation>DeleteStackSet
	cloudformation:DeregisterType
	cloudformation:DescribeAccountLimits
	cloudformation:DescribeChangeSet
	cloudformation:DescribeChangeSetHooks
	cloudformation:DescribeGeneratedTemplate
	cloudformation:DescribeOrganizationsAccess
	cloudformation:DescribePublisher

サービスプレフィックス	アクション
	cloudformation:DescribeResourceScan
	cloudformation:DescribeStackDriftDetectionStatus
	cloudformation:DescribeStackEvents
	cloudformation:DescribeStackInstance
	cloudformation:DescribeStackResource
	cloudformation:DescribeStackResourceDrifts
	cloudformation:DescribeStackResources
	cloudformation:DescribeStacks
	cloudformation:DescribeStackSet
	cloudformation:DescribeStackSetOperation
	cloudformation:DescribeType
	cloudformation:DescribeTypeRegistration
	cloudformation:DetectStackDrift
	cloudformation:DetectStackResourceDrift
	cloudformation:DetectStackSetDrift
	cloudformation:EstimateTemplateCost
	cloudformation:ExecuteChangeSet
	cloudformation:GetGeneratedTemplate
	cloudformation:GetStackPolicy
	cloudformation:GetTemplate
	cloudformation:GetTemplateSummary



サービスプレフィックス	アクション
	<code>cloudformation:ImportStacksToStackSet</code>
	<code>cloudformation:ListChangeSets</code>
	<code>cloudformation:ListExports</code>
	<code>cloudformation:ListGeneratedTemplates</code>
	<code>cloudformation:ListImports</code>
	<code>cloudformation:ListResourceScanRelatedResources</code>
	<code>cloudformation:ListResourceScanResources</code>
	<code>cloudformation:ListResourceScans</code>
	<code>cloudformation:ListStackInstanceResourceDrifts</code>
	<code>cloudformation:ListStackInstances</code>
	<code>cloudformation:ListStackResources</code>
	<code>cloudformation:ListStackSetAutoDeploymentTargets</code>
	<code>cloudformation:ListStackSetOperationResults</code>
	<code>cloudformation:ListStackSetOperations</code>
	<code>cloudformation:ListStackSets</code>
	<code>cloudformation:ListTypeRegistrations</code>
	<code>cloudformation:ListTypes</code>
	<code>cloudformation:ListTypeVersions</code>
	<code>cloudformation:PublishType</code>
	<code>cloudformation:RecordHandlerProgress</code>
	<code>cloudformation:RegisterPublisher</code>

サービスプレフィックス	アクション
	cloudformation:RegisterType
	cloudformation:RollbackStack
	cloudformation:SetStackPolicy
	cloudformation:SetTypeConfiguration
	cloudformation:SetTypeDefaultVersion
	cloudformation:SignalResource
	cloudformation:StartResourceScan
	cloudformation:StopStackSetOperation
	cloudformation:TestType
	cloudformation:UpdateGeneratedTemplate
	cloudformation:UpdateStack
	cloudformation:UpdateStackInstances
	cloudformation:UpdateStackSet
	cloudformation:UpdateTerminationProtection
	cloudformation:ValidateTemplate

サービスプレフィックス	アクション
cloudfront	cloudfront:AssociateAlias
	cloudfront:CreateCachePolicy
	cloudfront:CreateCloudFrontOriginAccessIdentity
	cloudfront:CreateContinuousDeploymentPolicy
	cloudfront:CreateFieldLevelEncryptionConfig
	cloudfront:CreateFieldLevelEncryptionProfile
	cloudfront:CreateFunction
	cloudfront:CreateInvalidation
	cloudfront:CreateKeyGroup
	cloudfront:CreateKeyValueStore
	cloudfront:CreateMonitoringSubscription
	cloudfront:CreateOriginAccessControl
	cloudfront:CreateOriginRequestPolicy
	cloudfront:CreatePublicKey
	cloudfront:CreateRealtimeLogConfig
	cloudfront:CreateResponseHeadersPolicy
	cloudfront>DeleteCachePolicy
	cloudfront>DeleteCloudFrontOriginAccessIdentity
	cloudfront>DeleteContinuousDeploymentPolicy
	cloudfront>DeleteDistribution
	cloudfront>DeleteFieldLevelEncryptionConfig

サービスプレフィックス	アクション
	cloudfront:DeleteFieldLevelEncryptionProfile
	cloudfront:DeleteFunction
	cloudfront:DeleteKeyGroup
	cloudfront:DeleteKeyValueStore
	cloudfront:DeleteMonitoringSubscription
	cloudfront:DeleteOriginAccessControl
	cloudfront:DeleteOriginRequestPolicy
	cloudfront:DeletePublicKey
	cloudfront:DeleteRealtimeLogConfig
	cloudfront:DeleteResponseHeadersPolicy
	cloudfront:DeleteStreamingDistribution
	cloudfront:DescribeFunction
	cloudfront:DescribeKeyValueStore
	cloudfront:GetCachePolicy
	cloudfront:GetCachePolicyConfig
	cloudfront:GetCloudFrontOriginAccessIdentity
	cloudfront:GetCloudFrontOriginAccessIdentityConfig
	cloudfront:GetContinuousDeploymentPolicy
	cloudfront:GetContinuousDeploymentPolicyConfig
	cloudfront:GetDistributionConfig
	cloudfront:GetFieldLevelEncryption

サービスプレフィックス	アクション
	cloudfront:GetFieldLevelEncryptionConfig
	cloudfront:GetFieldLevelEncryptionProfile
	cloudfront:GetFieldLevelEncryptionProfileConfig
	cloudfront:GetFunction
	cloudfront:GetInvalidation
	cloudfront:GetKeyGroup
	cloudfront:GetKeyGroupConfig
	cloudfront:GetMonitoringSubscription
	cloudfront:GetOriginAccessControl
	cloudfront:GetOriginAccessControlConfig
	cloudfront:GetOriginRequestPolicy
	cloudfront:GetOriginRequestPolicyConfig
	cloudfront:GetPublicKey
	cloudfront:GetPublicKeyConfig
	cloudfront:GetRealtimeLogConfig
	cloudfront:GetResponseHeadersPolicy
	cloudfront:GetResponseHeadersPolicyConfig
	cloudfront:GetStreamingDistribution
	cloudfront:GetStreamingDistributionConfig
	cloudfront:ListCachePolicies
	cloudfront:ListCloudFrontOriginAccessIdentities

サービスプレフィックス	アクション
	cloudfront:ListConflictingAliases
	cloudfront:ListContinuousDeploymentPolicies
	cloudfront:ListDistributions
	cloudfront:ListDistributionsByCachePolicyId
	cloudfront:ListDistributionsByKeyGroup
	cloudfront:ListDistributionsByOriginRequestPolicyId
	cloudfront:ListDistributionsByRealtimeLogConfig
	cloudfront:ListDistributionsByResponseHeadersPolicyId
	cloudfront:ListDistributionsByWebACLId
	cloudfront:ListFieldLevelEncryptionConfigs
	cloudfront:ListFieldLevelEncryptionProfiles
	cloudfront:ListFunctions
	cloudfront:ListInvalidations
	cloudfront:ListKeyGroups
	cloudfront:ListKeyValueStores
	cloudfront:ListOriginAccessControls
	cloudfront:ListOriginRequestPolicies
	cloudfront:ListPublicKeys
	cloudfront:ListRealtimeLogConfigs
	cloudfront:ListResponseHeadersPolicies
	cloudfront:ListStreamingDistributions

サービスプレフィックス	アクション
	cloudfront:PublishFunction
	cloudfront:TestFunction
	cloudfront:UpdateCachePolicy
	cloudfront:UpdateCloudFrontOriginAccessIdentity
	cloudfront:UpdateContinuousDeploymentPolicy
	cloudfront:UpdateDistribution
	cloudfront:UpdateFieldLevelEncryptionConfig
	cloudfront:UpdateFieldLevelEncryptionProfile
	cloudfront:UpdateFunction
	cloudfront:UpdateKeyGroup
	cloudfront:UpdateKeyValueStore
	cloudfront:UpdateOriginAccessControl
	cloudfront:UpdateOriginRequestPolicy
	cloudfront:UpdatePublicKey
	cloudfront:UpdateRealtimeLogConfig
	cloudfront:UpdateResponseHeadersPolicy

サービスプレフィックス	アクション
cloudhsm	cloudhsm:CreateHapg
	cloudhsm:CreateHsm
	cloudhsm:CreateLunaClient
	cloudhsm>DeleteBackup
	cloudhsm>DeleteHapg
	cloudhsm>DeleteHsm
	cloudhsm>DeleteLunaClient
	cloudhsm:DescribeBackups
	cloudhsm:DescribeClusters
	cloudhsm:DescribeHapg
	cloudhsm:DescribeHsm
	cloudhsm:DescribeLunaClient
	cloudhsm:GetConfig
	cloudhsm:InitializeCluster
	cloudhsm>ListAvailableZones
	cloudhsm>ListHapgs
	cloudhsm>ListHsms
	cloudhsm>ListLunaClients
	cloudhsm:ModifyBackupAttributes
	cloudhsm:ModifyCluster
	cloudhsm:ModifyHapg



サービスプレフィックス	アクション
	cloudhsm:ModifyHsm
	cloudhsm:ModifyLunaClient
	cloudhsm:RestoreBackup

サービスプレフィックス	アクション
cloudsearch	cloudsearch:BuildSuggesters
	cloudsearch:CreateDomain
	cloudsearch:DefineAnalysisScheme
	cloudsearch:DefineExpression
	cloudsearch:DefineIndexField
	cloudsearch:DefineSuggester
	cloudsearch>DeleteAnalysisScheme
	cloudsearch>DeleteDomain
	cloudsearch>DeleteExpression
	cloudsearch>DeleteIndexField
	cloudsearch>DeleteSuggester
	cloudsearch:DescribeAnalysisSchemes
	cloudsearch:DescribeAvailabilityOptions
	cloudsearch:DescribeDomainEndpointOptions
	cloudsearch:DescribeDomains
	cloudsearch:DescribeExpressions
	cloudsearch:DescribeIndexFields
	cloudsearch:DescribeScalingParameters
	cloudsearch:DescribeServiceAccessPolicies
	cloudsearch:DescribeSuggesters
	cloudsearch:IndexDocuments

サービスプレフィックス	アクション
	cloudsearch:ListDomainNames cloudsearch:UpdateAvailabilityOptions cloudsearch:UpdateDomainEndpointOptions cloudsearch:UpdateScalingParameters cloudsearch:UpdateServiceAccessPolicies

サービスプレフィックス	アクション
CloudTrail	cloudtrail:CancelQuery
	cloudtrail:CreateChannel
	cloudtrail:CreateEventDataStore
	cloudtrail:CreateTrail
	cloudtrail>DeleteChannel
	cloudtrail>DeleteEventDataStore
	cloudtrail>DeleteResourcePolicy
	cloudtrail>DeleteTrail
	cloudtrail:DeregisterOrganizationDelegatedAdmin
	cloudtrail:DescribeQuery
	cloudtrail:DescribeTrails
	cloudtrail:DisableFederation
	cloudtrail:GetChannel
	cloudtrail:GetEventDataStore
	cloudtrail:GetEventDataStoreData
	cloudtrail:GetEventSelectors
	cloudtrail:GetImport
	cloudtrail:GetInsightSelectors
	cloudtrail:GetQueryResults
	cloudtrail:GetResourcePolicy
	cloudtrail:GetTrail

サービスプレフィックス	アクション
	cloudtrail:GetTrailStatus
	cloudtrail:ListChannels
	cloudtrail:ListEventDataStores
	cloudtrail:ListImportFailures
	cloudtrail:ListImports
	cloudtrail:ListPublicKeys
	cloudtrail:ListQueries
	cloudtrail:ListTrails
	cloudtrail:LookupEvents
	cloudtrail:PutEventSelectors
	cloudtrail:PutInsightSelectors
	cloudtrail:PutResourcePolicy
	cloudtrail:RegisterOrganizationDelegatedAdmin
	cloudtrail:RestoreEventDataStore
	cloudtrail:StartEventDataStoreIngestion
	cloudtrail:StartImport
	cloudtrail:StartLogging
	cloudtrail:StartQuery
	cloudtrail:StopEventDataStoreIngestion
	cloudtrail:StopImport
	cloudtrail:StopLogging

サービスプレフィックス	アクション
	cloudtrail:UpdateChannel
	cloudtrail:UpdateEventDataStore
	cloudtrail:UpdateTrail

サービスプレフィックス	アクション
cloudwatch	cloudwatch:DeleteAlarms
	cloudwatch:DeleteAnomalyDetector
	[cloudwatch:DeleteDashboards]
	cloudwatch:DeleteInsightRules
	cloudwatch:DeleteMetricStream
	cloudwatch:DescribeAlarmHistory
	cloudwatch:DescribeAlarms
	cloudwatch:DescribeAlarmsForMetric
	cloudwatch:DescribeAnomalyDetectors
	cloudwatch:DescribeInsightRules
	cloudwatch:DisableAlarmActions
	cloudwatch:DisableInsightRules
	cloudwatch:EnableAlarmActions
	cloudwatch:EnableInsightRules
	cloudwatch:GetDashboard
	cloudwatch:GetInsightRuleReport
	cloudwatch:GetMetricStream
	cloudwatch:ListDashboards
	cloudwatch:ListManagedInsightRules
	cloudwatch:ListMetricStreams
	cloudwatch:PutAnomalyDetector

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>cloudwatch:PutCompositeAlarm</li><li>cloudwatch:PutDashboard</li><li>cloudwatch:PutInsightRule</li><li>cloudwatch:PutManagedInsightRules</li><li>cloudwatch:PutMetricAlarm</li><li>cloudwatch:PutMetricStream</li><li>cloudwatch:SetAlarmState</li><li>cloudwatch:StartMetricStreams</li><li>cloudwatch:StopMetricStreams</li></ul>



サービスプレフィックス	アクション
codeartifact	codeartifact:AssociateExternalConnection
	codeartifact:CopyPackageVersions
	codeartifact:CreateDomain
	codeartifact:CreateRepository
	codeartifact>DeleteDomain
	codeartifact>DeleteDomainPermissionsPolicy
	codeartifact>DeletePackage
	codeartifact>DeletePackageVersions
	codeartifact>DeleteRepository
	codeartifact>DeleteRepositoryPermissionsPolicy
	codeartifact:DescribeDomain
	codeartifact:DescribePackage
	codeartifact:DescribePackageVersion
	codeartifact:DescribeRepository
	codeartifact:DisassociateExternalConnection
	codeartifact:DisposePackageVersions
	codeartifact:GetAssociatedPackageGroup
	codeartifact:GetAuthorizationToken
	codeartifact:GetDomainPermissionsPolicy
	codeartifact:GetPackageVersionAsset
	codeartifact:GetPackageVersionReadme

サービスプレフィックス	アクション
	<code>codeartifact:GetRepositoryEndpoint</code>
	<code>codeartifact:GetRepositoryPermissionsPolicy</code>
	<code>codeartifact:ListDomains</code>
	<code>codeartifact:ListPackageGroups</code>
	<code>codeartifact:ListPackages</code>
	<code>codeartifact:ListPackageVersionAssets</code>
	<code>codeartifact:ListPackageVersionDependencies</code>
	<code>codeartifact:ListPackageVersions</code>
	<code>codeartifact:ListRepositories</code>
	<code>codeartifact:ListRepositoriesInDomain</code>
	<code>codeartifact:PublishPackageVersion</code>
	<code>codeartifact:PutDomainPermissionsPolicy</code>
	<code>codeartifact:PutPackageMetadata</code>
	<code>codeartifact:PutPackageOriginConfiguration</code>
	<code>codeartifact:PutRepositoryPermissionsPolicy</code>
	<code>codeartifact:ReadFromRepository</code>
	<code>codeartifact:UpdatePackageVersionsStatus</code>
	<code>codeartifact:UpdateRepository</code>

サービスプレフィックス	アクション
codedeploy	codedeploy:BatchGetApplicationRevisions
	codedeploy:BatchGetApplications
	codedeploy:BatchGetDeploymentGroups
	codedeploy:BatchGetDeploymentInstances
	codedeploy:BatchGetDeployments
	codedeploy:BatchGetDeploymentTargets
	codedeploy:BatchGetOnPremisesInstances
	codedeploy:ContinueDeployment
	codedeploy:CreateApplication
	codedeploy:CreateDeployment
	codedeploy:CreateDeploymentConfig
	codedeploy:CreateDeploymentGroup
	codedeploy>DeleteApplication
	codedeploy>DeleteDeploymentConfig
	codedeploy>DeleteDeploymentGroup
	codedeploy>DeleteGitHubAccountToken
	codedeploy>DeleteResourcesByExternalId
	codedeploy:DeregisterOnPremisesInstance
	codedeploy:GetApplication
	codedeploy:GetApplicationRevision
	codedeploy:GetDeployment

サービスプレフィックス	アクション
	<p>codedeploy:GetDeploymentConfig</p> <p>codedeploy:GetDeploymentGroup</p> <p>codedeploy:GetDeploymentInstance</p> <p>codedeploy:GetDeploymentTarget</p> <p>codedeploy:GetOnPremisesInstance</p> <p>codedeploy:ListApplicationRevisions</p> <p>codedeploy:ListApplications</p> <p>codedeploy:ListDeploymentConfigs</p> <p>codedeploy:ListDeploymentGroups</p> <p>codedeploy:ListDeploymentInstances</p> <p>codedeploy:ListDeployments</p> <p>codedeploy:ListDeploymentTargets</p> <p>codedeploy:ListGitHubAccountTokenNames</p> <p>codedeploy:ListOnPremisesInstances</p> <p>codedeploy:PutLifecycleEventHookExecutionStatus</p> <p>codedeploy:RegisterApplicationRevision</p> <p>codedeploy:RegisterOnPremisesInstance</p> <p>codedeploy:SkipWaitTimeForInstanceTermination</p> <p>codedeploy:StopDeployment</p> <p>codedeploy:UpdateApplication</p> <p>codedeploy:UpdateDeploymentGroup</p>

サービスプレフィックス	アクション
codeguru-profiler	codeguru-profiler:AddNotificationChannels
	codeguru-profiler:BatchGetFrameMetricData
	codeguru-profiler:ConfigureAgent
	codeguru-profiler:CreateProfilingGroup
	codeguru-profiler>DeleteProfilingGroup
	codeguru-profiler:DescribeProfilingGroup
	codeguru-profiler:GetFindingsReportAccountSummary
	codeguru-profiler:GetNotificationConfiguration
	codeguru-profiler:GetPolicy
	codeguru-profiler:GetProfile
	codeguru-profiler:GetRecommendations
	codeguru-profiler:ListFindingsReports
	codeguru-profiler:ListProfileTimes
	codeguru-profiler:ListProfilingGroups
	codeguru-profiler:PutPermission
	codeguru-profiler:RemoveNotificationChannel
	codeguru-profiler:RemovePermission
	codeguru-profiler:SubmitFeedback
	codeguru-profiler:UpdateProfilingGroup

サービスプレフィックス	アクション
codeguru-reviewer	codeguru-reviewer:AssociateRepository codeguru-reviewer:CreateCodeReview codeguru-reviewer:DescribeCodeReview codeguru-reviewer:DescribeRecommendationFeedback codeguru-reviewer:DescribeRepositoryAssociation codeguru-reviewer:DisassociateRepository codeguru-reviewer:ListCodeReviews codeguru-reviewer:ListRecommendationFeedback codeguru-reviewer:ListRecommendations codeguru-reviewer:ListRepositoryAssociations codeguru-reviewer:PutRecommendationFeedback

サービスプレフィックス	アクション
codepipeline	codepipeline:AcknowledgeJob
	codepipeline:AcknowledgeThirdPartyJob
	codepipeline:CreateCustomActionType
	codepipeline:CreatePipeline
	codepipeline>DeleteCustomActionType
	codepipeline>DeletePipeline
	codepipeline>DeleteWebhook
	codepipeline:DeregisterWebhookWithThirdParty
	codepipeline:GetActionType
	codepipeline:GetJobDetails
	codepipeline:GetPipeline
	codepipeline:GetPipelineExecution
	codepipeline:GetPipelineState
	codepipeline:GetThirdPartyJobDetails
	codepipeline:ListActionExecutions
	codepipeline:ListActionTypes
	codepipeline:ListPipelineExecutions
	codepipeline:ListPipelines
	codepipeline:ListWebhooks
	codepipeline:PollForJobs
	codepipeline:PollForThirdPartyJobs

サービスプレフィックス	アクション
	<code>codepipeline:PutActionRevision</code>
	<code>codepipeline:PutApprovalResult</code>
	<code>codepipeline:PutJobFailureResult</code>
	<code>codepipeline:PutJobSuccessResult</code>
	<code>codepipeline:PutThirdPartyJobFailureResult</code>
	<code>codepipeline:PutThirdPartyJobSuccessResult</code>
	<code>codepipeline:PutWebhook</code>
	<code>codepipeline:RegisterWebhookWithThirdParty</code>
	<code>codepipeline:RollbackStage</code>
	<code>codepipeline:StartPipelineExecution</code>
	<code>codepipeline:StopPipelineExecution</code>
	<code>codepipeline:UpdateActionType</code>
	<code>codepipeline:UpdatePipeline</code>



サービスプレフィックス	アクション
codestar	codestar:AssociateTeamMember
	codestar:CreateProject
	codestar:CreateUserProfile
	codestar>DeleteProject
	codestar>DeleteUserProfile
	codestar:DescribeProject
	codestar:DescribeUserProfile
	codestar:DisassociateTeamMember
	codestar:ListProjects
	codestar:ListResources
	codestar:ListTeamMembers
	codestar:ListUserProfiles
	codestar:UpdateProject
	codestar:UpdateTeamMember
	codestar:UpdateUserProfile

サービスプレフィックス	アクション
codestar-notifications	codestar-notifications:CreateNotificationRule codestar-notifications>DeleteNotificationRule codestar-notifications>DeleteTarget codestar-notifications:DescribeNotificationRule codestar-notifications:ListEventTypes codestar-notifications:ListNotificationRules codestar-notifications:ListTargets codestar-notifications:Subscribe codestar-notifications:Unsubscribe codestar-notifications:UpdateNotificationRule

サービスプレフィックス	アクション
cognito-identity	cognito-identity:CreateIdentityPool
	cognito-identity:DeleteIdentities
	cognito-identity:DeleteIdentityPool
	cognito-identity:DescribeIdentity
	cognito-identity:DescribeIdentityPool
	cognito-identity:GetIdentityPoolRoles
	cognito-identity:ListIdentities
	cognito-identity:ListIdentityPools
	cognito-identity:LookupDeveloperIdentity
	cognito-identity:MergeDeveloperIdentities
	cognito-identity:SetIdentityPoolRoles
	cognito-identity:UnlinkDeveloperIdentity
	cognito-identity:UpdateIdentityPool

サービスプレフィックス	アクション
cognito-idp	cognito-idp:AddCustomAttributes
	cognito-idp:AdminAddUserToGroup
	cognito-idp:AdminConfirmSignUp
	cognito-idp:AdminCreateUser
	cognito-idp:AdminDeleteUser
	cognito-idp:AdminDeleteUserAttributes
	cognito-idp:AdminDisableProviderForUser
	cognito-idp:AdminDisableUser
	cognito-idp:AdminEnableUser
	cognito-idp:AdminForgetDevice
	cognito-idp:AdminGetDevice
	cognito-idp:AdminGetUser
	cognito-idp:AdminInitiateAuth
	cognito-idp:AdminLinkProviderForUser
	cognito-idp:AdminListDevices
	cognito-idp:AdminListGroupsWithUser
	cognito-idp:AdminListUserAuthEvents
	cognito-idp:AdminRemoveUserFromGroup
	cognito-idp:AdminResetUserPassword
	cognito-idp:AdminRespondToAuthChallenge
	cognito-idp:AdminSetUserMFAPreference

サービスプレフィックス	アクション
	cognito-idp:AdminSetUserPassword
	cognito-idp:AdminSetUserSettings
	cognito-idp:AdminUpdateAuthEventFeedback
	cognito-idp:AdminUpdateDeviceStatus
	cognito-idp:AdminUpdateUserAttributes
	cognito-idp:AdminUserGlobalSignOut
	cognito-idp:AssociateSoftwareToken
	cognito-idp:ChangePassword
	cognito-idp:ConfirmDevice
	cognito-idp:ConfirmForgotPassword
	cognito-idp:ConfirmSignUp
	cognito-idp>CreateGroup
	cognito-idp:CreateIdentityProvider
	cognito-idp>CreateResourceServer
	cognito-idp>CreateUserImportJob
	cognito-idp>CreateUserPool
	cognito-idp>CreateUserPoolClient
	cognito-idp>CreateUserPoolDomain
	cognito-idp>DeleteGroup
	cognito-idp>DeleteIdentityProvider
	cognito-idp>DeleteResourceServer

サービスプレフィックス	アクション
	cognito-idp:DeleteUser
	cognito-idp:DeleteUserAttributes
	cognito-idp:DeleteUserPool
	cognito-idp:DeleteUserPoolClient
	cognito-idp:DeleteUserPoolDomain
	cognito-idp:DescribeIdentityProvider
	cognito-idp:DescribeResourceServer
	cognito-idp:DescribeRiskConfiguration
	cognito-idp:DescribeUserImportJob
	cognito-idp:DescribeUserPool
	cognito-idp:DescribeUserPoolClient
	cognito-idp:DescribeUserPoolDomain
	cognito-idp:ForgetDevice
	cognito-idp:ForgotPassword
	cognito-idp:GetCSVHeader
	cognito-idp:GetDevice
	cognito-idp:GetGroup
	cognito-idp:GetIdentityProviderByIdentifier
	cognito-idp:GetLogDeliveryConfiguration
	cognito-idp:GetSigningCertificate
	cognito-idp:GetUICustomization

サービスプレフィックス	アクション
	<code>cognito-idp:GetUser</code>
	<code>cognito-idp:GetUserAttributeVerificationCode</code>
	<code>cognito-idp:GetUserPoolMfaConfig</code>
	<code>cognito-idp:GlobalSignOut</code>
	<code>cognito-idp:InitiateAuth</code>
	<code>cognito-idp:ListDevices</code>
	<code>cognito-idp:ListGroups</code>
	<code>cognito-idp:ListIdentityProviders</code>
	<code>cognito-idp:ListResourceServers</code>
	<code>cognito-idp:ListUserImportJobs</code>
	<code>cognito-idp:ListUserPoolClients</code>
	<code>cognito-idp:ListUserPools</code>
	<code>cognito-idp:ListUsers</code>
	<code>cognito-idp:ListUsersInGroup</code>
	<code>cognito-idp:ResendConfirmationCode</code>
	<code>cognito-idp:RespondToAuthChallenge</code>
	<code>cognito-idp:RevokeToken</code>
	<code>cognito-idp:SetLogDeliveryConfiguration</code>
	<code>cognito-idp:SetRiskConfiguration</code>
	<code>cognito-idp:SetUICustomization</code>
	<code>cognito-idp:SetUserMFAPreference</code>

サービスプレフィックス	アクション
	<code>cognito-idp:SetUserPoolMfaConfig</code>
	<code>cognito-idp:SetUserSettings</code>
	<code>cognito-idp:SignUp</code>
	<code>cognito-idp:StartUserImportJob</code>
	<code>cognito-idp:StopUserImportJob</code>
	<code>cognito-idp:UpdateAuthEventFeedback</code>
	<code>cognito-idp:UpdateDeviceStatus</code>
	<code>cognito-idp:UpdateGroup</code>
	<code>cognito-idp:UpdateIdentityProvider</code>
	<code>cognito-idp:UpdateResourceServer</code>
	<code>cognito-idp:UpdateUserAttributes</code>
	<code>cognito-idp:UpdateUserPool</code>
	<code>cognito-idp:UpdateUserPoolClient</code>
	<code>cognito-idp:UpdateUserPoolDomain</code>
	<code>cognito-idp:VerifySoftwareToken</code>
	<code>cognito-idp:VerifyUserAttribute</code>



サービスプレフィックス	アクション
cognito-sync	cognito-sync:BulkPublish
	cognito-sync>DeleteDataset
	cognito-sync:DescribeDataset
	cognito-sync:DescribeIdentityPoolUsage
	cognito-sync:DescribeIdentityUsage
	cognito-sync:GetBulkPublishDetails
	cognito-sync:GetCognitoEvents
	cognito-sync:GetIdentityPoolConfiguration
	cognito-sync:ListDatasets
	cognito-sync:ListIdentityPoolUsage
	cognito-sync:ListRecords
	cognito-sync:RegisterDevice
	cognito-sync:SetCognitoEvents
	cognito-sync:SetIdentityPoolConfiguration
	cognito-sync:SubscribeToDataset
	cognito-sync:UnsubscribeFromDataset
	cognito-sync:UpdateRecords

サービスプレフィックス	アクション
comprehendmedical	comprehendmedical:DescribeEntitiesDetectionV2Job comprehendmedical:DescribeICD10CMIInferenceJob comprehendmedical:DescribePHIDetectionJob comprehendmedical:DescribeRxNormInferenceJob comprehendmedical:DescribeSNOMEDCTInferenceJob comprehendmedical:DetectEntitiesV2 comprehendmedical:DetectPHI comprehendmedical:InferICD10CM comprehendmedical:InferRxNorm comprehendmedical:InferSNOMEDCT comprehendmedical:ListEntitiesDetectionV2Jobs comprehendmedical:ListICD10CMIInferenceJobs comprehendmedical:ListPHIDetectionJobs comprehendmedical:ListRxNormInferenceJobs comprehendmedical:ListSNOMEDCTInferenceJobs comprehendmedical:StartEntitiesDetectionV2Job comprehendmedical:StartICD10CMIInferenceJob comprehendmedical:StartPHIDetectionJob comprehendmedical:StartRxNormInferenceJob comprehendmedical:StartSNOMEDCTInferenceJob comprehendmedical:StopEntitiesDetectionV2Job

サービスプレフィックス	アクション
	comprehendmedical:StopICD10CMIInferenceJob
	comprehendmedical:StopPHIDetectionJob
	comprehendmedical:StopRxNormInferenceJob
	comprehendmedical:StopSNOMEDCTInferenceJob

サービスプレフィックス	アクション
compute-optimizer	compute-optimizer:DeleteRecommendationPreferences
	compute-optimizer:DescribeRecommendationExportJobs
	compute-optimizer:ExportAutoScalingGroupRecommendations
	compute-optimizer:ExportEBSVolumeRecommendations
	compute-optimizer:ExportEC2InstanceRecommendations
	compute-optimizer:ExportECSServiceRecommendations
	compute-optimizer:ExportLambdaFunctionRecommendations
	compute-optimizer:ExportLicenseRecommendations
	compute-optimizer:GetEC2RecommendationProjectedMetrics
	compute-optimizer:GetECSServiceRecommendationProjectedMetrics
	compute-optimizer:GetEffectiveRecommendationPreferences
	compute-optimizer:GetEnrollmentStatus
	compute-optimizer:GetEnrollmentStatusesForOrganization
	compute-optimizer:GetRecommendationPreferences
	compute-optimizer:GetRecommendationSummaries
	compute-optimizer:PutRecommendationPreferences
	compute-optimizer:UpdateEnrollmentStatus

サービスプレフィックス	アクション
config	config:BatchGetResourceConfig
	config>DeleteAggregationAuthorization
	config>DeleteConfigRule
	config>DeleteConfigurationAggregator
	config>DeleteConfigurationRecorder
	config>DeleteConformancePack
	config>DeleteDeliveryChannel
	config>DeleteEvaluationResults
	config>DeleteOrganizationConfigRule
	config>DeleteOrganizationConformancePack
	config>DeletePendingAggregationRequest
	config>DeleteRemediationConfiguration
	config>DeleteRemediationExceptions
	config>DeleteResourceConfig
	config>DeleteRetentionConfiguration
	config>DeleteStoredQuery
	config:DeliverConfigSnapshot
	config:DescribeAggregateComplianceByConfigRules
	config:DescribeAggregateComplianceByConformancePacks
	config:DescribeAggregationAuthorizations
	config:DescribeComplianceByConfigRule

サービスプレフィックス	アクション
	<code>config:DescribeComplianceByResource</code>
	<code>config:DescribeConfigRuleEvaluationStatus</code>
	<code>config:DescribeConfigRules</code>
	<code>config:DescribeConfigurationAggregators</code>
	<code>config:DescribeConfigurationAggregatorSourcesStatus</code>
	<code>config:DescribeConfigurationRecorders</code>
	<code>config:DescribeConfigurationRecorderStatus</code>
	<code>config:DescribeConformancePackCompliance</code>
	<code>config:DescribeConformancePacks</code>
	<code>config:DescribeConformancePackStatus</code>
	<code>config:DescribeDeliveryChannels</code>
	<code>config:DescribeDeliveryChannelStatus</code>
	<code>config:DescribeOrganizationConfigRules</code>
	<code>config:DescribeOrganizationConfigRuleStatuses</code>
	<code>config:DescribeOrganizationConformancePacks</code>
	<code>config:DescribeOrganizationConformancePackStatuses</code>
	<code>config:DescribePendingAggregationRequests</code>
	<code>config:DescribeRemediationConfigurations</code>
	<code>config:DescribeRemediationExceptions</code>
	<code>config:DescribeRemediationExecutionStatus</code>
	<code>config:DescribeRetentionConfigurations</code>

サービスプレフィックス	アクション
	<code>config:GetComplianceDetailsByConfigRule</code>
	<code>config:GetComplianceDetailsByResource</code>
	<code>config:GetComplianceSummaryByConfigRule</code>
	<code>config:GetComplianceSummaryByResourceType</code>
	<code>config:GetConformancePackComplianceDetails</code>
	<code>config:GetConformancePackComplianceSummary</code>
	<code>config:GetCustomRulePolicy</code>
	<code>config:GetDiscoveredResourceCounts</code>
	<code>config:GetOrganizationConfigRuleDetailedStatus</code>
	<code>config:GetOrganizationConformancePackDetailedStatus</code>
	<code>config:GetOrganizationCustomRulePolicy</code>
	<code>config:GetResourceConfigHistory</code>
	<code>config:GetResourceEvaluationSummary</code>
	<code>config:GetStoredQuery</code>
	<code>config:ListConformancePackComplianceScores</code>
	<code>config:ListDiscoveredResources</code>
	<code>config:ListResourceEvaluations</code>
	<code>config:ListStoredQueries</code>
	<code>config:PutConfigRule</code>
	<code>config:PutConfigurationAggregator</code>
	<code>config:PutConfigurationRecorder</code>

サービスプレフィックス	アクション
	<code>config:PutConformancePack</code>
	<code>config:PutDeliveryChannel</code>
	<code>config:PutEvaluations</code>
	<code>config:PutExternalEvaluation</code>
	<code>config:PutOrganizationConfigRule</code>
	<code>config:PutOrganizationConformancePack</code>
	<code>config:PutRemediationConfigurations</code>
	<code>config:PutRemediationExceptions</code>
	<code>config:PutResourceConfig</code>
	<code>config:PutRetentionConfiguration</code>
	<code>config:PutStoredQuery</code>
	<code>config:SelectResourceConfig</code>
	<code>config:StartConfigRulesEvaluation</code>
	<code>config:StartConfigurationRecorder</code>
	<code>config:StartRemediationExecution</code>
	<code>config:StartResourceEvaluation</code>
	<code>config:StopConfigurationRecorder</code>



サービスプレフィックス	アクション
接続	connect:ActivateEvaluationForm
	connect:AssociateApprovedOrigin
	connect:AssociateBot
	connect:AssociateDefaultVocabulary
	connect:AssociateFlow
	connect:AssociateInstanceStorageConfig
	connect:AssociateLambdaFunction
	connect:AssociateLexBot
	connect:AssociatePhoneNumberContactFlow
	connect:AssociateQueueQuickConnects
	connect:AssociateRoutingProfileQueues
	connect:AssociateSecurityKey
	connect:AssociateUserProficiencies
	connect:BatchGetFlowAssociation
	connect:BatchPutContact
	connect:ClaimPhoneNumber
	connect:CreateAgentStatus
	connect:CreateContactFlow
	connect:CreateContactFlowModule
	connect:CreateEvaluationForm
	connect:CreateHoursOfOperation

サービスプレフィックス	アクション
	connect:CreateInstance
	connect:CreateIntegrationAssociation
	connect:CreateParticipant
	connect:CreatePersistentContactAssociation
	connect:CreatePredefinedAttribute
	connect:CreatePrompt
	connect:CreateQueue
	connect:CreateQuickConnect
	connect:CreateRoutingProfile
	connect:CreateRule
	connect:CreateSecurityProfile
	connect:CreateTaskTemplate
	connect:CreateTrafficDistributionGroup
	connect:CreateUseCase
	connect:CreateUser
	connect:CreateUserHierarchyGroup
	connect:CreateView
	connect:CreateViewVersion
	connect:CreateVocabulary
	connect:DeactivateEvaluationForm
	connect>DeleteContactEvaluation

サービスプレフィックス	アクション
	connect:DeleteContactFlow
	connect:DeleteContactFlowModule
	connect:DeleteEvaluationForm
	connect:DeleteHoursOfOperation
	connect:DeleteInstance
	connect:DeleteIntegrationAssociation
	connect:DeletePredefinedAttribute
	connect:DeletePrompt
	connect:DeleteQueue
	connect:DeleteQuickConnect
	connect:DeleteRoutingProfile
	connect:DeleteRule
	connect:DeleteSecurityProfile
	connect:DeleteTaskTemplate
	connect:DeleteTrafficDistributionGroup
	connect:DeleteUseCase
	connect:DeleteUser
	connect:DeleteUserHierarchyGroup
	connect:DeleteView
	connect:DeleteVocabulary
	connect:DescribeAgentStatus

サービスプレフィックス	アクション
	<p>connect:DescribeContactEvaluation</p> <p>connect:DescribeContactFlow</p> <p>connect:DescribeContactFlowModule</p> <p>connect:DescribeEvaluationForm</p> <p>connect:DescribeInstanceAttribute</p> <p>connect:DescribeInstanceStorageConfig</p> <p>connect:DescribePhoneNumber</p> <p>connect:DescribeRule</p> <p>connect:DescribeTrafficDistributionGroup</p> <p>connect:DescribeUserHierarchyGroup</p> <p>connect:DescribeUserHierarchyStructure</p> <p>connect:DescribeView</p> <p>connect:DescribeVocabulary</p> <p>connect:DisassociateApprovedOrigin</p> <p>connect:DisassociateBot</p> <p>connect:DisassociateFlow</p> <p>connect:DisassociateInstanceStorageConfig</p> <p>connect:DisassociateLambdaFunction</p> <p>connect:DisassociateLexBot</p> <p>connect:DisassociatePhoneNumberContactFlow</p> <p>connect:DisassociateQueueQuickConnects</p>

サービスプレフィックス	アクション
	<p>connect:DisassociateRoutingProfileQueues</p> <p>connect:DisassociateSecurityKey</p> <p>connect:DisassociateUserProficiencies</p> <p>connect:DismissUserContact</p> <p>connect:GetContactAttributes</p> <p>connect:GetCurrentMetricData</p> <p>connect:GetCurrentUserData</p> <p>connect:GetFederationToken</p> <p>connect:GetFlowAssociation</p> <p>connect:GetMetricData</p> <p>connect:GetMetricDataV2</p> <p>connect:GetPromptFile</p> <p>connect:GetTaskTemplate</p> <p>connect:GetTrafficDistribution</p> <p>connect:ImportPhoneNumber</p> <p>connect:ListApprovedOrigins</p> <p>connect:ListBots</p> <p>connect:ListContactEvaluations</p> <p>connect:ListContactFlowModules</p> <p>connect:ListContactFlows</p> <p>connect:ListContactReferences</p>

サービスプレフィックス	アクション
	<p>connect:ListDefaultVocabularies</p> <p>connect:ListEvaluationForms</p> <p>connect:ListEvaluationFormVersions</p> <p>connect:ListFlowAssociations</p> <p>connect:ListHoursOfOperations</p> <p>connect:ListInstanceAttributes</p> <p>connect:ListInstanceStorageConfigs</p> <p>connect:ListIntegrationAssociations</p> <p>connect:ListLambdaFunctions</p> <p>connect:ListLexBots</p> <p>connect:ListPhoneNumbers</p> <p>connect:ListPhoneNumbersV2</p> <p>connect:ListPredefinedAttributes</p> <p>connect:ListPrompts</p> <p>connect:ListQueueQuickConnects</p> <p>connect:ListQueues</p> <p>connect:ListQuickConnects</p> <p>connect:ListRealtimeContactAnalysisSegmentsV2</p> <p>connect:ListRoutingProfileQueues</p> <p>connect:ListRoutingProfiles</p> <p>connect:ListRules</p>

サービスプレフィックス	アクション
	<p>connect:ListSecurityKeys</p> <p>connect:ListSecurityProfileApplications</p> <p>connect:ListSecurityProfilePermissions</p> <p>connect:ListSecurityProfiles</p> <p>connect:ListTaskTemplates</p> <p>connect:ListTrafficDistributionGroups</p> <p>connect:ListUseCases</p> <p>connect:ListUserHierarchyGroups</p> <p>connect:ListUserProficiencies</p> <p>connect:ListUsers</p> <p>connect:ListViews</p> <p>connect:ListViewVersions</p> <p>connect:MonitorContact</p> <p>connect:PauseContact</p> <p>connect:PutUserStatus</p> <p>connect:ReleasePhoneNumber</p> <p>connect:ReplicateInstance</p> <p>connect:ResumeContact</p> <p>connect:ResumeContactRecording</p> <p>connect:SearchAvailablePhoneNumbers</p> <p>connect:SearchContacts</p>

サービスプレフィックス	アクション
	<p>connect:SearchHoursOfOperations</p> <p>connect:SearchPredefinedAttributes</p> <p>connect:SearchPrompts</p> <p>connect:SearchQueues</p> <p>connect:SearchQuickConnects</p> <p>connect:SearchRoutingProfiles</p> <p>connect:SearchSecurityProfiles</p> <p>connect:SearchVocabularies</p> <p>connect:SendChatIntegrationEvent</p> <p>connect:StartChatContact</p> <p>connect:StartContactEvaluation</p> <p>connect:StartContactRecording</p> <p>connect:StartContactStreaming</p> <p>connect:StartOutboundVoiceContact</p> <p>connect:StartTaskContact</p> <p>connect:StartWebRTCContact</p> <p>connect:StopContact</p> <p>connect:StopContactRecording</p> <p>connect:StopContactStreaming</p> <p>connect:SubmitContactEvaluation</p> <p>connect:SuspendContactRecording</p>



サービスプレフィックス	アクション
	<p>connect:TransferContact</p> <p>connect:UpdateAgentStatus</p> <p>connect:UpdateContact</p> <p>connect:UpdateContactAttributes</p> <p>connect:UpdateContactEvaluation</p> <p>connect:UpdateContactFlowContent</p> <p>connect:UpdateContactFlowMetadata</p> <p>connect:UpdateContactFlowModuleContent</p> <p>connect:UpdateContactFlowModuleMetadata</p> <p>connect:UpdateContactFlowName</p> <p>connect:UpdateContactRoutingData</p> <p>connect:UpdateContactSchedule</p> <p>connect:UpdateEvaluationForm</p> <p>connect:UpdateHoursOfOperation</p> <p>connect:UpdateInstanceAttribute</p> <p>connect:UpdateInstanceStorageConfig</p> <p>connect:UpdateParticipantRoleConfig</p> <p>connect:UpdatePhoneNumber</p> <p>connect:UpdatePhoneNumberMetadata</p> <p>connect:UpdatePredefinedAttribute</p> <p>connect:UpdatePrompt</p>

サービスプレフィックス	アクション
	connect:UpdateQueueHoursOfOperation
	connect:UpdateQueueMaxContacts
	connect:UpdateQueueName
	connect:UpdateQueueOutboundCallerConfig
	connect:UpdateQueueStatus
	connect:UpdateQuickConnectConfig
	connect:UpdateQuickConnectName
	connect:UpdateRoutingProfileAgentAvailabilityTimer
	connect:UpdateRoutingProfileConcurrency
	connect:UpdateRoutingProfileDefaultOutboundQueue
	connect:UpdateRoutingProfileName
	connect:UpdateRoutingProfileQueues
	connect:UpdateRule
	connect:UpdateSecurityProfile
	connect:UpdateTaskTemplate
	connect:UpdateTrafficDistribution
	connect:UpdateUserHierarchy
	connect:UpdateUserHierarchyGroupName
	connect:UpdateUserHierarchyStructure
	connect:UpdateUserIdentityInfo
	connect:UpdateUserPhoneConfig

サービスプレフィックス	アクション
	connect:UpdateUserProficiencies connect:UpdateUserRoutingProfile connect:UpdateUserSecurityProfiles connect:UpdateViewContent connect:UpdateViewMetadata
cur	cur:DeleteReportDefinition cur:DescribeReportDefinitions cur:ModifyReportDefinition cur:PutReportDefinition

サービスプレフィックス	アクション
databrew	databrew:BatchDeleteRecipeVersion
	databrew:CreateDataset
	databrew:CreateProfileJob
	databrew:CreateProject
	databrew:CreateRecipe
	databrew:CreateRecipeJob
	databrew:CreateRuleset
	databrew:CreateSchedule
	databrew>DeleteDataset
	databrew>DeleteJob
	databrew>DeleteProject
	databrew>DeleteRecipeVersion
	databrew>DeleteRuleset
	databrew>DeleteSchedule
	databrew:DescribeDataset
	databrew:DescribeJob
	databrew:DescribeJobRun
	databrew:DescribeProject
	databrew:DescribeRecipe
	databrew:DescribeRuleset
	databrew:DescribeSchedule

サービスプレフィックス	アクション
	<code>databrew:ListDatasets</code>
	<code>databrew:ListJobRuns</code>
	<code>databrew:ListJobs</code>
	<code>databrew:ListProjects</code>
	<code>databrew:ListRecipes</code>
	<code>databrew:ListRecipeVersions</code>
	<code>databrew:ListRulesets</code>
	<code>databrew:ListSchedules</code>
	<code>databrew:PublishRecipe</code>
	<code>databrew:SendProjectSessionAction</code>
	<code>databrew:StartJobRun</code>
	<code>databrew:StartProjectSession</code>
	<code>databrew:StopJobRun</code>
	<code>databrew:UpdateDataset</code>
	<code>databrew:UpdateProfileJob</code>
	<code>databrew:UpdateProject</code>
	<code>databrew:UpdateRecipe</code>
	<code>databrew:UpdateRecipeJob</code>
	<code>databrew:UpdateRuleset</code>
	<code>databrew:UpdateSchedule</code>

サービスプレフィックス	アクション
dataexchange	dataexchange:CancelJob
	dataexchange:CreateDataSet
	dataexchange:CreateEventAction
	dataexchange:CreateJob
	dataexchange:CreateRevision
	dataexchange>DeleteAsset
	dataexchange>DeleteEventAction
	dataexchange>DeleteRevision
	dataexchange:GetEventAction
	dataexchange:GetJob
	dataexchange:ListDataSetRevisions
	dataexchange:ListDataSets
	dataexchange:ListEventActions
	dataexchange:ListJobs
	dataexchange:ListRevisionAssets
	dataexchange:RevokeRevision
	dataexchange:SendDataSetNotification
	dataexchange:StartJob
	dataexchange:UpdateAsset
	dataexchange:UpdateDataSet
	dataexchange:UpdateEventAction

サービスプレフィックス	アクション
	dataexchange:UpdateRevision
datapipeline	datapipeline:ActivatePipeline
	datapipeline:CreatePipeline
	datapipeline:DeactivatePipeline
	datapipeline>DeletePipeline
	datapipeline:DescribeObjects
	datapipeline:DescribePipelines
	datapipeline:EvaluateExpression
	datapipeline:GetPipelineDefinition
	datapipeline:ListPipelines
	datapipeline:PollForTask
	datapipeline:PutPipelineDefinition
	datapipeline:QueryObjects
	datapipeline:ReportTaskProgress
	datapipeline:ReportTaskRunnerHeartbeat
	datapipeline:SetStatus
	datapipeline:SetTaskStatus
	datapipeline:ValidatePipelineDefinition

サービスプレフィックス	アクション
dax	dax:CreateCluster
	dax:DecreaseReplicationFactor
	dax>DeleteCluster
	dax>DeleteParameterGroup
	dax>DeleteSubnetGroup
	dax:DescribeClusters
	dax:DescribeDefaultParameters
	dax:DescribeEvents
	dax:DescribeParameterGroups
	dax:DescribeParameters
	dax:DescribeSubnetGroups
	dax:IncreaseReplicationFactor
	dax:RebootNode
	dax:UpdateCluster
	dax:UpdateParameterGroup
	dax:UpdateSubnetGroup



サービスプレフィックス	アクション
devicefarm	devicefarm:CreateDevicePool
	devicefarm:CreateInstanceProfile
	devicefarm:CreateNetworkProfile
	devicefarm:CreateProject
	devicefarm:CreateRemoteAccessSession
	devicefarm:CreateTestGridProject
	devicefarm:CreateTestGridUrl
	devicefarm:CreateUpload
	devicefarm:CreateVPCEConfiguration
	devicefarm>DeleteDevicePool
	devicefarm>DeleteInstanceProfile
	devicefarm>DeleteNetworkProfile
	devicefarm>DeleteProject
	devicefarm>DeleteRemoteAccessSession
	devicefarm>DeleteRun
	devicefarm>DeleteTestGridProject
	devicefarm>DeleteUpload
	devicefarm>DeleteVPCEConfiguration
	devicefarm:GetAccountSettings
	devicefarm:GetDevice
devicefarm:GetDeviceInstance	

サービスプレフィックス	アクション
	devicefarm:GetDevicePool
	devicefarm:GetDevicePoolCompatibility
	devicefarm:GetInstanceProfile
	devicefarm:GetJob
	devicefarm:GetNetworkProfile
	devicefarm:GetOfferingStatus
	devicefarm:GetProject
	devicefarm:GetRemoteAccessSession
	devicefarm:GetRun
	devicefarm:GetSuite
	devicefarm:GetTest
	devicefarm:GetTestGridProject
	devicefarm:GetTestGridSession
	devicefarm:GetUpload
	devicefarm:GetVPCEConfiguration
	devicefarm:ListArtifacts
	devicefarm:ListDeviceInstances
	devicefarm:ListDevicePools
	devicefarm:ListDevices
	devicefarm:ListInstanceProfiles
	devicefarm:ListJobs

サービスプレフィックス	アクション
	devicefarm:ListNetworkProfiles
	devicefarm:ListOfferingPromotions
	devicefarm:ListOfferings
	devicefarm:ListOfferingTransactions
	devicefarm:ListProjects
	devicefarm:ListRemoteAccessSessions
	devicefarm:ListRuns
	devicefarm:ListSamples
	devicefarm:ListSuites
	devicefarm:ListTestGridProjects
	devicefarm:ListTestGridSessionActions
	devicefarm:ListTestGridSessionArtifacts
	devicefarm:ListTestGridSessions
	devicefarm:ListTests
	devicefarm:ListUniqueProblems
	devicefarm:ListUploads
	devicefarm:ListVPCEConfigurations
	devicefarm:PurchaseOffering
	devicefarm:RenewOffering
	devicefarm:ScheduleRun
	devicefarm:StopJob

サービスプレフィックス	アクション
	devicefarm:StopRemoteAccessSession
	devicefarm:StopRun
	devicefarm:UpdateDeviceInstance
	devicefarm:UpdateDevicePool
	devicefarm:UpdateInstanceProfile
	devicefarm:UpdateNetworkProfile
	devicefarm:UpdateProject
	devicefarm:UpdateTestGridProject
	devicefarm:UpdateUpload
	devicefarm:UpdateVPCEConfiguration

サービスプレフィックス	アクション
devops-guru	devops-guru:AddNotificationChannel
	devops-guru:DeleteInsight
	devops-guru:DescribeAccountHealth
	devops-guru:DescribeAccountOverview
	devops-guru:DescribeAnomaly
	devops-guru:DescribeEventSourcesConfig
	devops-guru:DescribeFeedback
	devops-guru:DescribeInsight
	devops-guru:DescribeOrganizationHealth
	devops-guru:DescribeOrganizationOverview
	devops-guru:DescribeOrganizationResourceCollectionHealth
	devops-guru:DescribeResourceCollectionHealth
	devops-guru:DescribeServiceIntegration
	devops-guru:GetCostEstimation
	devops-guru:GetResourceCollection
	devops-guru:ListAnomaliesForInsight
	devops-guru:ListAnomalousLogGroups
	devops-guru:ListEvents
	devops-guru:ListInsights
	devops-guru:ListMonitoredResources
	devops-guru:ListNotificationChannels

サービスプレフィックス	アクション
	devops-guru:ListOrganizationInsights
	devops-guru:ListRecommendations
	devops-guru:PutFeedback
	devops-guru:RemoveNotificationChannel
	devops-guru:SearchInsights
	devops-guru:SearchOrganizationInsights
	devops-guru:StartCostEstimation
	devops-guru:UpdateEventSourcesConfig
	devops-guru:UpdateResourceCollection
	devops-guru:UpdateServiceIntegration

サービスプレフィックス	アクション
directconnect	directconnect:AcceptDirectConnectGatewayAssociationProposal
	directconnect:AllocateConnectionOnInterconnect
	directconnect:AllocateHostedConnection
	directconnect:AllocatePrivateVirtualInterface
	directconnect:AllocatePublicVirtualInterface
	directconnect:AllocateTransitVirtualInterface
	directconnect:AssociateConnectionWithLag
	directconnect:AssociateHostedConnection
	directconnect:AssociateMacSecKey
	directconnect:AssociateVirtualInterface
	directconnect:ConfirmConnection
	directconnect:ConfirmCustomerAgreement
	directconnect:ConfirmPrivateVirtualInterface
	directconnect:ConfirmPublicVirtualInterface
	directconnect:ConfirmTransitVirtualInterface
	directconnect:CreateBGPPeer
	directconnect:CreateConnection
	directconnect:CreateDirectConnectGateway
	directconnect:CreateDirectConnectGatewayAssociation
	directconnect:CreateDirectConnectGatewayAssociationProposal
	directconnect:CreateInterconnect

サービスプレフィックス	アクション
	<p>directconnect:CreateLag</p> <p>directconnect:CreatePrivateVirtualInterface</p> <p>directconnect:CreatePublicVirtualInterface</p> <p>directconnect:CreateTransitVirtualInterface</p> <p>directconnect&gt;DeleteBGPPeer</p> <p>directconnect&gt;DeleteConnection</p> <p>directconnect&gt;DeleteDirectConnectGateway</p> <p>directconnect&gt;DeleteDirectConnectGatewayAssociation</p> <p>directconnect&gt;DeleteDirectConnectGatewayAssociationProposal</p> <p>directconnect&gt;DeleteInterconnect</p> <p>directconnect&gt;DeleteLag</p> <p>directconnect&gt;DeleteVirtualInterface</p> <p>directconnect:DescribeConnectionLoa</p> <p>directconnect:DescribeConnections</p> <p>directconnect:DescribeConnectionsOnInterconnect</p> <p>directconnect:DescribeCustomerMetadata</p> <p>directconnect:DescribeDirectConnectGatewayAssociationProposals</p> <p>directconnect:DescribeDirectConnectGatewayAssociations</p> <p>directconnect:DescribeDirectConnectGatewayAttachments</p> <p>directconnect:DescribeDirectConnectGateways</p> <p>directconnect:DescribeHostedConnections</p>



サービスプレフィックス	アクション
	<p>directconnect:DescribeInterconnectLoa</p> <p>directconnect:DescribeInterconnects</p> <p>directconnect:DescribeLags</p> <p>directconnect:DescribeLoa</p> <p>directconnect:DescribeLocations</p> <p>directconnect:DescribeRouterConfiguration</p> <p>directconnect:DescribeVirtualGateways</p> <p>directconnect:DescribeVirtualInterfaces</p> <p>directconnect:DisassociateConnectionFromLag</p> <p>directconnect:DisassociateMacSecKey</p> <p>directconnect:ListVirtualInterfaceTestHistory</p> <p>directconnect:StartBgpFailoverTest</p> <p>directconnect:StopBgpFailoverTest</p> <p>directconnect:UpdateConnection</p> <p>directconnect:UpdateDirectConnectGateway</p> <p>directconnect:UpdateDirectConnectGatewayAssociation</p> <p>directconnect:UpdateLag</p> <p>directconnect:UpdateVirtualInterfaceAttributes</p>

サービスプレフィックス	アクション
dlm	dlm:CreateLifecyclePolicy dlm>DeleteLifecyclePolicy dlm:GetLifecyclePolicies dlm:GetLifecyclePolicy dlm:UpdateLifecyclePolicy

サービスプレフィックス	アクション
dms	dms:ApplyPendingMaintenanceAction
	dms:BatchStartRecommendations
	dms:CancelReplicationTaskAssessmentRun
	dms:CreateDataProvider
	dms:CreateEndpoint
	dms:CreateEventSubscription
	dms:CreateInstanceProfile
	dms:CreateMigrationProject
	dms:CreateReplicationConfig
	dms:CreateReplicationInstance
	dms:CreateReplicationSubnetGroup
	dms:CreateReplicationTask
	dms>DeleteCertificate
	dms>DeleteConnection
	dms>DeleteDataProvider
	dms>DeleteEndpoint
	dms>DeleteEventSubscription
	dms>DeleteFleetAdvisorCollector
	dms>DeleteFleetAdvisorDatabases
	dms>DeleteInstanceProfile
	dms>DeleteMigrationProject

サービスプレフィックス	アクション
	dms:DeleteReplicationConfig
	dms:DeleteReplicationInstance
	dms:DeleteReplicationSubnetGroup
	dms:DeleteReplicationTask
	dms:DeleteReplicationTaskAssessmentRun
	dms:DescribeAccountAttributes
	dms:DescribeApplicableIndividualAssessments
	dms:DescribeCertificates
	dms:DescribeConnections
	dms:DescribeEndpoints
	dms:DescribeEndpointSettings
	dms:DescribeEndpointTypes
	dms:DescribeEngineVersions
	dms:DescribeEventCategories
	dms:DescribeEvents
	dms:DescribeEventSubscriptions
	dms:DescribeFleetAdvisorCollectors
	dms:DescribeFleetAdvisorDatabases
	dms:DescribeFleetAdvisorLsaAnalysis
	dms:DescribeFleetAdvisorSchemaObjectSummary
	dms:DescribeFleetAdvisorSchemas

サービスプレフィックス	アクション
	dms:DescribeMetadataModelImports
	dms:DescribeOrderableReplicationInstances
	dms:DescribePendingMaintenanceActions
	dms:DescribeRecommendationLimitations
	dms:DescribeRecommendations
	dms:DescribeRefreshSchemasStatus
	dms:DescribeReplicationConfigs
	dms:DescribeReplicationInstances
	dms:DescribeReplicationInstanceTaskLogs
	dms:DescribeReplications
	dms:DescribeReplicationSubnetGroups
	dms:DescribeReplicationTableStatistics
	dms:DescribeReplicationTaskAssessmentResults
	dms:DescribeReplicationTaskAssessmentRuns
	dms:DescribeReplicationTaskIndividualAssessments
	dms:DescribeReplicationTasks
	dms:DescribeSchemas
	dms:DescribeTableStatistics
	dms:ExportMetadataModelAssessment
	dms:GetMetadataModel
	dms:ImportCertificate

サービスプレフィックス	アクション
	dms:ListMetadataModelAssessmentActionItems
	dms:ModifyEndpoint
	dms:ModifyEventSubscription
	dms:ModifyReplicationConfig
	dms:ModifyReplicationInstance
	dms:ModifyReplicationSubnetGroup
	dms:ModifyReplicationTask
	dms:MoveReplicationTask
	dms:RebootReplicationInstance
	dms:RefreshSchemas
	dms:ReloadReplicationTables
	dms:ReloadTables
	dms:RunFleetAdvisorLsaAnalysis
	dms:StartMetadataModelAssessment
	dms:StartMetadataModelConversion
	dms:StartMetadataModelExportToTarget
	dms:StartRecommendations
	dms:StartReplication
	dms:StartReplicationTask
	dms:StartReplicationTaskAssessment
	dms:StopReplicationTask

サービスプレフィックス	アクション
	dms:TestConnection
	dms:UpdateSubscriptionsToEventBridge
docdb-elastic	docdb-elastic:CopyClusterSnapshot
	docdb-elastic>DeleteCluster
	docdb-elastic>DeleteClusterSnapshot
	docdb-elastic:GetCluster
	docdb-elastic:GetClusterSnapshot
	docdb-elastic>ListClusters
	docdb-elastic>ListClusterSnapshots
	docdb-elastic:RestoreClusterFromSnapshot
	docdb-elastic:StartCluster
	docdb-elastic:StopCluster
	docdb-elastic:UpdateCluster

サービスプレフィックス	アクション
ds	ds:AcceptSharedDirectory
	ds:AddIpRoutes
	ds:AddRegion
	ds:CancelSchemaExtension
	ds:ConnectDirectory
	ds:CreateAlias
	ds:CreateComputer
	ds:CreateConditionalForwarder
	ds:CreateDirectory
	ds:CreateLogSubscription
	ds:CreateMicrosoftAD
	ds:CreateSnapshot
	ds:CreateTrust
	ds>DeleteConditionalForwarder
	ds>DeleteDirectory
	ds>DeleteLogSubscription
	ds>DeleteSnapshot
	ds>DeleteTrust
	ds:DeregisterCertificate
	ds:DeregisterEventTopic
	ds:DescribeCertificate



サービスプレフィックス	アクション
	ds:DescribeClientAuthenticationSettings
	ds:DescribeConditionalForwarders
	ds:DescribeDirectories
	ds:DescribeDomainControllers
	ds:DescribeEventTopics
	ds:DescribeLDAPSSettings
	ds:DescribeRegions
	ds:DescribeSettings
	ds:DescribeSharedDirectories
	ds:DescribeSnapshots
	ds:DescribeTrusts
	ds:DescribeUpdateDirectory
	ds:DisableClientAuthentication
	ds:DisableLDAPS
	ds:DisableRadius
	ds:DisableSso
	ds:EnableClientAuthentication
	ds:EnableLDAPS
	ds:EnableRadius
	ds:EnableSso
	ds:GetDirectoryLimits

サービスプレフィックス	アクション
	ds:GetSnapshotLimits
	ds:ListCertificates
	ds:ListIpRoutes
	ds:ListLogSubscriptions
	ds:ListSchemaExtensions
	ds:RegisterCertificate
	ds:RegisterEventTopic
	ds:RejectSharedDirectory
	ds:RemoveIpRoutes
	ds:RemoveRegion
	ds:ResetUserPassword
	ds:RestoreFromSnapshot
	ds:ShareDirectory
	ds:StartSchemaExtension
	ds:UnshareDirectory
	ds:UpdateConditionalForwarder
	ds:UpdateDirectorySetup
	ds:UpdateNumberOfDomainControllers
	ds:UpdateRadius
	ds:UpdateSettings
	ds:UpdateTrust

サービスプレフィックス	アクション
	ds:VerifyTrust

サービスプレフィックス	アクション
dynamodb	dynamodb:CreateBackup
	dynamodb:CreateGlobalTable
	dynamodb:CreateTable
	dynamodb>DeleteBackup
	dynamodb>DeleteTable
	dynamodb:DescribeBackup
	dynamodb:DescribeContinuousBackups
	dynamodb:DescribeContributorInsights
	dynamodb:DescribeEndpoints
	dynamodb:DescribeExport
	dynamodb:DescribeGlobalTable
	dynamodb:DescribeGlobalTableSettings
	dynamodb:DescribeImport
	dynamodb:DescribeKinesisStreamingDestination
	dynamodb:DescribeLimits
	dynamodb:DescribeStream
	dynamodb:DescribeTable
	dynamodb:DescribeTableReplicaAutoScaling
	dynamodb:DescribeTimeToLive
	dynamodb:DisableKinesisStreamingDestination
	dynamodb:EnableKinesisStreamingDestination

サービスプレフィックス	アクション
	dynamodb:ExportTableToPointInTime
	dynamodb:GetResourcePolicy
	dynamodb:ImportTable
	dynamodb:ListBackups
	dynamodb:ListContributorInsights
	dynamodb:ListExports
	dynamodb:ListGlobalTables
	dynamodb:ListImports
	dynamodb:ListStreams
	dynamodb:ListTables
	dynamodb:RestoreTableFromBackup
	dynamodb:RestoreTableToPointInTime
	dynamodb:UpdateContinuousBackups
	dynamodb:UpdateContributorInsights
	dynamodb:UpdateGlobalTable
	dynamodb:UpdateGlobalTableSettings
	dynamodb:UpdateKinesisStreamingDestination
	dynamodb:UpdateTable
	dynamodb:UpdateTableReplicaAutoScaling
	dynamodb:UpdateTimeToLive

サービスプレフィックス	アクション
ebs	ebs:CompleteSnapshot ebs:StartSnapshot

サービスプレフィックス	アクション
Ec2	ec2:AcceptAddressTransfer
	ec2:AcceptReservedInstancesExchangeQuote
	ec2:AcceptTransitGatewayMulticastDomainAssociations
	ec2:AcceptTransitGatewayPeeringAttachment
	ec2:AcceptTransitGatewayVpcAttachment
	ec2:AcceptVpcEndpointConnections
	ec2:AcceptVpcPeeringConnection
	ec2:AdvertiseByoipCidr
	ec2:AllocateAddress
	ec2:AllocateHosts
	ec2:AllocateIpamPoolCidr
	ec2:ApplySecurityGroupsToClientVpnTargetNetwork
	ec2:AssignIpv6Addresses
	ec2:AssignPrivateIpAddresses
	ec2:AssignPrivateNatGatewayAddress
	ec2:AssociateAddress
	ec2:AssociateClientVpnTargetNetwork
	ec2:AssociateDhcpOptions
	ec2:AssociateEnclaveCertificateIamRole
	ec2:AssociateIamInstanceProfile
	ec2:AssociateInstanceEventWindow

サービスプレフィックス	アクション
	ec2:AssociateIpamByoasn
	ec2:AssociateIpamResourceDiscovery
	ec2:AssociateNatGatewayAddress
	ec2:AssociateRouteTable
	ec2:AssociateSubnetCidrBlock
	ec2:AssociateTransitGatewayMulticastDomain
	ec2:AssociateTransitGatewayPolicyTable
	ec2:AssociateTransitGatewayRouteTable
	ec2:AssociateTrunkInterface
	ec2:AssociateVpcCidrBlock
	ec2:AttachClassicLinkVpc
	ec2:AttachInternetGateway
	ec2:AttachNetworkInterface
	ec2:AttachVerifiedAccessTrustProvider
	ec2:AttachVolume
	ec2:AttachVpnGateway
	ec2:AuthorizeClientVpnIngress
	ec2:AuthorizeSecurityGroupEgress
	ec2:AuthorizeSecurityGroupIngress
	ec2:BundleInstance
	ec2:CancelBundleTask



サービスプレフィックス	アクション
	ec2:CancelCapacityReservation
	ec2:CancelCapacityReservationFleets
	ec2:CancelConversionTask
	ec2:CancelExportTask
	ec2:CancellImageLaunchPermission
	ec2:CancellImportTask
	ec2:CancelReservedInstancesListing
	ec2:CancelSpotFleetRequests
	ec2:CancelSpotInstanceRequests
	ec2:ConfirmProductInstance
	ec2:CopyFpgaImage
	ec2:CopyImage
	ec2:CopySnapshot
	ec2:CreateCapacityReservation
	ec2:CreateCapacityReservationFleet
	ec2:CreateCarrierGateway
	ec2:CreateClientVpnEndpoint
	ec2:CreateClientVpnRoute
	ec2:CreateCoipCidr
	ec2:CreateCoipPool
	ec2:CreateCustomerGateway

サービスプレフィックス	アクション
	ec2:CreateDefaultSubnet
	ec2:CreateDefaultVpc
	ec2:CreateDhcpOptions
	ec2:CreateEgressOnlyInternetGateway
	ec2:CreateFleet
	ec2:CreateFlowLogs
	ec2:CreateFpgaImage
	ec2:CreateImage
	ec2:CreateInstanceConnectEndpoint
	ec2:CreateInstanceEventWindow
	ec2:CreateInstanceExportTask
	ec2:CreateInternetGateway
	ec2:CreateIam
	ec2:CreateIamPool
	ec2:CreateIamResourceDiscovery
	ec2:CreateIamScope
	ec2:CreateKeyPair
	ec2:CreateLaunchTemplate
	ec2:CreateLaunchTemplateVersion
	ec2:CreateLocalGatewayRoute
	ec2:CreateLocalGatewayRouteTable

サービスプレフィックス	アクション
	ec2:CreateLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:CreateLocalGatewayRouteTableVpcAssociation
	ec2:CreateManagedPrefixList
	ec2:CreateNatGateway
	ec2:CreateNetworkAcl
	ec2:CreateNetworkAclEntry
	ec2:CreateNetworkInsightsAccessScope
	ec2:CreateNetworkInsightsPath
	ec2:CreateNetworkInterface
	ec2:CreateNetworkInterfacePermission
	ec2:CreatePlacementGroup
	ec2:CreatePublicIpv4Pool
	ec2:CreateReplaceRootVolumeTask
	ec2:CreateReservedInstancesListing
	ec2:CreateRestoreImageTask
	ec2:CreateRoute
	ec2:CreateRouteTable
	ec2:CreateSecurityGroup
	ec2:CreateSnapshot
	ec2:CreateSnapshots

サービスプレフィックス	アクション
	ec2:CreateSpotDatafeedSubscription
	ec2:CreateStoreImageTask
	ec2:CreateSubnet
	ec2:CreateSubnetCidrReservation
	ec2:CreateTrafficMirrorFilter
	ec2:CreateTrafficMirrorFilterRule
	ec2:CreateTrafficMirrorSession
	ec2:CreateTrafficMirrorTarget
	ec2:CreateTransitGateway
	ec2:CreateTransitGatewayConnect
	ec2:CreateTransitGatewayConnectPeer
	ec2:CreateTransitGatewayMulticastDomain
	ec2:CreateTransitGatewayPeeringAttachment
	ec2:CreateTransitGatewayPolicyTable
	ec2:CreateTransitGatewayPrefixListReference
	ec2:CreateTransitGatewayRoute
	ec2:CreateTransitGatewayRouteTable
	ec2:CreateTransitGatewayRouteTableAnnouncement
	ec2:CreateTransitGatewayVpcAttachment
	ec2:CreateVerifiedAccessEndpoint
	ec2:CreateVerifiedAccessGroup

サービスプレフィックス	アクション
	ec2:CreateVerifiedAccessInstance
	ec2:CreateVerifiedAccessTrustProvider
	ec2:CreateVolume
	ec2:CreateVpc
	ec2:CreateVpcEndpoint
	ec2:CreateVpcEndpointConnectionNotification
	ec2:CreateVpcEndpointServiceConfiguration
	ec2:CreateVpcPeeringConnection
	ec2:CreateVpnConnection
	ec2:CreateVpnConnectionRoute
	ec2:CreateVpnGateway
	ec2>DeleteCarrierGateway
	ec2>DeleteClientVpnEndpoint
	ec2>DeleteClientVpnRoute
	ec2>DeleteCoipCidr
	ec2>DeleteCoipPool
	ec2>DeleteCustomerGateway
	ec2>DeleteDhcpOptions
	ec2>DeleteEgressOnlyInternetGateway
	ec2>DeleteFleets
	ec2>DeleteFlowLogs

サービスプレフィックス	アクション
	ec2:DeleteFpgaImage
	ec2:DeleteInstanceConnectEndpoint
	ec2:DeleteInstanceEventWindow
	ec2:DeleteInternetGateway
	ec2:DeleteIam
	ec2:DeleteIamPool
	ec2:DeleteIamResourceDiscovery
	ec2:DeleteIamScope
	ec2>DeleteKeyPair
	ec2>DeleteLaunchTemplate
	ec2>DeleteLaunchTemplateVersions
	ec2>DeleteLocalGatewayRoute
	ec2>DeleteLocalGatewayRouteTable
	ec2>DeleteLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2>DeleteLocalGatewayRouteTableVpcAssociation
	ec2>DeleteManagedPrefixList
	ec2>DeleteNatGateway
	ec2>DeleteNetworkAcl
	ec2>DeleteNetworkAclEntry
	ec2>DeleteNetworkInsightsAccessScope

サービスプレフィックス	アクション
	ec2:DeleteNetworkInsightsAccessScopeAnalysis
	ec2:DeleteNetworkInsightsAnalysis
	ec2:DeleteNetworkInsightsPath
	ec2:DeleteNetworkInterface
	ec2:DeleteNetworkInterfacePermission
	ec2:DeletePlacementGroup
	ec2:DeletePublicIpv4Pool
	ec2:DeleteQueuedReservedInstances
	ec2:DeleteRoute
	ec2:DeleteRouteTable
	ec2:DeleteSecurityGroup
	ec2:DeleteSnapshot
	ec2:DeleteSpotDatafeedSubscription
	ec2:DeleteSubnet
	ec2:DeleteSubnetCidrReservation
	ec2:DeleteTrafficMirrorFilter
	ec2:DeleteTrafficMirrorFilterRule
	ec2:DeleteTrafficMirrorSession
	ec2:DeleteTrafficMirrorTarget
	ec2:DeleteTransitGateway
	ec2:DeleteTransitGatewayConnect

サービスプレフィックス	アクション
	ec2:DeleteTransitGatewayConnectPeer
	ec2:DeleteTransitGatewayMulticastDomain
	ec2:DeleteTransitGatewayPeeringAttachment
	ec2:DeleteTransitGatewayPolicyTable
	ec2:DeleteTransitGatewayPrefixListReference
	ec2:DeleteTransitGatewayRoute
	ec2:DeleteTransitGatewayRouteTable
	ec2:DeleteTransitGatewayRouteTableAnnouncement
	ec2:DeleteTransitGatewayVpcAttachment
	ec2:DeleteVerifiedAccessEndpoint
	ec2:DeleteVerifiedAccessGroup
	ec2:DeleteVerifiedAccessInstance
	ec2:DeleteVerifiedAccessTrustProvider
	ec2:DeleteVolume
	ec2:DeleteVpc
	ec2:DeleteVpcEndpointConnectionNotifications
	ec2:DeleteVpcEndpoints
	ec2:DeleteVpcEndpointServiceConfigurations
	ec2:DeleteVpcPeeringConnection
	ec2:DeleteVpnConnection
	ec2:DeleteVpnConnectionRoute



サービスプレフィックス	アクション
	ec2:DeleteVpnGateway
	ec2:DeprovisionByoipCidr
	ec2:DeprovisionIpamByoasn
	ec2:DeprovisionIpamPoolCidr
	ec2:DeprovisionPublicIpv4PoolCidr
	ec2:DeregisterImage
	ec2:DeregisterInstanceEventNotificationAttributes
	ec2:DeregisterTransitGatewayMulticastGroupMembers
	ec2:DeregisterTransitGatewayMulticastGroupSources
	ec2:DescribeAccountAttributes
	ec2:DescribeAddresses
	ec2:DescribeAddressesAttribute
	ec2:DescribeAddressTransfers
	ec2:DescribeAggregateIdFormat
	ec2:DescribeAvailabilityZones
	ec2:DescribeAwsNetworkPerformanceMetricSubscriptions
	ec2:DescribeBundleTasks
	ec2:DescribeByoipCidrs
	ec2:DescribeCapacityReservationFleets
	ec2:DescribeCapacityReservations
	ec2:DescribeCarrierGateways

サービスプレフィックス	アクション
	ec2:DescribeClassicLinkInstances
	ec2:DescribeClientVpnAuthorizationRules
	ec2:DescribeClientVpnConnections
	ec2:DescribeClientVpnEndpoints
	ec2:DescribeClientVpnRoutes
	ec2:DescribeClientVpnTargetNetworks
	ec2:DescribeCoipPools
	ec2:DescribeConversionTasks
	ec2:DescribeCustomerGateways
	ec2:DescribeDhcpOptions
	ec2:DescribeEgressOnlyInternetGateways
	ec2:DescribeElasticGpus
	ec2:DescribeExportImageTasks
	ec2:DescribeExportTasks
	ec2:DescribeFastLaunchImages
	ec2:DescribeFastSnapshotRestores
	ec2:DescribeFleetHistory
	ec2:DescribeFleetInstances
	ec2:DescribeFleets
	ec2:DescribeFlowLogs
	ec2:DescribeFpgaImageAttribute

サービスプレフィックス	アクション
	ec2:DescribeFpgaImages
	ec2:DescribeHostReservationOfferings
	ec2:DescribeHostReservations
	ec2:DescribeHosts
	ec2:DescribeIamInstanceProfileAssociations
	ec2:DescribeIdentityIdFormat
	ec2:DescribeIdFormat
	ec2:DescribeImageAttribute
	ec2:DescribeImages
	ec2:DescribeImportImageTasks
	ec2:DescribeImportSnapshotTasks
	ec2:DescribeInstanceAttribute
	ec2:DescribeInstanceConnectEndpoints
	ec2:DescribeInstanceCreditSpecifications
	ec2:DescribeInstanceEventNotificationAttributes
	ec2:DescribeInstanceEventWindows
	ec2:DescribeInstances
	ec2:DescribeInstanceStatus
	ec2:DescribeInstanceTopology
	ec2:DescribeInstanceTypeOfferings
	ec2:DescribeInstanceTypes

サービスプレフィックス	アクション
	ec2:DescribeInternetGateways
	ec2:DescribeIpamByoasn
	ec2:DescribeIpamPools
	ec2:DescribeIpamResourceDiscoveries
	ec2:DescribeIpamResourceDiscoveryAssociations
	ec2:DescribeIpams
	ec2:DescribeIpamScopes
	ec2:DescribeIpv6Pools
	ec2:DescribeKeyPairs
	ec2:DescribeLaunchTemplates
	ec2:DescribeLaunchTemplateVersions
	ec2:DescribeLocalGatewayRouteTables
	ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations
	ec2:DescribeLocalGatewayRouteTableVpcAssociations
	ec2:DescribeLocalGateways
	ec2:DescribeLocalGatewayVirtualInterfaceGroups
	ec2:DescribeLocalGatewayVirtualInterfaces
	ec2:DescribeLockedSnapshots
	ec2:DescribeMacHosts
	ec2:DescribeManagedPrefixLists

サービスプレフィックス	アクション
	ec2:DescribeMovingAddresses
	ec2:DescribeNatGateways
	ec2:DescribeNetworkAcls
	ec2:DescribeNetworkInsightsAccessScopeAnalyses
	ec2:DescribeNetworkInsightsAccessScopes
	ec2:DescribeNetworkInsightsAnalyses
	ec2:DescribeNetworkInsightsPaths
	ec2:DescribeNetworkInterfaceAttribute
	ec2:DescribeNetworkInterfacePermissions
	ec2:DescribeNetworkInterfaces
	ec2:DescribePlacementGroups
	ec2:DescribePrefixLists
	ec2:DescribePrincipalIdFormat
	ec2:DescribePublicIpv4Pools
	ec2:DescribeRegions
	ec2:DescribeReplaceRootVolumeTasks
	ec2:DescribeReservedInstances
	ec2:DescribeReservedInstancesListings
	ec2:DescribeReservedInstancesModifications
	ec2:DescribeReservedInstancesOfferings
	ec2:DescribeRouteTables

サービスプレフィックス	アクション
	ec2:DescribeScheduledInstanceAvailability
	ec2:DescribeScheduledInstances
	ec2:DescribeSecurityGroupReferences
	ec2:DescribeSecurityGroupRules
	ec2:DescribeSecurityGroups
	ec2:DescribeSnapshotAttribute
	ec2:DescribeSnapshots
	ec2:DescribeSnapshotTierStatus
	ec2:DescribeSpotDatafeedSubscription
	ec2:DescribeSpotFleetInstances
	ec2:DescribeSpotFleetRequestHistory
	ec2:DescribeSpotFleetRequests
	ec2:DescribeSpotInstanceRequests
	ec2:DescribeSpotPriceHistory
	ec2:DescribeStaleSecurityGroups
	ec2:DescribeStoreImageTasks
	ec2:DescribeSubnets
	ec2:DescribeTrafficMirrorFilters
	ec2:DescribeTrafficMirrorSessions
	ec2:DescribeTrafficMirrorTargets
	ec2:DescribeTransitGatewayAttachments

サービスプレフィックス	アクション
	ec2:DescribeTransitGatewayConnectPeers
	ec2:DescribeTransitGatewayConnects
	ec2:DescribeTransitGatewayMulticastDomains
	ec2:DescribeTransitGatewayPeeringAttachments
	ec2:DescribeTransitGatewayPolicyTables
	ec2:DescribeTransitGatewayRouteTableAnnouncements
	ec2:DescribeTransitGatewayRouteTables
	ec2:DescribeTransitGateways
	ec2:DescribeTransitGatewayVpcAttachments
	ec2:DescribeTrunkInterfaceAssociations
	ec2:DescribeVerifiedAccessEndpoints
	ec2:DescribeVerifiedAccessGroups
	ec2:DescribeVerifiedAccessInstanceLoggingConfigurations
	ec2:DescribeVerifiedAccessInstances
	ec2:DescribeVerifiedAccessTrustProviders
	ec2:DescribeVolumeAttribute
	ec2:DescribeVolumes
	ec2:DescribeVolumesModifications
	ec2:DescribeVolumeStatus
	ec2:DescribeVpcAttribute
	ec2:DescribeVpcClassicLink

サービスプレフィックス	アクション
	ec2:DescribeVpcClassicLinkDnsSupport
	ec2:DescribeVpcEndpointConnectionNotifications
	ec2:DescribeVpcEndpointConnections
	ec2:DescribeVpcEndpoints
	ec2:DescribeVpcEndpointServiceConfigurations
	ec2:DescribeVpcEndpointServicePermissions
	ec2:DescribeVpcEndpointServices
	ec2:DescribeVpcPeeringConnections
	ec2:DescribeVpcs
	ec2:DescribeVpnConnections
	ec2:DescribeVpnGateways
	ec2:DetachClassicLinkVpc
	ec2:DetachInternetGateway
	ec2:DetachNetworkInterface
	ec2:DetachVerifiedAccessTrustProvider
	ec2:DetachVolume
	ec2:DetachVpnGateway
	ec2:DisableAddressTransfer
	ec2:DisableAwsNetworkPerformanceMetricSubscription
	ec2:DisableEbsEncryptionByDefault
	ec2:DisableFastLaunch



サービスプレフィックス	アクション
	ec2:DisableFastSnapshotRestores
	ec2:DisableImage
	ec2:DisableImageBlockPublicAccess
	ec2:DisableImageDeprecation
	ec2:DisableImageDeregistrationProtection
	ec2:DisableIpamOrganizationAdminAccount
	ec2:DisableSerialConsoleAccess
	ec2:DisableSnapshotBlockPublicAccess
	ec2:DisableTransitGatewayRouteTablePropagation
	ec2:DisableVgwRoutePropagation
	ec2:DisableVpcClassicLink
	ec2:DisableVpcClassicLinkDnsSupport
	ec2:DisassociateAddress
	ec2:DisassociateClientVpnTargetNetwork
	ec2:DisassociateEnclaveCertificateIamRole
	ec2:DisassociateIamInstanceProfile
	ec2:DisassociateInstanceEventWindow
	ec2:DisassociateIpamByoasn
	ec2:DisassociateIpamResourceDiscovery
	ec2:DisassociateNatGatewayAddress
	ec2:DisassociateRouteTable

サービスプレフィックス	アクション
	ec2:DisassociateSubnetCidrBlock
	ec2:DisassociateTransitGatewayMulticastDomain
	ec2:DisassociateTransitGatewayPolicyTable
	ec2:DisassociateTransitGatewayRouteTable
	ec2:DisassociateTrunkInterface
	ec2:DisassociateVpcCidrBlock
	ec2:EnableAddressTransfer
	ec2:EnableAwsNetworkPerformanceMetricSubscription
	ec2:EnableEbsEncryptionByDefault
	ec2:EnableFastLaunch
	ec2:EnableFastSnapshotRestores
	ec2:EnableImage
	ec2:EnableImageBlockPublicAccess
	ec2:EnableImageDeprecation
	ec2:EnableImageDeregistrationProtection
	ec2:EnableIamOrganizationAdminAccount
	ec2:EnableReachabilityAnalyzerOrganizationSharing
	ec2:EnableSerialConsoleAccess
	ec2:EnableSnapshotBlockPublicAccess
	ec2:EnableTransitGatewayRouteTablePropagation
	ec2:EnableVgwRoutePropagation

サービスプレフィックス	アクション
	ec2:EnableVolumeIO
	ec2:EnableVpcClassicLink
	ec2:EnableVpcClassicLinkDnsSupport
	ec2:ExportClientVpnClientCertificateRevocationList
	ec2:ExportClientVpnClientConfiguration
	ec2:ExportImage
	ec2:ExportTransitGatewayRoutes
	ec2:GetAssociatedEnclaveCertificateIamRoles
	ec2:GetAssociatedIpv6PoolCidrs
	ec2:GetAwsNetworkPerformanceData
	ec2:GetCapacityReservationUsage
	ec2:GetCoipPoolUsage
	ec2:GetConsoleOutput
	ec2:GetConsoleScreenshot
	ec2:GetDefaultCreditSpecification
	ec2:GetEbsDefaultKmsKeyId
	ec2:GetEbsEncryptionByDefault
	ec2:GetFlowLogsIntegrationTemplate
	ec2:GetGroupsForCapacityReservation
	ec2:GetHostReservationPurchasePreview
	ec2:GetImageBlockPublicAccessState

サービスプレフィックス	アクション
	ec2:GetInstanceMetadataDefaults
	ec2:GetInstanceTpmEkPub
	ec2:GetInstanceTypesFromInstanceRequirements
	ec2:GetInstanceUefiData
	ec2:GetIamAddressHistory
	ec2:GetIamDiscoveredAccounts
	ec2:GetIamDiscoveredPublicAddresses
	ec2:GetIamDiscoveredResourceCidrs
	ec2:GetIamPoolAllocations
	ec2:GetIamPoolCidrs
	ec2:GetIamResourceCidrs
	ec2:GetLaunchTemplateData
	ec2:GetManagedPrefixListAssociations
	ec2:GetManagedPrefixListEntries
	ec2:GetNetworkInsightsAccessScopeAnalysisFindings
	ec2:GetNetworkInsightsAccessScopeContent
	ec2:GetPasswordData
	ec2:GetReservedInstancesExchangeQuote
	ec2:GetSecurityGroupsForVpc
	ec2:GetSerialConsoleAccessStatus
	ec2:GetSnapshotBlockPublicAccessState

サービスプレフィックス	アクション
	ec2:GetSpotPlacementScores
	ec2:GetSubnetCidrReservations
	ec2:GetTransitGatewayAttachmentPropagations
	ec2:GetTransitGatewayMulticastDomainAssociations
	ec2:GetTransitGatewayPolicyTableAssociations
	ec2:GetTransitGatewayPolicyTableEntries
	ec2:GetTransitGatewayPrefixListReferences
	ec2:GetTransitGatewayRouteTableAssociations
	ec2:GetTransitGatewayRouteTablePropagations
	ec2:GetVerifiedAccessEndpointPolicy
	ec2:GetVerifiedAccessGroupPolicy
	ec2:GetVpnConnectionDeviceSampleConfiguration
	ec2:GetVpnConnectionDeviceTypes
	ec2:GetVpnTunnelReplacementStatus
	ec2:ImportClientVpnClientCertificateRevocationList
	ec2:ImportImage
	ec2:ImportInstance
	ec2:ImportKeyPair
	ec2:ImportSnapshot
	ec2:ImportVolume
	ec2:ListImagesInRecycleBin

サービスプレフィックス	アクション
	ec2:ListSnapshotsInRecycleBin
	ec2:LockSnapshot
	ec2:ModifyAddressAttribute
	ec2:ModifyAvailabilityZoneGroup
	ec2:ModifyCapacityReservation
	ec2:ModifyCapacityReservationFleet
	ec2:ModifyClientVpnEndpoint
	ec2:ModifyDefaultCreditSpecification
	ec2:ModifyEbsDefaultKmsKeyId
	ec2:ModifyFleet
	ec2:ModifyFpgaImageAttribute
	ec2:ModifyHosts
	ec2:ModifyIdentityIdFormat
	ec2:ModifyIdFormat
	ec2:ModifyImageAttribute
	ec2:ModifyInstanceAttribute
	ec2:ModifyInstanceCapacityReservationAttributes
	ec2:ModifyInstanceCreditSpecification
	ec2:ModifyInstanceEventStartTime
	ec2:ModifyInstanceEventWindow
	ec2:ModifyInstanceMaintenanceOptions

サービスプレフィックス	アクション
	ec2:ModifyInstanceMetadataDefaults
	ec2:ModifyInstanceMetadataOptions
	ec2:ModifyInstancePlacement
	ec2:ModifyIpam
	ec2:ModifyIpamPool
	ec2:ModifyIpamResourceCidr
	ec2:ModifyIpamResourceDiscovery
	ec2:ModifyIpamScope
	ec2:ModifyLaunchTemplate
	ec2:ModifyLocalGatewayRoute
	ec2:ModifyManagedPrefixList
	ec2:ModifyNetworkInterfaceAttribute
	ec2:ModifyPrivateDnsNameOptions
	ec2:ModifyReservedInstances
	ec2:ModifySecurityGroupRules
	ec2:ModifySnapshotAttribute
	ec2:ModifySnapshotTier
	ec2:ModifySpotFleetRequest
	ec2:ModifySubnetAttribute
	ec2:ModifyTrafficMirrorFilterNetworkServices
	ec2:ModifyTrafficMirrorFilterRule

サービスプレフィックス	アクション
	ec2:ModifyTrafficMirrorSession
	ec2:ModifyTransitGateway
	ec2:ModifyTransitGatewayPrefixListReference
	ec2:ModifyTransitGatewayVpcAttachment
	ec2:ModifyVerifiedAccessEndpoint
	ec2:ModifyVerifiedAccessEndpointPolicy
	ec2:ModifyVerifiedAccessGroup
	ec2:ModifyVerifiedAccessGroupPolicy
	ec2:ModifyVerifiedAccessInstance
	ec2:ModifyVerifiedAccessInstanceLoggingConfiguration
	ec2:ModifyVerifiedAccessTrustProvider
	ec2:ModifyVolume
	ec2:ModifyVolumeAttribute
	ec2:ModifyVpcAttribute
	ec2:ModifyVpcEndpoint
	ec2:ModifyVpcEndpointConnectionNotification
	ec2:ModifyVpcEndpointServiceConfiguration
	ec2:ModifyVpcEndpointServicePayerResponsibility
	ec2:ModifyVpcEndpointServicePermissions
	ec2:ModifyVpcPeeringConnectionOptions
	ec2:ModifyVpcTenancy



サービスプレフィックス	アクション
	ec2:ModifyVpnConnection
	ec2:ModifyVpnConnectionOptions
	ec2:ModifyVpnTunnelCertificate
	ec2:ModifyVpnTunnelOptions
	ec2:MonitorInstances
	ec2:MoveAddressToVpc
	ec2:MoveByoipCidrToIpam
	ec2:ProvisionByoipCidr
	ec2:ProvisionIpamByoasn
	ec2:ProvisionIpamPoolCidr
	ec2:ProvisionPublicIpv4PoolCidr
	ec2:PurchaseHostReservation
	ec2:PurchaseReservedInstancesOffering
	ec2:PurchaseScheduledInstances
	ec2:RebootInstances
	ec2:RegisterImage
	ec2:RegisterInstanceEventNotificationAttributes
	ec2:RegisterTransitGatewayMulticastGroupMembers
	ec2:RegisterTransitGatewayMulticastGroupSources
	ec2:RejectTransitGatewayMulticastDomainAssociations
	ec2:RejectTransitGatewayPeeringAttachment

サービスプレフィックス	アクション
	ec2:RejectTransitGatewayVpcAttachment
	ec2:RejectVpcEndpointConnections
	ec2:RejectVpcPeeringConnection
	ec2:ReleaseAddress
	ec2:ReleaseHosts
	ec2:ReleaseIpamPoolAllocation
	ec2:ReplaceIamInstanceProfileAssociation
	ec2:ReplaceNetworkAclAssociation
	ec2:ReplaceNetworkAclEntry
	ec2:ReplaceRoute
	ec2:ReplaceRouteTableAssociation
	ec2:ReplaceTransitGatewayRoute
	ec2:ReplaceVpnTunnel
	ec2:ReportInstanceStatus
	ec2:RequestSpotFleet
	ec2:RequestSpotInstances
	ec2:ResetAddressAttribute
	ec2:ResetEbsDefaultKmsKeyId
	ec2:ResetFpgaImageAttribute
	ec2:ResetImageAttribute
	ec2:ResetInstanceAttribute

サービスプレフィックス	アクション
	ec2:ResetNetworkInterfaceAttribute
	ec2:ResetSnapshotAttribute
	ec2:RestoreAddressToClassic
	ec2:RestoreImageFromRecycleBin
	ec2:RestoreManagedPrefixListVersion
	ec2:RestoreSnapshotFromRecycleBin
	ec2:RestoreSnapshotTier
	ec2:RevokeClientVpnIngress
	ec2:RevokeSecurityGroupEgress
	ec2:RevokeSecurityGroupIngress
	ec2:RunInstances
	ec2:RunScheduledInstances
	ec2:SearchLocalGatewayRoutes
	ec2:SearchTransitGatewayMulticastGroups
	ec2:SearchTransitGatewayRoutes
	ec2:SendDiagnosticInterrupt
	ec2:StartInstances
	ec2:StartNetworkInsightsAccessScopeAnalysis
	ec2:StartNetworkInsightsAnalysis
	ec2:StartVpcEndpointServicePrivateDnsVerification
	ec2:StopInstances

サービスプレフィックス	アクション
	ec2:TerminateClientVpnConnections
	ec2:TerminateInstances
	ec2:UnassignIpv6Addresses
	ec2:UnassignPrivateIpAddresses
	ec2:UnassignPrivateNatGatewayAddress
	ec2:UnlockSnapshot
	ec2:UnmonitorInstances
	ec2:UpdateSecurityGroupRuleDescriptionsEgress
	ec2:UpdateSecurityGroupRuleDescriptionsIngress
	ec2:WithdrawByoipCidr

サービスプレフィックス	アクション
ecr	ecr:BatchCheckLayerAvailability
	ecr:BatchDeleteImage
	ecr:BatchGetImage
	ecr:BatchGetRepositoryScanningConfiguration
	ecr:CompleteLayerUpload
	ecr:CreatePullThroughCacheRule
	ecr:CreateRepository
	ecr:CreateRepositoryCreationTemplate
	ecr>DeleteLifecyclePolicy
	ecr>DeletePullThroughCacheRule
	ecr>DeleteRegistryPolicy
	ecr>DeleteRepository
	ecr>DeleteRepositoryCreationTemplate
	ecr>DeleteRepositoryPolicy
	ecr:DescribeImageReplicationStatus
	ecr:DescribeImages
	ecr:DescribeImageScanFindings
	ecr:DescribePullThroughCacheRules
	ecr:DescribeRegistry
	ecr:DescribeRepositories
	ecr:GetAuthorizationToken

サービスプレフィックス	アクション
	ecr:GetDownloadUriForLayer
	ecr:GetLifecyclePolicy
	ecr:GetLifecyclePolicyPreview
	ecr:GetRegistryPolicy
	ecr:GetRegistryScanningConfiguration
	ecr:GetRepositoryPolicy
	ecr:InitiateLayerUpload
	ecr:ListImages
	ecr:PutImage
	ecr:PutImageScanningConfiguration
	ecr:PutRegistryPolicy
	ecr:PutRegistryScanningConfiguration
	ecr:PutReplicationConfiguration
	ecr:StartImageScan
	ecr:StartLifecyclePolicyPreview
	ecr:UpdatePullThroughCacheRule
	ecr:UploadLayerPart
	ecr:ValidatePullThroughCacheRule

サービスプレフィックス	アクション
ecr-public	ecr-public:BatchCheckLayerAvailability
	ecr-public:BatchDeleteImage
	ecr-public:CompleteLayerUpload
	ecr-public:CreateRepository
	ecr-public>DeleteRepository
	ecr-public>DeleteRepositoryPolicy
	ecr-public:DescribeImages
	ecr-public:DescribeRegistries
	ecr-public:DescribeRepositories
	ecr-public:GetAuthorizationToken
	ecr-public:GetRegistryCatalogData
	ecr-public:GetRepositoryCatalogData
	ecr-public:GetRepositoryPolicy
	ecr-public:InitiateLayerUpload
	ecr-public:PutImage
	ecr-public:PutRegistryCatalogData
	ecr-public:PutRepositoryCatalogData
	ecr-public:SetRepositoryPolicy
	ecr-public:UploadLayerPart

サービスプレフィックス	アクション
Ecs	ecs:CreateCapacityProvider
	ecs:CreateCluster
	ecs:CreateService
	ecs:CreateTaskSet
	ecs>DeleteAccountSetting
	ecs>DeleteAttributes
	ecs>DeleteCapacityProvider
	ecs>DeleteCluster
	ecs>DeleteService
	ecs>DeleteTaskDefinitions
	ecs>DeleteTaskSet
	ecs:DeregisterContainerInstance
	ecs:DeregisterTaskDefinition
	ecs:DescribeCapacityProviders
	ecs:DescribeClusters
	ecs:DescribeContainerInstances
	ecs:DescribeServices
	ecs:DescribeTaskDefinition
	ecs:DescribeTasks
	ecs:DescribeTaskSets
	ecs:DiscoverPollEndpoint



サービスプレフィックス	アクション
	ecs:ExecuteCommand
	ecs:GetTaskProtection
	ecs:ListAccountSettings
	ecs:ListAttributes
	ecs:ListClusters
	ecs:ListContainerInstances
	ecs:ListServices
	ecs:ListServicesByNamespace
	ecs:ListTaskDefinitionFamilies
	ecs:ListTaskDefinitions
	ecs:ListTasks
	ecs:PutAccountSetting
	ecs:PutAccountSettingDefault
	ecs:PutAttributes
	ecs:PutClusterCapacityProviders
	ecs:RegisterContainerInstance
	ecs:RegisterTaskDefinition
	ecs:RunTask
	ecs:StartTask
	ecs:StopTask
	ecs:SubmitAttachmentStateChanges

サービスプレフィックス	アクション
	ecs:SubmitContainerStateChange
	ecs:SubmitTaskStateChange
	ecs:UpdateCapacityProvider
	ecs:UpdateCluster
	ecs:UpdateClusterSettings
	ecs:UpdateContainerAgent
	ecs:UpdateContainerInstancesState
	ecs:UpdateService
	ecs:UpdateServicePrimaryTaskSet
	ecs:UpdateTaskProtection
	ecs:UpdateTaskSet

サービスプレフィックス	アクション
eks	eks:AssociateAccessPolicy
	eks:AssociateEncryptionConfig
	eks:AssociateIdentityProviderConfig
	eks:CreateAccessEntry
	eks:CreateAddon
	eks:CreateCluster
	eks:CreateEksAnywhereSubscription
	eks:CreateFargateProfile
	eks:CreateNodegroup
	eks>DeleteAccessEntry
	eks>DeleteAddon
	eks>DeleteCluster
	eks>DeleteEksAnywhereSubscription
	eks>DeleteFargateProfile
	eks>DeleteNodegroup
	eks>DeletePodIdentityAssociation
	eks:DeregisterCluster
	eks:DescribeAccessEntry
	eks:DescribeAddon
	eks:DescribeAddonConfiguration
	eks:DescribeAddonVersions

サービスプレフィックス	アクション
	<code>eks:DescribeCluster</code>
	<code>eks:DescribeEksAnywhereSubscription</code>
	<code>eks:DescribeFargateProfile</code>
	<code>eks:DescribeIdentityProviderConfig</code>
	<code>eks:DescribeInsight</code>
	<code>eks:DescribeNodegroup</code>
	<code>eks:DescribePodIdentityAssociation</code>
	<code>eks:DescribeUpdate</code>
	<code>eks:DisassociateAccessPolicy</code>
	<code>eks:DisassociateIdentityProviderConfig</code>
	<code>eks:ListAccessEntries</code>
	<code>eks:ListAccessPolicies</code>
	<code>eks:ListAddons</code>
	<code>eks:ListAssociatedAccessPolicies</code>
	<code>eks:ListClusters</code>
	<code>eks:ListEksAnywhereSubscriptions</code>
	<code>eks:ListFargateProfiles</code>
	<code>eks:ListIdentityProviderConfigs</code>
	<code>eks:ListInsights</code>
	<code>eks:ListNodegroups</code>
	<code>eks:ListPodIdentityAssociations</code>

サービスプレフィックス	アクション
	eks:ListUpdates eks:RegisterCluster eks:UpdateAccessEntry eks:UpdateAddon eks:UpdateClusterConfig eks:UpdateClusterVersion eks:UpdateEksAnywhereSubscription eks:UpdateNodegroupConfig eks:UpdateNodegroupVersion eks:UpdatePodIdentityAssociation
elastic-inference	elastic-inference:DescribeAcceleratorOfferings elastic-inference:DescribeAccelerators elastic-inference:DescribeAcceleratorTypes

サービスプレフィックス	アクション
elasticache	elasticache:AuthorizeCacheSecurityGroupIngress
	elasticache:BatchApplyUpdateAction
	elasticache:BatchStopUpdateAction
	elasticache:CompleteMigration
	elasticache:CopyServerlessCacheSnapshot
	elasticache:CopySnapshot
	elasticache:CreateCacheCluster
	elasticache:CreateCacheParameterGroup
	elasticache:CreateCacheSecurityGroup
	elasticache:CreateCacheSubnetGroup
	elasticache:CreateGlobalReplicationGroup
	elasticache:CreateReplicationGroup
	elasticache:CreateServerlessCache
	elasticache:CreateServerlessCacheSnapshot
	elasticache:CreateSnapshot
	elasticache:CreateUser
	elasticache:CreateUserGroup
	elasticache:DecreaseNodeGroupsInGlobalReplicationGroup
	elasticache:DecreaseReplicaCount
	elasticache>DeleteCacheCluster
	elasticache>DeleteCacheParameterGroup

サービスプレフィックス	アクション
	<code>elasticache:DeleteCacheSecurityGroup</code>
	<code>elasticache:DeleteCacheSubnetGroup</code>
	<code>elasticache:DeleteGlobalReplicationGroup</code>
	<code>elasticache:DeleteReplicationGroup</code>
	<code>elasticache:DeleteServerlessCache</code>
	<code>elasticache:DeleteServerlessCacheSnapshot</code>
	<code>elasticache:DeleteSnapshot</code>
	<code>elasticache:DeleteUser</code>
	<code>elasticache:DeleteUserGroup</code>
	<code>elasticache:DescribeCacheClusters</code>
	<code>elasticache:DescribeCacheEngineVersions</code>
	<code>elasticache:DescribeCacheParameterGroups</code>
	<code>elasticache:DescribeCacheParameters</code>
	<code>elasticache:DescribeCacheSecurityGroups</code>
	<code>elasticache:DescribeCacheSubnetGroups</code>
	<code>elasticache:DescribeEngineDefaultParameters</code>
	<code>elasticache:DescribeEvents</code>
	<code>elasticache:DescribeGlobalReplicationGroups</code>
	<code>elasticache:DescribeReplicationGroups</code>
	<code>elasticache:DescribeReservedCacheNodes</code>
	<code>elasticache:DescribeReservedCacheNodesOfferings</code>

サービスプレフィックス	アクション
	elasticache:DescribeServerlessCaches
	elasticache:DescribeServerlessCacheSnapshots
	elasticache:DescribeServiceUpdates
	elasticache:DescribeSnapshots
	elasticache:DescribeUpdateActions
	elasticache:DescribeUserGroups
	elasticache:DescribeUsers
	elasticache:DisassociateGlobalReplicationGroup
	elasticache:ExportServerlessCacheSnapshot
	elasticache:FailoverGlobalReplicationGroup
	elasticache:IncreaseNodeGroupsInGlobalReplicationGroup
	elasticache:IncreaseReplicaCount
	elasticache:ListAllowedNodeTypeModifications
	elasticache:ModifyCacheCluster
	elasticache:ModifyCacheParameterGroup
	elasticache:ModifyCacheSubnetGroup
	elasticache:ModifyGlobalReplicationGroup
	elasticache:ModifyReplicationGroup
	elasticache:ModifyReplicationGroupShardConfiguration
	elasticache:ModifyServerlessCache
	elasticache:ModifyUser



サービスプレフィックス	アクション
	<code>elasticache:ModifyUserGroup</code>
	<code>elasticache:PurchaseReservedCacheNodesOffering</code>
	<code>elasticache:RebalanceSlotsInGlobalReplicationGroup</code>
	<code>elasticache:RebootCacheCluster</code>
	<code>elasticache:ResetCacheParameterGroup</code>
	<code>elasticache:RevokeCacheSecurityGroupIngress</code>
	<code>elasticache:StartMigration</code>
	<code>elasticache:TestFailover</code>
	<code>elasticache:TestMigration</code>

サービスプレフィックス	アクション
elasticbeanstalk	elasticbeanstalk:AbortEnvironmentUpdate
	elasticbeanstalk:ApplyEnvironmentManagedAction
	elasticbeanstalk:AssociateEnvironmentOperationsRole
	elasticbeanstalk:CheckDNSAvailability
	elasticbeanstalk:ComposeEnvironments
	elasticbeanstalk:CreateApplication
	elasticbeanstalk:CreateApplicationVersion
	elasticbeanstalk:CreateConfigurationTemplate
	elasticbeanstalk:CreateEnvironment
	elasticbeanstalk:CreatePlatformVersion
	elasticbeanstalk:CreateStorageLocation
	elasticbeanstalk>DeleteApplication
	elasticbeanstalk>DeleteApplicationVersion
	elasticbeanstalk>DeleteConfigurationTemplate
	elasticbeanstalk>DeleteEnvironmentConfiguration
	elasticbeanstalk>DeletePlatformVersion
	elasticbeanstalk:DescribeAccountAttributes
	elasticbeanstalk:DescribeApplications
	elasticbeanstalk:DescribeApplicationVersions
	elasticbeanstalk:DescribeConfigurationOptions
	elasticbeanstalk:DescribeConfigurationSettings

サービスプレフィックス	アクション
	elasticbeanstalk:DescribeEnvironmentHealth
	elasticbeanstalk:DescribeEnvironmentManagedActionHistory
	elasticbeanstalk:DescribeEnvironmentManagedActions
	elasticbeanstalk:DescribeEnvironmentResources
	elasticbeanstalk:DescribeEnvironments
	elasticbeanstalk:DescribeEvents
	elasticbeanstalk:DescribeInstancesHealth
	elasticbeanstalk:DescribePlatformVersion
	elasticbeanstalk:DisassociateEnvironmentOperationsRole
	elasticbeanstalk:ListAvailableSolutionStacks
	elasticbeanstalk:ListPlatformBranches
	elasticbeanstalk:ListPlatformVersions
	elasticbeanstalk:RebuildEnvironment
	elasticbeanstalk:RequestEnvironmentInfo
	elasticbeanstalk:RestartAppServer
	elasticbeanstalk:RetrieveEnvironmentInfo
	elasticbeanstalk:SwapEnvironmentCNAMEs
	elasticbeanstalk:TerminateEnvironment
	elasticbeanstalk:UpdateApplication
	elasticbeanstalk:UpdateApplicationResourceLifecycle
	elasticbeanstalk:UpdateApplicationVersion

サービスプレフィックス	アクション
	elasticbeanstalk:UpdateConfigurationTemplate
	elasticbeanstalk:UpdateEnvironment
	elasticbeanstalk:ValidateConfigurationSettings

サービスプレフィックス	アクション
elasticfilesystem	elasticfilesystem:CreateAccessPoint
	elasticfilesystem:CreateFileSystem
	elasticfilesystem:CreateMountTarget
	elasticfilesystem:CreateReplicationConfiguration
	elasticfilesystem>DeleteAccessPoint
	elasticfilesystem>DeleteFileSystem
	elasticfilesystem>DeleteFileSystemPolicy
	elasticfilesystem>DeleteMountTarget
	elasticfilesystem>DeleteReplicationConfiguration
	elasticfilesystem:DescribeAccessPoints
	elasticfilesystem:DescribeAccountPreferences
	elasticfilesystem:DescribeBackupPolicy
	elasticfilesystem:DescribeFileSystemPolicy
	elasticfilesystem:DescribeFileSystems
	elasticfilesystem:DescribeLifecycleConfiguration
	elasticfilesystem:DescribeMountTargets
	elasticfilesystem:DescribeMountTargetSecurityGroups
	elasticfilesystem:DescribeReplicationConfigurations
	elasticfilesystem:ModifyMountTargetSecurityGroups
	elasticfilesystem:PutAccountPreferences
	elasticfilesystem:PutBackupPolicy

サービスプレフィックス	アクション
	elasticfilesystem:PutFileSystemPolicy
	elasticfilesystem:PutLifecycleConfiguration
	elasticfilesystem:UpdateFileSystem
	elasticfilesystem:UpdateFileSystemProtection

サービスプレフィックス	アクション
elasticloadbalancing	elasticloadbalancing:AddListenerCertificates
	elasticloadbalancing:AddTrustStoreRevocations
	elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
	elasticloadbalancing:AttachLoadBalancerToSubnets
	elasticloadbalancing:ConfigureHealthCheck
	elasticloadbalancing>CreateAppCookieStickinessPolicy
	elasticloadbalancing>CreateLBCookieStickinessPolicy
	elasticloadbalancing>CreateListener
	elasticloadbalancing>CreateLoadBalancer
	elasticloadbalancing>CreateLoadBalancerListeners
	elasticloadbalancing>CreateLoadBalancerPolicy
	elasticloadbalancing>CreateRule
	elasticloadbalancing>CreateTargetGroup
	elasticloadbalancing>CreateTrustStore
	elasticloadbalancing>DeleteListener
	elasticloadbalancing>DeleteLoadBalancer
	elasticloadbalancing>DeleteLoadBalancerListeners
	elasticloadbalancing>DeleteLoadBalancerPolicy
	elasticloadbalancing>DeleteRule
	elasticloadbalancing>DeleteTargetGroup
	elasticloadbalancing>DeleteTrustStore

サービスプレフィックス	アクション
	elasticloadbalancing:DeregisterInstancesFromLoadBalancer elasticloadbalancing:DeregisterTargets elasticloadbalancing:DescribeAccountLimits elasticloadbalancing:DescribeInstanceHealth elasticloadbalancing:DescribeListenerCertificates elasticloadbalancing:DescribeListeners elasticloadbalancing:DescribeLoadBalancerAttributes elasticloadbalancing:DescribeLoadBalancerPolicies elasticloadbalancing:DescribeLoadBalancerPolicyTypes elasticloadbalancing:DescribeLoadBalancers elasticloadbalancing:DescribeRules elasticloadbalancing:DescribeSSLPolicies elasticloadbalancing:DescribeTargetGroupAttributes elasticloadbalancing:DescribeTargetGroups elasticloadbalancing:DescribeTargetHealth elasticloadbalancing:DescribeTrustStoreAssociations elasticloadbalancing:DescribeTrustStoreRevocations elasticloadbalancing:DescribeTrustStores elasticloadbalancing:DetachLoadBalancerFromSubnets elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer



サービスプレフィックス	アクション
	elasticloadbalancing:GetTrustStoreCaCertificatesBundle
	elasticloadbalancing:GetTrustStoreRevocationContent
	elasticloadbalancing:ModifyListener
	elasticloadbalancing:ModifyLoadBalancerAttributes
	elasticloadbalancing:ModifyRule
	elasticloadbalancing:ModifyTargetGroup
	elasticloadbalancing:ModifyTargetGroupAttributes
	elasticloadbalancing:ModifyTrustStore
	elasticloadbalancing:RegisterInstancesWithLoadBalancer
	elasticloadbalancing:RegisterTargets
	elasticloadbalancing:RemoveListenerCertificates
	elasticloadbalancing:RemoveTrustStoreRevocations
	elasticloadbalancing:SetIpAddressType
	elasticloadbalancing:SetLoadBalancerListenerSSLCertificate
	elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer
	elasticloadbalancing:SetLoadBalancerPoliciesOfListener
	elasticloadbalancing:SetRulePriorities
	elasticloadbalancing:SetSecurityGroups
	elasticloadbalancing:SetSubnets

サービスプレフィックス	アクション
elastictranscoder	elastictranscoder:CancelJob
	elastictranscoder:CreateJob
	elastictranscoder:CreatePipeline
	elastictranscoder:CreatePreset
	elastictranscoder>DeletePipeline
	elastictranscoder>DeletePreset
	elastictranscoder:ListJobsByPipeline
	elastictranscoder:ListJobsByStatus
	elastictranscoder:ListPipelines
	elastictranscoder:ListPresets
	elastictranscoder:ReadJob
	elastictranscoder:ReadPipeline
	elastictranscoder:ReadPreset
	elastictranscoder:TestRole
	elastictranscoder:UpdatePipeline
	elastictranscoder:UpdatePipelineNotifications
	elastictranscoder:UpdatePipelineStatus

サービスプレフィックス	アクション
emr-containers	emr-containers:CancelJobRun
	emr-containers:CreateJobTemplate
	emr-containers:CreateManagedEndpoint
	emr-containers:CreateSecurityConfiguration
	emr-containers:CreateVirtualCluster
	emr-containers>DeleteJobTemplate
	emr-containers>DeleteManagedEndpoint
	emr-containers>DeleteVirtualCluster
	emr-containers:DescribeJobRun
	emr-containers:DescribeJobTemplate
	emr-containers:DescribeManagedEndpoint
	emr-containers:DescribeSecurityConfiguration
	emr-containers:DescribeVirtualCluster
	emr-containers:GetManagedEndpointSessionCredentials
	emr-containers:ListJobRuns
	emr-containers:ListJobTemplates
	emr-containers:ListManagedEndpoints
	emr-containers:ListSecurityConfigurations
	emr-containers:ListVirtualClusters
	emr-containers:StartJobRun

サービスプレフィックス	アクション
emr-serverless	emr-serverless:CancelJobRun
	emr-serverless:CreateApplication
	emr-serverless>DeleteApplication
	emr-serverless:GetApplication
	emr-serverless:GetDashboardForJobRun
	emr-serverless:GetJobRun
	emr-serverless:ListApplications
	emr-serverless:ListJobRuns
	emr-serverless:StartApplication
	emr-serverless:StartJobRun
	emr-serverless:StopApplication
	emr-serverless:UpdateApplication

サービスプレフィックス	アクション
es	es:AcceptInboundConnection
	es:AcceptInboundCrossClusterSearchConnection
	es:AssociatePackage
	es:AuthorizeVpcEndpointAccess
	es:CancelElasticsearchServiceSoftwareUpdate
	es:CancelServiceSoftwareUpdate
	es:CreateDomain
	es:CreateElasticsearchDomain
	es:CreateOutboundConnection
	es:CreateOutboundCrossClusterSearchConnection
	es:CreatePackage
	es:CreateVpcEndpoint
	es>DeleteDomain
	es>DeleteElasticsearchDomain
	es>DeleteElasticsearchServiceRole
	es>DeleteInboundConnection
	es>DeleteInboundCrossClusterSearchConnection
	es>DeleteOutboundConnection
	es>DeleteOutboundCrossClusterSearchConnection
	es>DeletePackage
	es>DeleteVpcEndpoint

サービスプレフィックス	アクション
	es:DescribeDomain
	es:DescribeDomainAutoTunes
	es:DescribeDomainChangeProgress
	es:DescribeDomainConfig
	es:DescribeDomainHealth
	es:DescribeDomainNodes
	es:DescribeDomains
	es:DescribeDryRunProgress
	es:DescribeElasticsearchDomain
	es:DescribeElasticsearchDomainConfig
	es:DescribeElasticsearchDomains
	es:DescribeElasticsearchInstanceTypeLimits
	es:DescribeInboundConnections
	es:DescribeInboundCrossClusterSearchConnections
	es:DescribeInstanceTypeLimits
	es:DescribeOutboundConnections
	es:DescribeOutboundCrossClusterSearchConnections
	es:DescribePackages
	es:DescribeReservedElasticsearchInstanceOfferings
	es:DescribeReservedElasticsearchInstances
	es:DescribeReservedInstanceOfferings

サービスプレフィックス	アクション
	es:DescribeReservedInstances
	es:DescribeVpcEndpoints
	es:DissociatePackage
	es:GetCompatibleElasticsearchVersions
	es:GetCompatibleVersions
	es:GetDataSource
	es:GetDomainMaintenanceStatus
	es:GetPackageVersionHistory
	es:GetUpgradeHistory
	es:GetUpgradeStatus
	es:ListDataSources
	es:ListDomainNames
	es:ListDomainsForPackage
	es:ListElasticsearchInstanceTypes
	es:ListElasticsearchVersions
	es:ListInstanceTypeDetails
	es:ListPackagesForDomain
	es:ListScheduledActions
	es:ListVersions
	es:ListVpcEndpointAccess
	es:ListVpcEndpoints

サービスプレフィックス	アクション
	es:ListVpcEndpointsForDomain
	es:PurchaseReservedElasticsearchInstanceOffering
	es:PurchaseReservedInstanceOffering
	es:RejectInboundConnection
	es:RejectInboundCrossClusterSearchConnection
	es:RevokeVpcEndpointAccess
	es:StartDomainMaintenance
	es:StartElasticsearchServiceSoftwareUpdate
	es:StartServiceSoftwareUpdate
	es:UpdateDataSource
	es:UpdateDomainConfig
	es:UpdateElasticsearchDomainConfig
	es:UpdatePackage
	es:UpdateScheduledAction
	es:UpdateVpcEndpoint
	es:UpgradeDomain
	es:UpgradeElasticsearchDomain



サービスプレフィックス	アクション
events	events:ActivateEventSource
	events:CancelReplay
	events:CreateApiDestination
	events:CreateArchive
	events:CreateConnection
	events:CreateEndpoint
	events:CreateEventBus
	events:CreatePartnerEventSource
	events:DeactivateEventSource
	events:DeauthorizeConnection
	events>DeleteApiDestination
	events>DeleteArchive
	events>DeleteConnection
	events>DeleteEndpoint
	events>DeleteEventBus
	events>DeletePartnerEventSource
	events>DeleteRule
	events:DescribeApiDestination
	events:DescribeArchive
	events:DescribeConnection
	events:DescribeEndpoint

サービスプレフィックス	アクション
	events:DescribeEventBus
	events:DescribeEventSource
	events:DescribePartnerEventSource
	events:DescribeReplay
	events:DescribeRule
	events:DisableRule
	events:EnableRule
	events:ListApiDestinations
	events:ListArchives
	events:ListConnections
	events:ListEndpoints
	events:ListEventBuses
	events:ListEventSources
	events:ListPartnerEventSourceAccounts
	events:ListPartnerEventSources
	events:ListReplays
	events:ListRuleNamesByTarget
	events:ListRules
	events:ListTargetsByRule
	events:PutPermission
	events:PutRule

サービスプレフィックス	アクション
	events:PutTargets
	events:RemovePermission
	events:RemoveTargets
	events:StartReplay
	events:TestEventPattern
	events:UpdateApiDestination
	events:UpdateArchive
	events:UpdateConnection
	events:UpdateEndpoint

サービスプレフィックス	アクション
evidently	evidently:CreateExperiment
	evidently:CreateFeature
	evidently:CreateLaunch
	evidently:CreateProject
	evidently:CreateSegment
	evidently>DeleteExperiment
	evidently>DeleteFeature
	evidently>DeleteLaunch
	evidently>DeleteProject
	evidently>DeleteSegment
	evidently:GetExperiment
	evidently:GetExperimentResults
	evidently:GetFeature
	evidently:GetLaunch
	evidently:GetProject
	evidently:GetSegment
	evidently:ListExperiments
	evidently:ListFeatures
	evidently:ListLaunches
	evidently:ListProjects
	evidently:ListSegmentReferences

サービスプレフィックス	アクション
	<p>evidently:ListSegments</p> <p>evidently:StartExperiment</p> <p>evidently:StartLaunch</p> <p>evidently:StopExperiment</p> <p>evidently:StopLaunch</p> <p>evidently:TestSegmentPattern</p> <p>evidently:UpdateExperiment</p> <p>evidently:UpdateFeature</p> <p>evidently:UpdateLaunch</p> <p>evidently:UpdateProject</p> <p>evidently:UpdateProjectDataDelivery</p>

サービスプレフィックス	アクション
finspace	finspace:CreateEnvironment
	finspace:CreateKxChangeset
	finspace:CreateKxCluster
	finspace:CreateKxDatabase
	finspace:CreateKxDataview
	finspace:CreateKxEnvironment
	finspace:CreateKxScalingGroup
	finspace:CreateKxUser
	finspace:CreateKxVolume
	finspace:CreateUser
	finspace>DeleteEnvironment
	finspace>DeleteKxCluster
	finspace>DeleteKxClusterNode
	finspace>DeleteKxDatabase
	finspace>DeleteKxDataview
	finspace>DeleteKxEnvironment
	finspace>DeleteKxScalingGroup
	finspace>DeleteKxUser
	finspace>DeleteKxVolume
	finspace:GetEnvironment
	finspace:GetKxChangeset

サービスプレフィックス	アクション
	<code>finspace:GetKxCluster</code>
	<code>finspace:GetKxConnectionString</code>
	<code>finspace:GetKxDatabase</code>
	<code>finspace:GetKxDataview</code>
	<code>finspace:GetKxEnvironment</code>
	<code>finspace:GetKxScalingGroup</code>
	<code>finspace:GetKxUser</code>
	<code>finspace:GetKxVolume</code>
	<code>finspace:GetLoadSampleDataSetGroupIntoEnvironmentStatus</code>
	<code>finspace:GetUser</code>
	<code>finspace:ListEnvironments</code>
	<code>finspace:ListKxChangesets</code>
	<code>finspace:ListKxClusterNodes</code>
	<code>finspace:ListKxClusters</code>
	<code>finspace:ListKxDatabases</code>
	<code>finspace:ListKxDataviews</code>
	<code>finspace:ListKxEnvironments</code>
	<code>finspace:ListKxScalingGroups</code>
	<code>finspace:ListKxUsers</code>
	<code>finspace:ListKxVolumes</code>
	<code>finspace:ListUsers</code>

サービスプレフィックス	アクション
	<p>finospace:LoadSampleDataSetGroupIntoEnvironment</p> <p>finospace:ResetUserPassword</p> <p>finospace:UpdateEnvironment</p> <p>finospace:UpdateKxClusterCodeConfiguration</p> <p>finospace:UpdateKxClusterDatabases</p> <p>finospace:UpdateKxDatabase</p> <p>finospace:UpdateKxDataview</p> <p>finospace:UpdateKxEnvironment</p> <p>finospace:UpdateKxEnvironmentNetwork</p> <p>finospace:UpdateKxUser</p> <p>finospace:UpdateKxVolume</p> <p>finospace:UpdateUser</p>
firehose	<p>firehose:CreateDeliveryStream</p> <p>firehose&gt;DeleteDeliveryStream</p> <p>firehose:DescribeDeliveryStream</p> <p>firehose:ListDeliveryStreams</p> <p>firehose:StartDeliveryStreamEncryption</p> <p>firehose:StopDeliveryStreamEncryption</p> <p>firehose:UpdateDestination</p>



サービスプレフィックス	アクション
fis	fis:CreateExperimentTemplate
	fis:CreateTargetAccountConfiguration
	fis>DeleteExperimentTemplate
	fis>DeleteTargetAccountConfiguration
	fis:GetAction
	fis:GetExperiment
	fis:GetExperimentTargetAccountConfiguration
	fis:GetExperimentTemplate
	fis:GetTargetAccountConfiguration
	fis:GetTargetResourceType
	fis:ListActions
	fis:ListExperimentResolvedTargets
	fis:ListExperiments
	fis:ListExperimentTargetAccountConfigurations
	fis:ListExperimentTemplates
	fis:ListTargetAccountConfigurations
	fis:ListTargetResourceTypes
	fis:StartExperiment
	fis:StopExperiment
	fis:UpdateExperimentTemplate
fis:UpdateTargetAccountConfiguration	

サービスプレフィックス	アクション
fms	fms:AssociateAdminAccount
	fms:AssociateThirdPartyFirewall
	fms:BatchAssociateResource
	fms:BatchDisassociateResource
	fms>DeleteAppsList
	fms>DeleteNotificationChannel
	fms>DeletePolicy
	fms>DeleteProtocolsList
	fms>DeleteResourceSet
	fms:DisassociateAdminAccount
	fms:DisassociateThirdPartyFirewall
	fms:GetAdminAccount
	fms:GetAdminScope
	fms:GetAppsList
	fms:GetComplianceDetail
	fms:GetNotificationChannel
	fms:GetPolicy
	fms:GetProtectionStatus
	fms:GetProtocolsList
	fms:GetResourceSet
	fms:GetThirdPartyFirewallAssociationStatus

サービスプレフィックス	アクション
	fms:GetViolationDetails
	fms:ListAdminAccountsForOrganization
	fms:ListAdminsManagingAccount
	fms:ListAppsLists
	fms:ListComplianceStatus
	fms:ListDiscoveredResources
	fms:ListMemberAccounts
	fms:ListPolicies
	fms:ListProtocolsLists
	fms:ListResourceSetResources
	fms:ListResourceSets
	fms:ListThirdPartyFirewallFirewallPolicies
	fms:PutAdminAccount
	fms:PutAppsList
	fms:PutNotificationChannel
	fms:PutPolicy
	fms:PutProtocolsList
	fms:PutResourceSet

サービスプレフィックス	アクション
frauddetector	frauddetector:BatchCreateVariable
	frauddetector:BatchGetVariable
	frauddetector:CancelBatchImportJob
	frauddetector:CancelBatchPredictionJob
	frauddetector:CreateBatchImportJob
	frauddetector:CreateBatchPredictionJob
	frauddetector:CreateDetectorVersion
	frauddetector:CreateList
	frauddetector:CreateModel
	frauddetector:CreateModelVersion
	frauddetector:CreateRule
	frauddetector:CreateVariable
	frauddetector>DeleteBatchImportJob
	frauddetector>DeleteBatchPredictionJob
	frauddetector>DeleteDetector
	frauddetector>DeleteDetectorVersion
	frauddetector>DeleteEntityType
	frauddetector>DeleteEvent
	frauddetector>DeleteEventsByEventType
	frauddetector>DeleteEventType
	frauddetector>DeleteExternalModel

サービスプレフィックス	アクション
	frauddetector:DeleteLabel
	frauddetector:DeleteList
	frauddetector:DeleteModel
	frauddetector:DeleteModelVersion
	frauddetector:DeleteOutcome
	frauddetector:DeleteRule
	frauddetector:DeleteVariable
	frauddetector:DescribeDetector
	frauddetector:DescribeModelVersions
	frauddetector:GetBatchImportJobs
	frauddetector:GetBatchPredictionJobs
	frauddetector:GetDeleteEventsByEventTypeStatus
	frauddetector:GetDetectors
	frauddetector:GetDetectorVersion
	frauddetector:GetEntityTypeTypes
	frauddetector:GetEvent
	frauddetector:GetEventPrediction
	frauddetector:GetEventPredictionMetadata
	frauddetector:GetEventTypes
	frauddetector:GetExternalModels
	frauddetector:GetKMSEncryptionKey

サービスプレフィックス	アクション
	frauddetector:GetLabels
	frauddetector:GetListElements
	frauddetector:GetListsMetadata
	frauddetector:GetModels
	frauddetector:GetModelVersion
	frauddetector:GetOutcomes
	frauddetector:GetRules
	frauddetector:GetVariables
	frauddetector:ListEventPredictions
	frauddetector:PutDetector
	frauddetector:PutEntityType
	frauddetector:PutEventType
	frauddetector:PutExternalModel
	frauddetector:PutKMSEncryptionKey
	frauddetector:PutLabel
	frauddetector:PutOutcome
	frauddetector:SendEvent
	frauddetector:UpdateDetectorVersion
	frauddetector:UpdateDetectorVersionMetadata
	frauddetector:UpdateDetectorVersionStatus
	frauddetector:UpdateEventLabel

サービスプレフィックス	アクション
	frauddetector:UpdateList
	frauddetector:UpdateModel
	frauddetector:UpdateModelVersion
	frauddetector:UpdateModelVersionStatus
	frauddetector:UpdateRuleMetadata
	frauddetector:UpdateRuleVersion
	frauddetector:UpdateVariable

サービスプレフィックス	アクション
fsx	fsx:AssociateFileSystemAliases
	fsx:CancelDataRepositoryTask
	fsx:CopyBackup
	fsx:CreateDataRepositoryTask
	fsx:CreateFileCache
	fsx:CreateFileSystem
	fsx:CreateFileSystemFromBackup
	fsx:CreateSnapshot
	fsx:CreateStorageVirtualMachine
	fsx:CreateVolume
	fsx:CreateVolumeFromBackup
	fsx>DeleteBackup
	fsx>DeleteFileCache
	fsx>DeleteFileSystem
	fsx>DeleteSnapshot
	fsx>DeleteStorageVirtualMachine
	fsx>DeleteVolume
	fsx:DescribeBackups
	fsx:DescribeDataRepositoryAssociations
	fsx:DescribeDataRepositoryTasks
	fsx:DescribeFileCaches



サービスプレフィックス	アクション
	fsx:DescribeFileSystemAliases
	fsx:DescribeFileSystems
	fsx:DescribeSharedVpcConfiguration
	fsx:DescribeSnapshots
	fsx:DescribeStorageVirtualMachines
	fsx:DescribeVolumes
	fsx:DisassociateFileSystemAliases
	fsx:ReleaseFileSystemNfsV3Locks
	fsx:RestoreVolumeFromSnapshot
	fsx:StartMisconfiguredStateRecovery
	fsx:UpdateDataRepositoryAssociation
	fsx:UpdateFileCache
	fsx:UpdateFileSystem
	fsx:UpdateSharedVpcConfiguration
	fsx:UpdateSnapshot
	fsx:UpdateStorageVirtualMachine
	fsx:UpdateVolume

サービスプレフィックス	アクション
gamelift	gamelift:AcceptMatch
	gamelift:ClaimGameServer
	gamelift:CreateAlias
	gamelift:CreateBuild
	gamelift:CreateContainerGroupDefinition
	gamelift:CreateFleet
	gamelift:CreateFleetLocations
	gamelift:CreateGameServerGroup
	gamelift:CreateGameSession
	gamelift:CreateGameSessionQueue
	gamelift:CreateLocation
	gamelift:CreateMatchmakingConfiguration
	gamelift:CreateMatchmakingRuleSet
	gamelift:CreatePlayerSession
	gamelift:CreatePlayerSessions
	gamelift:CreateScript
	gamelift:CreateVpcPeeringAuthorization
	gamelift:CreateVpcPeeringConnection
	gamelift>DeleteAlias
	gamelift>DeleteBuild
	gamelift>DeleteContainerGroupDefinition

サービスプレフィックス	アクション
	gamelift:DeleteFleet
	gamelift:DeleteFleetLocations
	gamelift:DeleteGameServerGroup
	gamelift:DeleteGameSessionQueue
	gamelift:DeleteLocation
	gamelift:DeleteMatchmakingConfiguration
	gamelift:DeleteMatchmakingRuleSet
	gamelift:DeleteScalingPolicy
	gamelift:DeleteScript
	gamelift:DeleteVpcPeeringAuthorization
	gamelift:DeleteVpcPeeringConnection
	gamelift:DeregisterCompute
	gamelift:DeregisterGameServer
	gamelift:DescribeAlias
	gamelift:DescribeBuild
	gamelift:DescribeCompute
	gamelift:DescribeContainerGroupDefinition
	gamelift:DescribeEC2InstanceLimits
	gamelift:DescribeFleetAttributes
	gamelift:DescribeFleetCapacity
	gamelift:DescribeFleetEvents

サービスプレフィックス	アクション
	gamelift:DescribeFleetLocationAttributes
	gamelift:DescribeFleetLocationCapacity
	gamelift:DescribeFleetLocationUtilization
	gamelift:DescribeFleetPortSettings
	gamelift:DescribeFleetUtilization
	gamelift:DescribeGameServer
	gamelift:DescribeGameServerGroup
	gamelift:DescribeGameServerInstances
	gamelift:DescribeGameSessionDetails
	gamelift:DescribeGameSessionPlacement
	gamelift:DescribeGameSessionQueues
	gamelift:DescribeGameSessions
	gamelift:DescribeInstances
	gamelift:DescribeMatchmaking
	gamelift:DescribeMatchmakingConfigurations
	gamelift:DescribeMatchmakingRuleSets
	gamelift:DescribePlayerSessions
	gamelift:DescribeRuntimeConfiguration
	gamelift:DescribeScalingPolicies
	gamelift:DescribeScript
	gamelift:DescribeVpcPeeringAuthorizations

サービスプレフィックス	アクション
	gamelift:DescribeVpcPeeringConnections
	gamelift:GetComputeAccess
	gamelift:GetComputeAuthToken
	gamelift:GetGameSessionLogUrl
	gamelift:GetInstanceAccess
	gamelift:ListAliases
	gamelift:ListBuilds
	gamelift:ListCompute
	gamelift:ListContainerGroupDefinitions
	gamelift:ListFleets
	gamelift:ListGameServerGroups
	gamelift:ListGameServers
	gamelift:ListLocations
	gamelift:ListScripts
	gamelift:PutScalingPolicy
	gamelift:RegisterCompute
	gamelift:RegisterGameServer
	gamelift:RequestUploadCredentials
	gamelift:ResolveAlias
	gamelift:ResumeGameServerGroup
	gamelift:SearchGameSessions

サービスプレフィックス	アクション
	gamelift:StartFleetActions
	gamelift:StartGameSessionPlacement
	gamelift:StartMatchBackfill
	gamelift:StartMatchmaking
	gamelift:StopFleetActions
	gamelift:StopGameSessionPlacement
	gamelift:StopMatchmaking
	gamelift:SuspendGameServerGroup
	gamelift:UpdateAlias
	gamelift:UpdateBuild
	gamelift:UpdateFleetAttributes
	gamelift:UpdateFleetCapacity
	gamelift:UpdateFleetPortSettings
	gamelift:UpdateGameServer
	gamelift:UpdateGameServerGroup
	gamelift:UpdateGameSession
	gamelift:UpdateGameSessionQueue
	gamelift:UpdateMatchmakingConfiguration
	gamelift:UpdateRuntimeConfiguration
	gamelift:UpdateScript
	gamelift:ValidateMatchmakingRuleSet

サービスプレフィックス	アクション
geo	geo:AssociateTrackerConsumer
	geo:BatchDeleteDevicePositionHistory
	geo:BatchDeleteGeofence
	geo:BatchEvaluateGeofences
	geo:BatchGetDevicePosition
	geo:BatchPutGeofence
	geo:BatchUpdateDevicePosition
	geo:CalculateRoute
	geo:CalculateRouteMatrix
	geo>CreateGeofenceCollection
	geo>CreateMap
	geo>CreatePlaceIndex
	geo>CreateRouteCalculator
	geo>CreateTracker
	geo>DeleteGeofenceCollection
	geo>DeleteKey
	geo>DeleteMap
	geo>DeletePlaceIndex
	geo>DeleteRouteCalculator
	geo>DeleteTracker
	geo:DescribeGeofenceCollection

サービスプレフィックス	アクション
	geo:DescribeKey
	geo:DescribeMap
	geo:DescribePlaceIndex
	geo:DescribeRouteCalculator
	geo:DescribeTracker
	geo:DisassociateTrackerConsumer
	geo:GetDevicePosition
	geo:GetDevicePositionHistory
	geo:GetGeofence
	geo:GetMapGlyphs
	geo:GetMapSprites
	geo:GetMapStyleDescriptor
	geo:GetMapTile
	geo:GetPlace
	geo:ListDevicePositions
	geo:ListGeofenceCollections
	geo:ListGeofences
	geo:ListKeys
	geo:ListMaps
	geo:ListPlaceIndexes
	geo:ListRouteCalculators



サービスプレフィックス	アクション
	geo:ListTrackerConsumers
	geo:ListTrackers
	geo:PutGeofence
	geo:SearchPlaceIndexForPosition
	geo:SearchPlaceIndexForSuggestions
	geo:SearchPlaceIndexForText
	geo:UpdateGeofenceCollection
	geo:UpdateKey
	geo:UpdateMap
	geo:UpdatePlaceIndex
	geo:UpdateRouteCalculator
	geo:UpdateTracker

サービスプレフィックス	アクション
glacier	glacier:AbortMultipartUpload
	glacier:AbortVaultLock
	glacier:CompleteMultipartUpload
	glacier:CompleteVaultLock
	glacier:CreateVault
	glacier>DeleteArchive
	glacier>DeleteVault
	glacier>DeleteVaultAccessPolicy
	glacier>DeleteVaultNotifications
	glacier:DescribeJob
	glacier:DescribeVault
	glacier:GetDataRetrievalPolicy
	glacier:GetJobOutput
	glacier:GetVaultAccessPolicy
	glacier:GetVaultLock
	glacier:GetVaultNotifications
	glacier:InitiateJob
	glacier:InitiateMultipartUpload
	glacier:InitiateVaultLock
	glacier:ListJobs
	glacier:ListMultipartUploads

サービスプレフィックス	アクション
	glacier:ListParts
	glacier:ListProvisionedCapacity
	glacier:ListVaults
	glacier:PurchaseProvisionedCapacity
	glacier:SetDataRetrievalPolicy
	glacier:SetVaultAccessPolicy
	glacier:SetVaultNotifications
	glacier:UploadArchive
	glacier:UploadMultipartPart

サービスプレフィックス	アクション
grafana	grafana:AssociateLicense
	grafana:CreateWorkspace
	grafana:CreateWorkspaceApiKey
	grafana>DeleteWorkspace
	grafana>DeleteWorkspaceApiKey
	grafana:DescribeWorkspace
	grafana:DescribeWorkspaceAuthentication
	grafana:DescribeWorkspaceConfiguration
	grafana:DisassociateLicense
	grafana:ListPermissions
	grafana:ListVersions
	grafana:ListWorkspaces
	grafana:UpdatePermissions
	grafana:UpdateWorkspace
	grafana:UpdateWorkspaceAuthentication
	grafana:UpdateWorkspaceConfiguration

サービスプレフィックス	アクション
greengrass	greengrass:AssociateRoleToGroup
	greengrass:AssociateServiceRoleToAccount
	greengrass:BatchAssociateClientDeviceWithCoreDevice
	greengrass:BatchDisassociateClientDeviceFromCoreDevice
	greengrass:CancelDeployment
	greengrass:CreateComponentVersion
	greengrass:CreateConnectorDefinition
	greengrass:CreateConnectorDefinitionVersion
	greengrass:CreateCoreDefinition
	greengrass:CreateCoreDefinitionVersion
	greengrass:CreateDeployment
	greengrass:CreateDeviceDefinition
	greengrass:CreateDeviceDefinitionVersion
	greengrass:CreateFunctionDefinition
	greengrass:CreateFunctionDefinitionVersion
	greengrass:CreateGroup
	greengrass:CreateGroupCertificateAuthority
	greengrass:CreateGroupVersion
	greengrass:CreateLoggerDefinition
	greengrass:CreateLoggerDefinitionVersion
	greengrass:CreateResourceDefinition

サービスプレフィックス	アクション
	greengrass:CreateResourceDefinitionVersion
	greengrass:CreateSoftwareUpdateJob
	greengrass:CreateSubscriptionDefinition
	greengrass:CreateSubscriptionDefinitionVersion
	greengrass>DeleteComponent
	greengrass>DeleteConnectorDefinition
	greengrass>DeleteCoreDefinition
	greengrass>DeleteCoreDevice
	greengrass>DeleteDeployment
	greengrass>DeleteDeviceDefinition
	greengrass>DeleteFunctionDefinition
	greengrass>DeleteGroup
	greengrass>DeleteLoggerDefinition
	greengrass>DeleteResourceDefinition
	greengrass>DeleteSubscriptionDefinition
	greengrass:DescribeComponent
	greengrass:DisassociateRoleFromGroup
	greengrass:DisassociateServiceRoleFromAccount
	greengrass:GetAssociatedRole
	greengrass:GetBulkDeploymentStatus
	greengrass:GetComponent

サービスプレフィックス	アクション
	greengrass:GetComponentVersionArtifact
	greengrass:GetConnectivityInfo
	greengrass:GetConnectorDefinition
	greengrass:GetConnectorDefinitionVersion
	greengrass:GetCoreDefinition
	greengrass:GetCoreDefinitionVersion
	greengrass:GetCoreDevice
	greengrass:GetDeployment
	greengrass:GetDeploymentStatus
	greengrass:GetDeviceDefinition
	greengrass:GetDeviceDefinitionVersion
	greengrass:GetFunctionDefinition
	greengrass:GetFunctionDefinitionVersion
	greengrass:GetGroup
	greengrass:GetGroupCertificateAuthority
	greengrass:GetGroupCertificateConfiguration
	greengrass:GetGroupVersion
	greengrass:GetLoggerDefinition
	greengrass:GetLoggerDefinitionVersion
	greengrass:GetResourceDefinition
	greengrass:GetResourceDefinitionVersion

サービスプレフィックス	アクション
	greengrass:GetServiceRoleForAccount
	greengrass:GetSubscriptionDefinition
	greengrass:GetSubscriptionDefinitionVersion
	greengrass:GetThingRuntimeConfiguration
	greengrass:ListBulkDeploymentDetailedReports
	greengrass:ListBulkDeployments
	greengrass:ListClientDevicesAssociatedWithCoreDevice
	greengrass:ListComponents
	greengrass:ListComponentVersions
	greengrass:ListConnectorDefinitions
	greengrass:ListConnectorDefinitionVersions
	greengrass:ListCoreDefinitions
	greengrass:ListCoreDefinitionVersions
	greengrass:ListCoreDevices
	greengrass:ListDeployments
	greengrass:ListDeviceDefinitions
	greengrass:ListDeviceDefinitionVersions
	greengrass:ListEffectiveDeployments
	greengrass:ListFunctionDefinitions
	greengrass:ListFunctionDefinitionVersions
	greengrass:ListGroupCertificateAuthorities



サービスプレフィックス	アクション
	greengrass:ListGroup
	greengrass:ListGroupVersions
	greengrass:ListInstalledComponents
	greengrass:ListLoggerDefinitions
	greengrass:ListLoggerDefinitionVersions
	greengrass:ListResourceDefinitions
	greengrass:ListResourceDefinitionVersions
	greengrass:ListSubscriptionDefinitions
	greengrass:ListSubscriptionDefinitionVersions
	greengrass:ResetDeployments
	greengrass:StartBulkDeployment
	greengrass:StopBulkDeployment
	greengrass:UpdateConnectivityInfo
	greengrass:UpdateConnectorDefinition
	greengrass:UpdateCoreDefinition
	greengrass:UpdateDeviceDefinition
	greengrass:UpdateFunctionDefinition
	greengrass:UpdateGroup
	greengrass:UpdateGroupCertificateConfiguration
	greengrass:UpdateLoggerDefinition
	greengrass:UpdateResourceDefinition

サービスプレフィックス	アクション
	greengrass:UpdateSubscriptionDefinition  greengrass:UpdateThingRuntimeConfiguration

サービスプレフィックス	アクション
groundstation	groundstation:CancelContact
	groundstation:CreateConfig
	groundstation:CreateDataflowEndpointGroup
	groundstation:CreateEphemeris
	groundstation:CreateMissionProfile
	groundstation>DeleteConfig
	groundstation>DeleteDataflowEndpointGroup
	groundstation>DeleteEphemeris
	groundstation>DeleteMissionProfile
	groundstation:DescribeContact
	groundstation:DescribeEphemeris
	groundstation:GetConfig
	groundstation:GetDataflowEndpointGroup
	groundstation:GetMinuteUsage
	groundstation:GetMissionProfile
	groundstation:GetSatellite
	groundstation:ListConfigs
	groundstation:ListContacts
	groundstation:ListDataflowEndpointGroups
	groundstation:ListEphemerides
	groundstation:ListGroundStations

サービスプレフィックス	アクション
	<p>groundstation:ListMissionProfiles</p> <p>groundstation:ListSatellites</p> <p>groundstation:RegisterAgent</p> <p>groundstation:ReserveContact</p> <p>groundstation:UpdateAgentStatus</p> <p>groundstation:UpdateConfig</p> <p>groundstation:UpdateEphemeris</p> <p>groundstation:UpdateMissionProfile</p>

サービスプレフィックス	アクション
guardduty	guardduty:AcceptAdministratorInvitation
	guardduty:AcceptInvitation
	guardduty:ArchiveFindings
	guardduty:CreateDetector
	guardduty:CreateFilter
	guardduty:CreateIPSet
	guardduty:CreateMembers
	guardduty:CreatePublishingDestination
	guardduty:CreateSampleFindings
	guardduty:CreateThreatIntelSet
	guardduty:DeclineInvitations
	guardduty>DeleteDetector
	guardduty>DeleteFilter
	guardduty>DeleteInvitations
	guardduty>DeleteIPSet
	guardduty>DeleteMembers
	guardduty>DeletePublishingDestination
	guardduty>DeleteThreatIntelSet
	guardduty:DescribeMalwareScans
	guardduty:DescribeOrganizationConfiguration
	guardduty:DescribePublishingDestination

サービスプレフィックス	アクション
	<code>guardduty:DisableOrganizationAdminAccount</code>
	<code>guardduty:DisassociateFromAdministratorAccount</code>
	<code>guardduty:DisassociateFromMasterAccount</code>
	<code>guardduty:DisassociateMembers</code>
	<code>guardduty:EnableOrganizationAdminAccount</code>
	<code>guardduty:GetAdministratorAccount</code>
	<code>guardduty:GetCoverageStatistics</code>
	<code>guardduty:GetDetector</code>
	<code>guardduty:GetFilter</code>
	<code>guardduty:GetFindings</code>
	<code>guardduty:GetFindingsStatistics</code>
	<code>guardduty:GetInvitationsCount</code>
	<code>guardduty:GetIPSet</code>
	<code>guardduty:GetMalwareScanSettings</code>
	<code>guardduty:GetMasterAccount</code>
	<code>guardduty:GetMemberDetectors</code>
	<code>guardduty:GetMembers</code>
	<code>guardduty:GetOrganizationStatistics</code>
	<code>guardduty:GetRemainingFreeTrialDays</code>
	<code>guardduty:GetThreatIntelSet</code>
	<code>guardduty:GetUsageStatistics</code>

サービスプレフィックス	アクション
	guardduty:InviteMembers
	guardduty:ListCoverage
	guardduty:ListDetectors
	guardduty:ListFilters
	guardduty:ListFindings
	guardduty:ListInvitations
	guardduty:ListIPSets
	guardduty:ListMembers
	guardduty:ListOrganizationAdminAccounts
	guardduty:ListPublishingDestinations
	guardduty:ListThreatIntelSets
	guardduty:SendSecurityTelemetry
	guardduty:StartMalwareScan
	guardduty:StartMonitoringMembers
	guardduty:StopMonitoringMembers
	guardduty:UnarchiveFindings
	guardduty:UpdateDetector
	guardduty:UpdateFilter
	guardduty:UpdateFindingsFeedback
	guardduty:UpdateIPSet
	guardduty:UpdateMalwareScanSettings

サービスプレフィックス	アクション
	guardduty:UpdateMemberDetectors guardduty:UpdateOrganizationConfiguration guardduty:UpdatePublishingDestination guardduty:UpdateThreatIntelSet
healthlake	healthlake:CreateFHIRDatastore healthlake:CreateResource healthlake>DeleteFHIRDatastore healthlake>DeleteResource healthlake:DescribeFHIRDatastore healthlake:DescribeFHIRExportJob healthlake:DescribeFHIRImportJob healthlake:GetCapabilities healthlake>ListFHIRDatastores healthlake>ListFHIRExportJobs healthlake>ListFHIRImportJobs healthlake:ReadResource healthlake:SearchWithGet healthlake:SearchWithPost healthlake:StartFHIRExportJob healthlake:StartFHIRImportJob healthlake:UpdateResource



サービスプレフィックス	アクション
honeycode	honeycode:BatchCreateTableRows honeycode:BatchDeleteTableRows honeycode:BatchUpdateTableRows honeycode:BatchUpsertTableRows honeycode:DescribeTableDataImportJob honeycode:GetScreenData honeycode:InvokeScreenAutomation honeycode>ListTableColumns honeycode>ListTableRows honeycode>ListTables honeycode:QueryTableRows honeycode:StartTableDataImportJob

サービスプレフィックス	アクション
iam	iam:AddClientIDToOpenIDConnectProvider
	iam:AddRoleToInstanceProfile
	iam:AddUserToGroup
	iam:AttachGroupPolicy
	iam:AttachRolePolicy
	iam:AttachUserPolicy
	iam:ChangePassword
	iam:CreateAccessKey
	iam:CreateAccountAlias
	iam:CreateGroup
	iam:CreateInstanceProfile
	iam:CreateLoginProfile
	iam:CreateOpenIDConnectProvider
	iam:CreatePolicy
	iam:CreatePolicyVersion
	iam:CreateRole
	iam:CreateSAMLProvider
	iam:CreateServiceLinkedRole
	iam:CreateServiceSpecificCredential
	iam:CreateUser
	iam:CreateVirtualMFADevice

サービスプレフィックス	アクション
	iam:DeactivateMFADevice
	iam>DeleteAccessKey
	iam>DeleteAccountAlias
	iam>DeleteAccountPasswordPolicy
	iam>DeleteCloudFrontPublicKey
	iam>DeleteGroup
	iam>DeleteGroupPolicy
	iam>DeleteInstanceProfile
	iam>DeleteLoginProfile
	iam>DeleteOpenIDConnectProvider
	iam>DeletePolicy
	iam>DeletePolicyVersion
	iam>DeleteRole
	iam>DeleteRolePermissionsBoundary
	iam>DeleteRolePolicy
	iam>DeleteSAMLProvider
	iam>DeleteServerCertificate
	iam>DeleteServiceLinkedRole
	iam>DeleteServiceSpecificCredential
	iam>DeleteSigningCertificate
	iam>DeleteSSHPublicKey

サービスプレフィックス	アクション
	iam:DeleteUser
	iam:DeleteUserPermissionsBoundary
	iam:DeleteUserPolicy
	iam:DeleteVirtualMFADevice
	iam:DetachGroupPolicy
	iam:DetachRolePolicy
	iam:DetachUserPolicy
	iam:EnableMFADevice
	iam:GenerateCredentialReport
	iam:GenerateOrganizationsAccessReport
	iam:GenerateServiceLastAccessedDetails
	iam:GetAccessKeyLastUsed
	iam:GetAccountAuthorizationDetails
	iam:GetAccountEmailAddress
	iam:GetAccountName
	iam:GetAccountPasswordPolicy
	iam:GetAccountSummary
	iam:GetCloudFrontPublicKey
	iam:GetContextKeysForCustomPolicy
	iam:GetContextKeysForPrincipalPolicy
	iam:GetCredentialReport

サービスプレフィックス	アクション
	iam:GetGroup
	iam:GetGroupPolicy
	iam:GetInstanceProfile
	iam:GetLoginProfile
	iam:GetMFADevice
	iam:GetOpenIDConnectProvider
	iam:GetOrganizationsAccessReport
	iam:GetPolicy
	iam:GetPolicyVersion
	iam:GetRole
	iam:GetRolePolicy
	iam:GetSAMLProvider
	iam:GetServerCertificate
	iam:GetServiceLastAccessedDetails
	iam:GetServiceLastAccessedDetailsWithEntities
	iam:GetServiceLinkedRoleDeletionStatus
	iam:GetSSHPublicKey
	iam:GetUser
	iam:GetUserPolicy
	iam:ListAccessKeys
	iam:ListAccountAliases

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>iam:ListAttachedGroupPolicies</li><li>iam:ListAttachedRolePolicies</li><li>iam:ListAttachedUserPolicies</li><li>iam:ListCloudFrontPublicKeys</li><li>iam:ListEntitiesForPolicy</li><li>iam:ListGroupPolicies</li><li>iam:ListGroups</li><li>iam:ListGroupsForUser</li><li>iam:ListInstanceProfiles</li><li>iam:ListInstanceProfilesForRole</li><li>iam:ListMFADevices</li><li>iam:ListOpenIDConnectProviders</li><li>iam:ListPolicies</li><li>iam:ListPoliciesGrantingServiceAccess</li><li>iam:ListPolicyVersions</li><li>iam:ListRolePolicies</li><li>iam:ListRoles</li><li>iam:ListSAMLProviders</li><li>iam:ListServerCertificates</li><li>iam:ListServiceSpecificCredentials</li><li>iam:ListSigningCertificates</li></ul>

サービスプレフィックス	アクション
	iam:ListSSHPublicKeys
	iam:ListSTSRegionalEndpointsStatus
	iam:ListUserPolicies
	iam:ListUsers
	iam:ListVirtualMFADevices
	iam:PutGroupPolicy
	iam:PutRolePermissionsBoundary
	iam:PutRolePolicy
	iam:PutUserPermissionsBoundary
	iam:PutUserPolicy
	iam:RemoveClientIDFromOpenIDConnectProvider
	iam:RemoveRoleFromInstanceProfile
	iam:RemoveUserFromGroup
	iam:ResetServiceSpecificCredential
	iam:ResyncMFADevice
	iam:SetDefaultPolicyVersion
	iam:SetSecurityTokenServicePreferences
	iam:SetSTSRegionalEndpointStatus
	iam:SimulateCustomPolicy
	iam:SimulatePrincipalPolicy
	iam:UpdateAccessKey

サービスプレフィックス	アクション
	iam:UpdateAccountEmailAddress
	iam:UpdateAccountName
	iam:UpdateAccountPasswordPolicy
	iam:UpdateAssumeRolePolicy
	iam:UpdateCloudFrontPublicKey
	iam:UpdateGroup
	iam:UpdateLoginProfile
	iam:UpdateOpenIDConnectProviderThumbprint
	iam:UpdateRole
	iam:UpdateRoleDescription
	iam:UpdateSAMLProvider
	iam:UpdateServerCertificate
	iam:UpdateServiceSpecificCredential
	iam:UpdateSigningCertificate
	iam:UpdateSSHPublicKey
	iam:UpdateUser
	iam:UploadCloudFrontPublicKey
	iam:UploadServerCertificate
	iam:UploadSigningCertificate
	iam:UploadSSHPublicKey



サービスプレフィックス	アクション
identitystore	identitystore:CreateGroup
	identitystore:CreateGroupMembership
	identitystore:CreateUser
	identitystore>DeleteGroup
	identitystore>DeleteGroupMembership
	identitystore>DeleteUser
	identitystore:DescribeGroup
	identitystore:DescribeGroupMembership
	identitystore:DescribeUser
	identitystore:GetGroupId
	identitystore:GetGroupMembershipId
	identitystore:GetUserId
	identitystore:IsMemberInGroups
	identitystore:ListGroupMemberships
	identitystore:ListGroupMembershipsForMember
	identitystore:ListGroups
	identitystore:ListUsers
	identitystore:UpdateGroup
	identitystore:UpdateUser

サービスプレフィックス	アクション
imagebuilder	imagebuilder:CancelImageCreation
	imagebuilder:CancelLifecycleExecution
	imagebuilder:CreateComponent
	imagebuilder:CreateContainerRecipe
	imagebuilder:CreateDistributionConfiguration
	imagebuilder:CreateImage
	imagebuilder:CreateImagePipeline
	imagebuilder:CreateImageRecipe
	imagebuilder:CreateInfrastructureConfiguration
	imagebuilder:CreateLifecyclePolicy
	imagebuilder:CreateWorkflow
	imagebuilder>DeleteComponent
	imagebuilder>DeleteContainerRecipe
	imagebuilder>DeleteDistributionConfiguration
	imagebuilder:DeleteImage
	imagebuilder:DeleteImagePipeline
	imagebuilder:DeleteImageRecipe
	imagebuilder:DeleteInfrastructureConfiguration
	imagebuilder>DeleteLifecyclePolicy
	imagebuilder>DeleteWorkflow
imagebuilder:GetComponentPolicy	

サービスプレフィックス	アクション
	<p>imagebuilder:GetContainerRecipePolicy</p> <p>imagebuilder:GetImagePolicy</p> <p>imagebuilder:GetImageRecipePolicy</p> <p>imagebuilder:GetLifecycleExecution</p> <p>imagebuilder:GetLifecyclePolicy</p> <p>imagebuilder:GetWorkflowExecution</p> <p>imagebuilder:GetWorkflowStepExecution</p> <p>imagebuilder:ImportComponent</p> <p>imagebuilder:ImportVmImage</p> <p>imagebuilder:ListComponentBuildVersions</p> <p>imagebuilder:ListComponents</p> <p>imagebuilder:ListContainerRecipes</p> <p>imagebuilder:ListDistributionConfigurations</p> <p>imagebuilder:ListImageBuildVersions</p> <p>imagebuilder:ListImagePackages</p> <p>imagebuilder:ListImagePipelineImages</p> <p>imagebuilder:ListImagePipelines</p> <p>imagebuilder:ListImageRecipes</p> <p>imagebuilder:ListImages</p> <p>imagebuilder:ListImageScanFindingAggregations</p> <p>imagebuilder:ListImageScanFindings</p>

サービスプレフィックス	アクション
	imagebuilder:ListInfrastructureConfigurations
	imagebuilder:ListLifecycleExecutionResources
	imagebuilder:ListLifecycleExecutions
	imagebuilder:ListLifecyclePolicies
	imagebuilder:ListWaitingWorkflowSteps
	imagebuilder:ListWorkflowExecutions
	imagebuilder:ListWorkflows
	imagebuilder:ListWorkflowStepExecutions
	imagebuilder:PutComponentPolicy
	imagebuilder:PutContainerRecipePolicy
	imagebuilder:PutImagePolicy
	imagebuilder:PutImageRecipePolicy
	imagebuilder:SendWorkflowStepAction
	imagebuilder:StartImagePipelineExecution
	imagebuilder:StartResourceStateUpdate
	imagebuilder:UpdateDistributionConfiguration
	imagebuilder:UpdateImagePipeline
	imagebuilder:UpdateInfrastructureConfiguration

サービスプレフィックス	アクション
inspector	inspector:AddAttributesToFindings
	inspector:CreateAssessmentTarget
	inspector:CreateAssessmentTemplate
	inspector:CreateExclusionsPreview
	inspector:CreateResourceGroup
	inspector>DeleteAssessmentRun
	inspector>DeleteAssessmentTarget
	inspector>DeleteAssessmentTemplate
	inspector:DescribeAssessmentRuns
	inspector:DescribeAssessmentTargets
	inspector:DescribeAssessmentTemplates
	inspector:DescribeCrossAccountAccessRole
	inspector:DescribeExclusions
	inspector:DescribeFindings
	inspector:DescribeResourceGroups
	inspector:DescribeRulesPackages
	inspector:GetAssessmentReport
	inspector:GetExclusionsPreview
	inspector:GetTelemetryMetadata
	inspector:ListAssessmentRunAgents
	inspector:ListAssessmentRuns

サービスプレフィックス	アクション
	<code>inspector:ListAssessmentTargets</code>
	<code>inspector:ListAssessmentTemplates</code>
	<code>inspector:ListEventSubscriptions</code>
	<code>inspector:ListExclusions</code>
	<code>inspector:ListFindings</code>
	<code>inspector:ListRulesPackages</code>
	<code>inspector:PreviewAgents</code>
	<code>inspector:RegisterCrossAccountAccessRole</code>
	<code>inspector:RemoveAttributesFromFindings</code>
	<code>inspector:StartAssessmentRun</code>
	<code>inspector:StopAssessmentRun</code>
	<code>inspector:SubscribeToEvent</code>
	<code>inspector:UnsubscribeFromEvent</code>
	<code>inspector:UpdateAssessmentTarget</code>

サービスプレフィックス	アクション
inspector2	inspector2:AssociateMember
	inspector2:BatchGetAccountStatus
	inspector2:BatchGetCodeSnippet
	inspector2:BatchGetFindingDetails
	inspector2:BatchGetFreeTrialInfo
	inspector2:BatchGetMemberEc2DeepInspectionStatus
	inspector2:BatchUpdateMemberEc2DeepInspectionStatus
	inspector2:CancelFindingsReport
	inspector2:CancelSbomExport
	inspector2:CreateCisScanConfiguration
	inspector2:CreateFilter
	inspector2:CreateFindingsReport
	inspector2:CreateSbomExport
	inspector2>DeleteCisScanConfiguration
	inspector2>DeleteFilter
	inspector2:DescribeOrganizationConfiguration
	inspector2:Disable
	inspector2:DisableDelegatedAdminAccount
	inspector2:DisassociateMember
	inspector2:Enable
	inspector2:EnableDelegatedAdminAccount

サービスプレフィックス	アクション
	<p>inspector2:GetCisScanReport</p> <p>inspector2:GetCisScanResultDetails</p> <p>inspector2:GetConfiguration</p> <p>inspector2:GetDelegatedAdminAccount</p> <p>inspector2:GetEc2DeepInspectionConfiguration</p> <p>inspector2:GetEncryptionKey</p> <p>inspector2:GetFindingsReportStatus</p> <p>inspector2:GetMember</p> <p>inspector2:GetSbomExport</p> <p>inspector2:ListAccountPermissions</p> <p>inspector2:ListCisScanConfigurations</p> <p>inspector2:ListCisScanResultsAggregatedByChecks</p> <p>inspector2:ListCisScanResultsAggregatedByTargetResource</p> <p>inspector2:ListCisScans</p> <p>inspector2:ListCoverage</p> <p>inspector2:ListCoverageStatistics</p> <p>inspector2:ListDelegatedAdminAccounts</p> <p>inspector2:ListFilters</p> <p>inspector2:ListFindingAggregations</p> <p>inspector2:ListFindings</p> <p>inspector2:ListMembers</p>



サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>inspector2:ListUsageTotals</li><li>inspector2:ResetEncryptionKey</li><li>inspector2:SearchVulnerabilities</li><li>inspector2:SendCisSessionHealth</li><li>inspector2:SendCisSessionTelemetry</li><li>inspector2:StartCisSession</li><li>inspector2:StopCisSession</li><li>inspector2:UpdateCisScanConfiguration</li><li>inspector2:UpdateConfiguration</li><li>inspector2:UpdateEc2DeepInspectionConfiguration</li><li>inspector2:UpdateEncryptionKey</li><li>inspector2:UpdateFilter</li><li>inspector2:UpdateOrganizationConfiguration</li><li>inspector2:UpdateOrgEc2DeepInspectionConfiguration</li></ul>

サービスプレフィックス	アクション
iot	iot:AcceptCertificateTransfer
	iot:AddThingToBillingGroup
	iot:AddThingToThingGroup
	iot:AssociateTargetsWithJob
	iot:AttachPolicy
	iot:AttachPrincipalPolicy
	iot:AttachSecurityProfile
	iot:AttachThingPrincipal
	iot:CancelAuditMitigationActionsTask
	iot:CancelAuditTask
	iot:CancelCertificateTransfer
	iot:CancelDetectMitigationActionsTask
	iot:CancelJob
	iot:CancelJobExecution
	iot:ClearDefaultAuthorizer
	iot:ConfirmTopicRuleDestination
	iot>CreateAuditSuppression
	iot>CreateAuthorizer
	iot>CreateBillingGroup
	iot>CreateCertificateFromCsr
	iot>CreateCertificateProvider

サービスプレフィックス	アクション
	iot:CreateCustomMetric
	iot:CreateDimension
	iot:CreateDomainConfiguration
	iot:CreateDynamicThingGroup
	iot:CreateFleetMetric
	iot:CreateJob
	iot:CreateJobTemplate
	iot:CreateKeysAndCertificate
	iot:CreateMitigationAction
	iot:CreateOTAUpdate
	iot:CreatePackage
	iot:CreatePackageVersion
	iot:CreatePolicy
	iot:CreatePolicyVersion
	iot:CreateProvisioningClaim
	iot:CreateProvisioningTemplate
	iot:CreateProvisioningTemplateVersion
	iot:CreateRoleAlias
	iot:CreateScheduledAudit
	iot:CreateSecurityProfile
	iot:CreateStream

サービスプレフィックス	アクション
	iot:CreateThing
	iot:CreateThingGroup
	iot:CreateThingType
	iot:CreateTopicRule
	iot:CreateTopicRuleDestination
	iot>DeleteAccountAuditConfiguration
	iot>DeleteAuditSuppression
	iot>DeleteAuthorizer
	iot>DeleteBillingGroup
	iot>DeleteCACertificate
	iot>DeleteCertificate
	iot>DeleteCertificateProvider
	iot>DeleteCustomMetric
	iot>DeleteDimension
	iot>DeleteDomainConfiguration
	iot>DeleteDynamicThingGroup
	iot>DeleteFleetMetric
	iot>DeleteJob
	iot>DeleteJobExecution
	iot>DeleteJobTemplate
	iot>DeleteMitigationAction

サービスプレフィックス	アクション
	iot:DeleteOTAUpdate
	iot:DeletePackage
	iot:DeletePackageVersion
	iot:DeletePolicy
	iot:DeletePolicyVersion
	iot:DeleteProvisioningTemplate
	iot:DeleteProvisioningTemplateVersion
	iot:DeleteRegistrationCode
	iot:DeleteRoleAlias
	iot:DeleteScheduledAudit
	iot:DeleteSecurityProfile
	iot:DeleteStream
	iot:DeleteThing
	iot:DeleteThingGroup
	iot:DeleteThingType
	iot:DeleteTopicRule
	iot:DeleteTopicRuleDestination
	iot:DeleteV2LoggingLevel
	iot:DeprecateThingType
	iot:DescribeAccountAuditConfiguration
	iot:DescribeAuditFinding

サービスプレフィックス	アクション
	iot:DescribeAuditMitigationActionsTask
	iot:DescribeAuditSuppression
	iot:DescribeAuditTask
	iot:DescribeAuthorizer
	iot:DescribeBillingGroup
	iot:DescribeCACertificate
	iot:DescribeCertificate
	iot:DescribeCertificateProvider
	iot:DescribeCustomMetric
	iot:DescribeDefaultAuthorizer
	iot:DescribeDetectMitigationActionsTask
	iot:DescribeDimension
	iot:DescribeDomainConfiguration
	iot:DescribeEndpoint
	iot:DescribeEventConfigurations
	iot:DescribeFleetMetric
	iot:DescribeIndex
	iot:DescribeJob
	iot:DescribeJobExecution
	iot:DescribeJobTemplate
	iot:DescribeManagedJobTemplate

サービスプレフィックス	アクション
	iot:DescribeMitigationAction
	iot:DescribeProvisioningTemplate
	iot:DescribeProvisioningTemplateVersion
	iot:DescribeRoleAlias
	iot:DescribeScheduledAudit
	iot:DescribeSecurityProfile
	iot:DescribeStream
	iot:DescribeThing
	iot:DescribeThingGroup
	iot:DescribeThingRegistrationTask
	iot:DescribeThingType
	iot:DetachPolicy
	iot:DetachPrincipalPolicy
	iot:DetachSecurityProfile
	iot:DetachThingPrincipal
	iot:DisableTopicRule
	iot:EnableTopicRule
	iot:GetBehaviorModelTrainingSummaries
	iot:GetBucketsAggregation
	iot:GetCardinality
	iot:GetEffectivePolicies

サービスプレフィックス	アクション
	iot:GetJobDocument
	iot:GetLoggingOptions
	iot:GetOTAUpdate
	iot:GetPackage
	iot:GetPackageConfiguration
	iot:GetPackageVersion
	iot:GetPercentiles
	iot:GetPolicy
	iot:GetPolicyVersion
	iot:GetRegistrationCode
	iot:GetStatistics
	iot:GetTopicRule
	iot:GetTopicRuleDestination
	iot:GetV2LoggingOptions
	iot:ListActiveViolations
	iot:ListAttachedPolicies
	iot:ListAuditFindings
	iot:ListAuditMitigationActionsExecutions
	iot:ListAuditMitigationActionsTasks
	iot:ListAuditSuppressions
	iot:ListAuditTasks



サービスプレフィックス	アクション
	iot:ListAuthorizers
	iot:ListBillingGroups
	iot:ListCACertificates
	iot:ListCertificateProviders
	iot:ListCertificates
	iot:ListCertificatesByCA
	iot:ListCustomMetrics
	iot:ListDetectMitigationActionsExecutions
	iot:ListDetectMitigationActionsTasks
	iot:ListDimensions
	iot:ListDomainConfigurations
	iot:ListFleetMetrics
	iot:ListIndices
	iot:ListJobExecutionsForJob
	iot:ListJobExecutionsForThing
	iot:ListJobs
	iot:ListJobTemplates
	iot:ListManagedJobTemplates
	iot:ListMetricValues
	iot:ListMitigationActions
	iot:ListOTAUpdates

サービスプレフィックス	アクション
	iot:ListOutgoingCertificates
	iot:ListPackages
	iot:ListPackageVersions
	iot:ListPolicies
	iot:ListPolicyPrincipals
	iot:ListPolicyVersions
	iot:ListPrincipalPolicies
	iot:ListPrincipalThings
	iot:ListProvisioningTemplates
	iot:ListProvisioningTemplateVersions
	iot:ListRelatedResourcesForAuditFinding
	iot:ListRoleAliases
	iot:ListScheduledAudits
	iot:ListSecurityProfiles
	iot:ListSecurityProfilesForTarget
	iot:ListStreams
	iot:ListTargetsForPolicy
	iot:ListTargetsForSecurityProfile
	iot:ListThingGroups
	iot:ListThingGroupsForThing
	iot:ListThingPrincipals

サービスプレフィックス	アクション
	iot:ListThingRegistrationTaskReports
	iot:ListThingRegistrationTasks
	iot:ListThings
	iot:ListThingsInBillingGroup
	iot:ListThingsInThingGroup
	iot:ListThingTypes
	iot:ListTopicRuleDestinations
	iot:ListTopicRules
	iot:ListV2LoggingLevels
	iot:ListViolationEvents
	iot:PutVerificationStateOnViolation
	iot:RegisterCACertificate
	iot:RegisterCertificate
	iot:RegisterCertificateWithoutCA
	iot:RegisterThing
	iot:RejectCertificateTransfer
	iot:RemoveThingFromBillingGroup
	iot:RemoveThingFromThingGroup
	iot:ReplaceTopicRule
	iot:SearchIndex
	iot:SetDefaultAuthorizer

サービスプレフィックス	アクション
	<code>iot:SetDefaultPolicyVersion</code>
	<code>iot:SetLoggingOptions</code>
	<code>iot:SetV2LoggingLevel</code>
	<code>iot:SetV2LoggingOptions</code>
	<code>iot:StartAuditMitigationActionsTask</code>
	<code>iot:StartDetectMitigationActionsTask</code>
	<code>iot:StartOnDemandAuditTask</code>
	<code>iot:StartThingRegistrationTask</code>
	<code>iot:StopThingRegistrationTask</code>
	<code>iot:TestAuthorization</code>
	<code>iot:TestInvokeAuthorizer</code>
	<code>iot:TransferCertificate</code>
	<code>iot:UpdateAccountAuditConfiguration</code>
	<code>iot:UpdateAuditSuppression</code>
	<code>iot:UpdateAuthorizer</code>
	<code>iot:UpdateBillingGroup</code>
	<code>iot:UpdateCACertificate</code>
	<code>iot:UpdateCertificate</code>
	<code>iot:UpdateCertificateProvider</code>
	<code>iot:UpdateCustomMetric</code>
	<code>iot:UpdateDimension</code>

サービスプレフィックス	アクション
	iot:UpdateDomainConfiguration
	iot:UpdateDynamicThingGroup
	iot:UpdateEventConfigurations
	iot:UpdateFleetMetric
	iot:UpdateIndexingConfiguration
	iot:UpdateJob
	iot:UpdateMitigationAction
	iot:UpdatePackage
	iot:UpdatePackageConfiguration
	iot:UpdatePackageVersion
	iot:UpdateProvisioningTemplate
	iot:UpdateRoleAlias
	iot:UpdateScheduledAudit
	iot:UpdateSecurityProfile
	iot:UpdateStream
	iot:UpdateThing
	iot:UpdateThingGroup
	iot:UpdateThingGroupsForThing
	iot:UpdateTopicRuleDestination
	iot:ValidateSecurityProfileBehaviors

サービスプレフィックス	アクション
iotanalytics	iotanalytics:CancelPipelineReprocessing
	iotanalytics:CreateChannel
	iotanalytics:CreateDataset
	iotanalytics:CreateDatasetContent
	iotanalytics:CreateDatastore
	iotanalytics:CreatePipeline
	iotanalytics>DeleteChannel
	iotanalytics>DeleteDataset
	iotanalytics>DeleteDatasetContent
	iotanalytics>DeleteDatastore
	iotanalytics>DeletePipeline
	iotanalytics:DescribeChannel
	iotanalytics:DescribeDataset
	iotanalytics:DescribeDatastore
	iotanalytics:DescribeLoggingOptions
	iotanalytics:DescribePipeline
	iotanalytics:GetDatasetContent
	iotanalytics:ListChannels
	iotanalytics:ListDatasetContents
	iotanalytics:ListDatasets
iotanalytics:ListDatastores	

サービスプレフィックス	アクション
	iotanalytics:ListPipelines
	iotanalytics:PutLoggingOptions
	iotanalytics:RunPipelineActivity
	iotanalytics:SampleChannelData
	iotanalytics:StartPipelineReprocessing
	iotanalytics:UpdateChannel
	iotanalytics:UpdateDataset
	iotanalytics:UpdateDatastore
	iotanalytics:UpdatePipeline
iotdeviceadvisor	iotdeviceadvisor:CreateSuiteDefinition
	iotdeviceadvisor>DeleteSuiteDefinition
	iotdeviceadvisor:GetEndpoint
	iotdeviceadvisor:GetSuiteDefinition
	iotdeviceadvisor:GetSuiteRun
	iotdeviceadvisor:GetSuiteRunReport
	iotdeviceadvisor:ListSuiteDefinitions
	iotdeviceadvisor:ListSuiteRuns
	iotdeviceadvisor:StartSuiteRun
	iotdeviceadvisor:StopSuiteRun
	iotdeviceadvisor:UpdateSuiteDefinition

サービスプレフィックス	アクション
iotevents	iotevents:BatchAcknowledgeAlarm
	iotevents:BatchDeleteDetector
	iotevents:BatchDisableAlarm
	iotevents:BatchEnableAlarm
	iotevents:BatchResetAlarm
	iotevents:BatchSnoozeAlarm
	iotevents:BatchUpdateDetector
	iotevents:CreateAlarmModel
	iotevents:CreateDetectorModel
	iotevents:CreateInput
	iotevents>DeleteAlarmModel
	iotevents>DeleteDetectorModel
	iotevents>DeleteInput
	iotevents:DescribeAlarm
	iotevents:DescribeAlarmModel
	iotevents:DescribeDetector
	iotevents:DescribeDetectorModel
	iotevents:DescribeDetectorModelAnalysis
	iotevents:DescribeInput
	iotevents:DescribeLoggingOptions
	iotevents:GetDetectorModelAnalysisResults



サービスプレフィックス	アクション
	<code>iotevents:ListAlarmModels</code>
	<code>iotevents:ListAlarmModelVersions</code>
	<code>iotevents:ListAlarms</code>
	<code>iotevents:ListDetectorModels</code>
	<code>iotevents:ListDetectorModelVersions</code>
	<code>iotevents:ListDetectors</code>
	<code>iotevents:ListInputRoutings</code>
	<code>iotevents:ListInputs</code>
	<code>iotevents:PutLoggingOptions</code>
	<code>iotevents:StartDetectorModelAnalysis</code>
	<code>iotevents:UpdateAlarmModel</code>
	<code>iotevents:UpdateDetectorModel</code>
	<code>iotevents:UpdateInput</code>
<code>iotfleethub</code>	<code>iotfleethub:CreateApplication</code>
	<code>iotfleethub&gt;DeleteApplication</code>
	<code>iotfleethub:DescribeApplication</code>
	<code>iotfleethub:ListApplications</code>
	<code>iotfleethub:UpdateApplication</code>

サービスプレフィックス	アクション
iotsitewise	iotsitewise:AssociateAssets
	iotsitewise:AssociateTimeSeriesToAssetProperty
	iotsitewise:BatchAssociateProjectAssets
	iotsitewise:BatchDisassociateProjectAssets
	iotsitewise:BatchGetAssetPropertyValue
	iotsitewise:BatchGetAssetPropertyValueHistory
	iotsitewise:BatchPutAssetPropertyValue
	iotsitewise:CreateAccessPolicy
	iotsitewise:CreateAsset
	iotsitewise:CreateAssetModel
	iotsitewise:CreateAssetModelCompositeModel
	iotsitewise:CreateBulkImportJob
	iotsitewise:CreateDashboard
	iotsitewise:CreateGateway
	iotsitewise:CreatePortal
	iotsitewise:CreateProject
	iotsitewise>DeleteAccessPolicy
	iotsitewise>DeleteAsset
	iotsitewise>DeleteAssetModel
	iotsitewise>DeleteAssetModelCompositeModel
iotsitewise>DeleteDashboard	

サービスプレフィックス	アクション
	<p>iotsitewise:DeleteGateway</p> <p>iotsitewise:DeletePortal</p> <p>iotsitewise:DeleteProject</p> <p>iotsitewise:DeleteTimeSeries</p> <p>iotsitewise:DescribeAccessPolicy</p> <p>iotsitewise:DescribeAsset</p> <p>iotsitewise:DescribeAssetCompositeModel</p> <p>iotsitewise:DescribeAssetModel</p> <p>iotsitewise:DescribeAssetModelCompositeModel</p> <p>iotsitewise:DescribeAssetProperty</p> <p>iotsitewise:DescribeBulkImportJob</p> <p>iotsitewise:DescribeDashboard</p> <p>iotsitewise:DescribeDefaultEncryptionConfiguration</p> <p>iotsitewise:DescribeGateway</p> <p>iotsitewise:DescribeGatewayCapabilityConfiguration</p> <p>iotsitewise:DescribeLoggingOptions</p> <p>iotsitewise:DescribePortal</p> <p>iotsitewise:DescribeProject</p> <p>iotsitewise:DescribeStorageConfiguration</p> <p>iotsitewise:DescribeTimeSeries</p> <p>iotsitewise:DisassociateAssets</p>

サービスプレフィックス	アクション
	iotsitewise:DisassociateTimeSeriesFromAssetProperty
	iotsitewise:ExecuteAction
	iotsitewise:ExecuteQuery
	iotsitewise:ListAccessPolicies
	iotsitewise:ListActions
	iotsitewise:ListAssetModelCompositeModels
	iotsitewise:ListAssetModelProperties
	iotsitewise:ListAssetModels
	iotsitewise:ListAssetProperties
	iotsitewise:ListAssetRelationships
	iotsitewise:ListAssets
	iotsitewise:ListAssociatedAssets
	iotsitewise:ListBulkImportJobs
	iotsitewise:ListCompositionRelationships
	iotsitewise:ListDashboards
	iotsitewise:ListGateways
	iotsitewise:ListPortals
	iotsitewise:ListProjectAssets
	iotsitewise:ListProjects
	iotsitewise:ListTimeSeries
	iotsitewise:PutDefaultEncryptionConfiguration

サービスプレフィックス	アクション
	iotsitewise:PutLoggingOptions
	iotsitewise:PutStorageConfiguration
	iotsitewise:UpdateAccessPolicy
	iotsitewise:UpdateAsset
	iotsitewise:UpdateAssetModel
	iotsitewise:UpdateAssetModelCompositeModel
	iotsitewise:UpdateAssetProperty
	iotsitewise:UpdateDashboard
	iotsitewise:UpdateGateway
	iotsitewise:UpdateGatewayCapabilityConfiguration
	iotsitewise:UpdatePortal
	iotsitewise:UpdateProject

サービスプレフィックス	アクション
iottwinmaker	iottwinmaker:CancelMetadataTransferJob
	iottwinmaker:CreateComponentType
	iottwinmaker:CreateEntity
	iottwinmaker:CreateMetadataTransferJob
	iottwinmaker:CreateScene
	iottwinmaker:CreateSyncJob
	iottwinmaker:CreateWorkspace
	iottwinmaker>DeleteComponentType
	iottwinmaker>DeleteEntity
	iottwinmaker>DeleteScene
	iottwinmaker>DeleteSyncJob
	iottwinmaker>DeleteWorkspace
	iottwinmaker:ExecuteQuery
	iottwinmaker:GetMetadataTransferJob
	iottwinmaker:GetPricingPlan
	iottwinmaker:GetScene
	iottwinmaker:GetSyncJob
	iottwinmaker:ListComponents
	iottwinmaker:ListComponentTypes
	iottwinmaker:ListEntities
	iottwinmaker:ListMetadataTransferJobs

サービスプレフィックス	アクション
	iottwinmaker:ListProperties
	iottwinmaker:ListScenes
	iottwinmaker:ListSyncJobs
	iottwinmaker:ListSyncResources
	iottwinmaker:ListWorkspaces
	iottwinmaker:UpdateComponentType
	iottwinmaker:UpdateEntity
	iottwinmaker:UpdatePricingPlan
	iottwinmaker:UpdateScene
	iottwinmaker:UpdateWorkspace

サービスプレフィックス	アクション
iotwireless	iotwireless:AssociateAwsAccountWithPartnerAccount
	iotwireless:AssociateMulticastGroupWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithMulticastGroup
	iotwireless:AssociateWirelessDeviceWithThing
	iotwireless:AssociateWirelessGatewayWithCertificate
	iotwireless:AssociateWirelessGatewayWithThing
	iotwireless:CancelMulticastGroupSession
	iotwireless:CreateDestination
	iotwireless:CreateDeviceProfile
	iotwireless:CreateFuotaTask
	iotwireless:CreateMulticastGroup
	iotwireless:CreateNetworkAnalyzerConfiguration
	iotwireless:CreateServiceProfile
	iotwireless:CreateWirelessDevice
	iotwireless:CreateWirelessGateway
	iotwireless:CreateWirelessGatewayTask
	iotwireless:CreateWirelessGatewayTaskDefinition
	iotwireless>DeleteDestination
	iotwireless>DeleteDeviceProfile
	iotwireless>DeleteFuotaTask



サービスプレフィックス	アクション
	iotwireless:DeleteMulticastGroup
	iotwireless:DeleteNetworkAnalyzerConfiguration
	iotwireless:DeleteQueuedMessages
	iotwireless:DeleteServiceProfile
	iotwireless:DeleteWirelessDevice
	iotwireless:DeleteWirelessDeviceImportTask
	iotwireless:DeleteWirelessGateway
	iotwireless:DeleteWirelessGatewayTask
	iotwireless:DeleteWirelessGatewayTaskDefinition
	iotwireless:DeregisterWirelessDevice
	iotwireless:DisassociateAwsAccountFromPartnerAccount
	iotwireless:DisassociateMulticastGroupFromFwotaTask
	iotwireless:DisassociateWirelessDeviceFromFwotaTask
	iotwireless:DisassociateWirelessDeviceFromMulticastGroup
	iotwireless:DisassociateWirelessDeviceFromThing
	iotwireless:DisassociateWirelessGatewayFromCertificate
	iotwireless:DisassociateWirelessGatewayFromThing
	iotwireless:GetDestination
	iotwireless:GetDeviceProfile
	iotwireless:GetEventConfigurationByResourceTypes
	iotwireless:GetFwotaTask

サービスプレフィックス	アクション
	iotwireless:GetLogLevelsByResourceTypes
	iotwireless:GetMetricConfiguration
	iotwireless:GetMetrics
	iotwireless:GetMulticastGroup
	iotwireless:GetMulticastGroupSession
	iotwireless:GetNetworkAnalyzerConfiguration
	iotwireless:GetPartnerAccount
	iotwireless:GetPosition
	iotwireless:GetPositionConfiguration
	iotwireless:GetPositionEstimate
	iotwireless:GetResourceEventConfiguration
	iotwireless:GetResourceLogLevel
	iotwireless:GetResourcePosition
	iotwireless:GetServiceEndpoint
	iotwireless:GetServiceProfile
	iotwireless:GetWirelessDevice
	iotwireless:GetWirelessDeviceImportTask
	iotwireless:GetWirelessDeviceStatistics
	iotwireless:GetWirelessGateway
	iotwireless:GetWirelessGatewayCertificate
	iotwireless:GetWirelessGatewayFirmwareInformation

サービスプレフィックス	アクション
	<p>iotwireless:GetWirelessGatewayStatistics</p> <p>iotwireless:GetWirelessGatewayTask</p> <p>iotwireless:GetWirelessGatewayTaskDefinition</p> <p>iotwireless:ListDestinations</p> <p>iotwireless:ListDeviceProfiles</p> <p>iotwireless:ListDevicesForWirelessDeviceImportTask</p> <p>iotwireless:ListEventConfigurations</p> <p>iotwireless:ListFuotaTasks</p> <p>iotwireless:ListMulticastGroups</p> <p>iotwireless:ListMulticastGroupsByFuotaTask</p> <p>iotwireless:ListNetworkAnalyzerConfigurations</p> <p>iotwireless:ListPartnerAccounts</p> <p>iotwireless:ListPositionConfigurations</p> <p>iotwireless:ListQueuedMessages</p> <p>iotwireless:ListServiceProfiles</p> <p>iotwireless:ListWirelessDeviceImportTasks</p> <p>iotwireless:ListWirelessDevices</p> <p>iotwireless:ListWirelessGateways</p> <p>iotwireless:ListWirelessGatewayTaskDefinitions</p> <p>iotwireless:PutPositionConfiguration</p> <p>iotwireless:PutResourceLogLevel</p>

サービスプレフィックス	アクション
	iotwireless:ResetAllResourceLogLevels
	iotwireless:ResetResourceLogLevel
	iotwireless:SendDataToMulticastGroup
	iotwireless:SendDataToWirelessDevice
	iotwireless:StartBulkAssociateWirelessDeviceWithMulticastGroup
	iotwireless:StartBulkDisassociateWirelessDeviceFromMulticastGroup
	iotwireless:StartFuotaTask
	iotwireless:StartMulticastGroupSession
	iotwireless:StartNetworkAnalyzerStream
	iotwireless:StartSingleWirelessDeviceImportTask
	iotwireless:StartWirelessDeviceImportTask
	iotwireless:TestWirelessDevice
	iotwireless:UpdateDestination
	iotwireless:UpdateEventConfigurationByResourceTypes
	iotwireless:UpdateFuotaTask
	iotwireless:UpdateLogLevelsByResourceTypes
	iotwireless:UpdateMetricConfiguration
	iotwireless:UpdateMulticastGroup
	iotwireless:UpdateNetworkAnalyzerConfiguration
	iotwireless:UpdatePartnerAccount

サービスプレフィックス	アクション
	iotwireless:UpdatePosition iotwireless:UpdateResourceEventConfiguration iotwireless:UpdateResourcePosition iotwireless:UpdateWirelessDevice iotwireless:UpdateWirelessDeviceImportTask iotwireless:UpdateWirelessGateway

サービスプレフィックス	アクション
ivs	ivs:BatchGetChannel
	ivs:BatchGetStreamKey
	ivs:BatchStartViewerSessionRevocation
	ivs:CreateChannel
	ivs:CreateEncoderConfiguration
	ivs:CreateParticipantToken
	ivs:CreatePlaybackRestrictionPolicy
	ivs:CreateRecordingConfiguration
	ivs:CreateStorageConfiguration
	ivs:CreateStreamKey
	ivs>DeleteChannel
	ivs>DeleteEncoderConfiguration
	ivs>DeletePlaybackKeyPair
	ivs>DeletePlaybackRestrictionPolicy
	ivs>DeleteRecordingConfiguration
	ivs>DeleteStorageConfiguration
	ivs>DeleteStreamKey
	ivs:DisconnectParticipant
	ivs:GetChannel
	ivs:GetComposition
	ivs:GetEncoderConfiguration

サービスプレフィックス	アクション
	<code>ivs:GetParticipant</code>
	<code>ivs:GetPlaybackKeyPair</code>
	<code>ivs:GetPlaybackRestrictionPolicy</code>
	<code>ivs:GetRecordingConfiguration</code>
	<code>ivs:GetStorageConfiguration</code>
	<code>ivs:GetStream</code>
	<code>ivs:GetStreamKey</code>
	<code>ivs:GetStreamSession</code>
	<code>ivs:ImportPlaybackKeyPair</code>
	<code>ivs:ListChannels</code>
	<code>ivs:ListCompositions</code>
	<code>ivs:ListEncoderConfigurations</code>
	<code>ivs:ListParticipantEvents</code>
	<code>ivs:ListParticipants</code>
	<code>ivs:ListPlaybackKeyPairs</code>
	<code>ivs:ListPlaybackRestrictionPolicies</code>
	<code>ivs:ListRecordingConfigurations</code>
	<code>ivs:ListStorageConfigurations</code>
	<code>ivs:ListStreamKeys</code>
	<code>ivs:ListStreams</code>
	<code>ivs:ListStreamSessions</code>

サービスプレフィックス	アクション
	ivs:PutMetadata
	ivs:StartComposition
	ivs:StartViewerSessionRevocation
	ivs:StopComposition
	ivs:StopStream
	ivs:UpdateChannel
	ivs:UpdatePlaybackRestrictionPolicy
ivschat	ivschat:CreateChatToken
	ivschat:CreateLoggingConfiguration
	ivschat:CreateRoom
	ivschat>DeleteLoggingConfiguration
	ivschat>DeleteMessage
	ivschat>DeleteRoom
	ivschat:DisconnectUser
	ivschat:GetLoggingConfiguration
	ivschat:GetRoom
	ivschat:ListLoggingConfigurations
	ivschat:ListRooms
	ivschat:SendEvent
	ivschat:UpdateLoggingConfiguration
	ivschat:UpdateRoom



サービスプレフィックス	アクション
kafka	kafka:BatchAssociateScramSecret
	kafka:BatchDisassociateScramSecret
	kafka:CreateCluster
	kafka:CreateClusterV2
	kafka:CreateConfiguration
	kafka:CreateReplicator
	kafka:CreateVpcConnection
	kafka>DeleteCluster
	kafka>DeleteClusterPolicy
	kafka>DeleteConfiguration
	kafka>DeleteReplicator
	kafka>DeleteVpcConnection
	kafka:DescribeCluster
	kafka:DescribeClusterOperation
	kafka:DescribeClusterOperationV2
	kafka:DescribeClusterV2
	kafka:DescribeConfiguration
	kafka:DescribeConfigurationRevision
	kafka:DescribeVpcConnection
	kafka:GetBootstrapBrokers
	kafka:GetClusterPolicy

サービスプレフィックス	アクション
	kafka:GetCompatibleKafkaVersions
	kafka:ListClientVpcConnections
	kafka:ListClusterOperations
	kafka:ListClusterOperationsV2
	kafka:ListClusters
	kafka:ListClustersV2
	kafka:ListConfigurationRevisions
	kafka:ListConfigurations
	kafka:ListKafkaVersions
	kafka:ListNodes
	kafka:ListReplicators
	kafka:ListScramSecrets
	kafka:ListVpcConnections
	kafka:PutClusterPolicy
	kafka:RebootBroker
	kafka:RejectClientVpcConnection
	kafka:UpdateBrokerCount
	kafka:UpdateBrokerStorage
	kafka:UpdateBrokerType
	kafka:UpdateClusterConfiguration
	kafka:UpdateClusterKafkaVersion

サービスプレフィックス	アクション
	kafka:UpdateConfiguration
	kafka:UpdateConnectivity
	kafka:UpdateMonitoring
	kafka:UpdateReplicationInfo
	kafka:UpdateSecurity
	kafka:UpdateStorage
kafkaconnect	kafkaconnect:CreateConnector
	kafkaconnect:CreateCustomPlugin
	kafkaconnect:CreateWorkerConfiguration
	kafkaconnect>DeleteConnector
	kafkaconnect>DeleteCustomPlugin
	kafkaconnect>DeleteWorkerConfiguration
	kafkaconnect:DescribeConnector
	kafkaconnect:DescribeCustomPlugin
	kafkaconnect:DescribeWorkerConfiguration
	kafkaconnect:ListConnectors
	kafkaconnect:ListCustomPlugins
	kafkaconnect:ListWorkerConfigurations
	kafkaconnect:UpdateConnector

サービスプレフィックス	アクション
kendra	kendra:AssociateEntitiesToExperience
	kendra:AssociatePersonasToEntities
	kendra:BatchDeleteDocument
	kendra:BatchDeleteFeaturedResultsSet
	kendra:BatchGetDocumentStatus
	kendra:BatchPutDocument
	kendra:ClearQuerySuggestions
	kendra:CreateAccessControlConfiguration
	kendra:CreateDataSource
	kendra:CreateExperience
	kendra:CreateFaq
	kendra:CreateFeaturedResultsSet
	kendra:CreateIndex
	kendra:CreateQuerySuggestionsBlockList
	kendra:CreateThesaurus
	kendra>DeleteDataSource
	kendra>DeleteExperience
	kendra>DeleteFaq
	kendra>DeleteIndex
	kendra>DeletePrincipalMapping
	kendra>DeleteQuerySuggestionsBlockList

サービスプレフィックス	アクション
	kendra:DeleteThesaurus
	kendra:DescribeAccessControlConfiguration
	kendra:DescribeDataSource
	kendra:DescribeExperience
	kendra:DescribeFaq
	kendra:DescribeFeaturedResultsSet
	kendra:DescribeIndex
	kendra:DescribePrincipalMapping
	kendra:DescribeQuerySuggestionsBlockList
	kendra:DescribeQuerySuggestionsConfig
	kendra:DescribeThesaurus
	kendra:DisassociateEntitiesFromExperience
	kendra:DisassociatePersonasFromEntities
	kendra:GetQuerySuggestions
	kendra:GetSnapshots
	kendra:ListAccessControlConfigurations
	kendra:ListDataSources
	kendra:ListDataSourceSyncJobs
	kendra:ListEntityPersonas
	kendra:ListExperienceEntities
	kendra:ListExperiences

サービスプレフィックス	アクション
	kendra:ListFaqs
	kendra:ListFeaturedResultsSets
	kendra:ListGroupsOlderThanOrderingId
	kendra:ListIndices
	kendra:ListQuerySuggestionsBlockLists
	kendra:ListThesauri
	kendra:PutPrincipalMapping
	kendra:Query
	kendra:Retrieve
	kendra:StartDataSourceSyncJob
	kendra:StopDataSourceSyncJob
	kendra:SubmitFeedback
	kendra:UpdateDataSource
	kendra:UpdateExperience
	kendra:UpdateFeaturedResultsSet
	kendra:UpdateIndex
	kendra:UpdateQuerySuggestionsBlockList
	kendra:UpdateQuerySuggestionsConfig
	kendra:UpdateThesaurus

サービスプレフィックス	アクション
kinesis	kinesis:CreateStream
	kinesis:DecreaseStreamRetentionPeriod
	kinesis>DeleteStream
	kinesis:DeregisterStreamConsumer
	kinesis:DescribeLimits
	kinesis:DescribeStream
	kinesis:DescribeStreamConsumer
	kinesis:DescribeStreamSummary
	kinesis:DisableEnhancedMonitoring
	kinesis:EnableEnhancedMonitoring
	kinesis:IncreaseStreamRetentionPeriod
	kinesis:ListShards
	kinesis:ListStreamConsumers
	kinesis:ListStreams
	kinesis:MergeShards
	kinesis:RegisterStreamConsumer
	kinesis:SplitShard
	kinesis:StartStreamEncryption
	kinesis:StopStreamEncryption
	kinesis:UpdateShardCount
	kinesis:UpdateStreamMode

サービスプレフィックス	アクション
kinesisanalytics	kinesisanalytics:AddApplicationCloudWatchLoggingOption
	kinesisanalytics:AddApplicationInput
	kinesisanalytics:AddApplicationInputProcessingConfiguration
	kinesisanalytics:AddApplicationOutput
	kinesisanalytics:AddApplicationReferenceDataSource
	kinesisanalytics:AddApplicationVpcConfiguration
	kinesisanalytics:CreateApplication
	kinesisanalytics:CreateApplicationPresignedUrl
	kinesisanalytics:CreateApplicationSnapshot
	kinesisanalytics>DeleteApplication
	kinesisanalytics>DeleteApplicationCloudWatchLoggingOption
	kinesisanalytics>DeleteApplicationInputProcessingConfiguration
	kinesisanalytics>DeleteApplicationOutput
	kinesisanalytics>DeleteApplicationReferenceDataSource
	kinesisanalytics>DeleteApplicationSnapshot
	kinesisanalytics>DeleteApplicationVpcConfiguration
	kinesisanalytics:DescribeApplication
	kinesisanalytics:DescribeApplicationSnapshot
	kinesisanalytics:DescribeApplicationVersion
	kinesisanalytics:DiscoverInputSchema
	kinesisanalytics:ListApplications



サービスプレフィックス	アクション
	kinesisanalytics:ListApplicationSnapshots
	kinesisanalytics:ListApplicationVersions
	kinesisanalytics:RollbackApplication
	kinesisanalytics:StartApplication
	kinesisanalytics:StopApplication
	kinesisanalytics:UpdateApplication
	kinesisanalytics:UpdateApplicationMaintenanceConfiguration

サービスプレフィックス	アクション
kms	kms:CancelKeyDeletion
	kms:ConnectCustomKeyStore
	kms:CreateAlias
	kms:CreateCustomKeyStore
	kms:CreateGrant
	kms:CreateKey
	kms:Decrypt
	kms>DeleteAlias
	kms>DeleteCustomKeyStore
	kms>DeleteImportedKeyMaterial
	kms:DescribeCustomKeyStores
	kms:DescribeKey
	kms:DisableKey
	kms:DisableKeyRotation
	kms:DisconnectCustomKeyStore
	kms:EnableKey
	kms:EnableKeyRotation
	kms:Encrypt
	kms:GenerateDataKey
	kms:GenerateDataKeyPair
	kms:GenerateDataKeyPairWithoutPlaintext

サービスプレフィックス	アクション
	<p>kms:GenerateDataKeyWithoutPlaintext</p> <p>kms:GenerateMac</p> <p>kms:GenerateRandom</p> <p>kms:GetKeyPolicy</p> <p>kms:GetKeyRotationStatus</p> <p>kms:GetParametersForImport</p> <p>kms:GetPublicKey</p> <p>kms:ImportKeyMaterial</p> <p>kms:ListAliases</p> <p>kms:ListGrants</p> <p>kms:ListKeyPolicies</p> <p>kms:ListKeys</p> <p>kms:ListRetirableGrants</p> <p>kms:ReplicateKey</p> <p>kms:RetireGrant</p> <p>kms:RevokeGrant</p> <p>kms:ScheduleKeyDeletion</p> <p>kms:Sign</p> <p>kms:UpdateAlias</p> <p>kms:UpdateCustomKeyStore</p> <p>kms:UpdateKeyDescription</p>

サービスプレフィックス	アクション
	kms:UpdatePrimaryRegion kms:Verify kms:VerifyMac

サービスプレフィックス	アクション
lambda	lambda:AddLayerVersionPermission
	lambda:AddLayerVersionPermission
	lambda:AddPermission
	lambda:AddPermission
	lambda:AddPermission
	lambda:CreateAlias
	lambda:CreateAlias
	lambda:CreateCodeSigningConfig
	lambda:CreateEventSourceMapping
	lambda:CreateEventSourceMapping
	lambda:CreateFunction
	lambda:CreateFunction
	lambda:CreateFunctionUrlConfig
	lambda>DeleteAlias
	lambda>DeleteAlias
	lambda>DeleteCodeSigningConfig
	lambda>DeleteEventSourceMapping
	lambda>DeleteEventSourceMapping
	lambda>DeleteFunction
	lambda>DeleteFunction
	lambda>DeleteFunctionCodeSigningConfig

サービスプレフィックス	アクション
	lambda:DeleteFunctionConcurrency
	lambda:DeleteFunctionConcurrency
	lambda:DeleteFunctionEventInvokeConfig
	lambda:DeleteFunctionUrlConfig
	lambda:DeleteLayerVersion
	lambda:DeleteLayerVersion
	lambda:DeleteProvisionedConcurrencyConfig
	lambda:GetAccountSettings
	lambda:GetAccountSettings
	lambda:GetAlias
	lambda:GetAlias
	lambda:GetCodeSigningConfig
	lambda:GetEventSourceMapping
	lambda:GetEventSourceMapping
	lambda:GetFunction
	lambda:GetFunction
	lambda:GetFunction
	lambda:GetFunctionCodeSigningConfig
	lambda:GetFunctionConcurrency
	lambda:GetFunctionConfiguration
	lambda:GetFunctionConfiguration

サービスプレフィックス	アクション
	lambda:GetFunctionConfiguration
	lambda:GetFunctionEventInvokeConfig
	lambda:GetFunctionUrlConfig
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersionPolicy
	lambda:GetLayerVersionPolicy
	lambda:GetPolicy
	lambda:GetPolicy
	lambda:GetPolicy
	lambda:GetProvisionedConcurrencyConfig
	lambda:GetRuntimeManagementConfig
	lambda:ListAliases
	lambda:ListAliases
	lambda:ListCodeSigningConfigs
	lambda:ListEventSourceMappings
	lambda:ListEventSourceMappings
	lambda:ListFunctionEventInvokeConfigs
	lambda:ListFunctions

サービスプレフィックス	アクション
	lambda:ListFunctions
	lambda:ListFunctionsByCodeSigningConfig
	lambda:ListFunctionUrlConfigs
	lambda:ListLayers
	lambda:ListLayers
	lambda:ListLayerVersions
	lambda:ListLayerVersions
	lambda:ListProvisionedConcurrencyConfigs
	lambda:ListVersionsByFunction
	lambda:ListVersionsByFunction
	lambda:PublishLayerVersion
	lambda:PublishLayerVersion
	lambda:PublishVersion
	lambda:PublishVersion
	lambda:PutFunctionCodeSigningConfig
	lambda:PutFunctionConcurrency
	lambda:PutFunctionConcurrency
	lambda:PutFunctionEventInvokeConfig
	lambda:PutProvisionedConcurrencyConfig
	lambda:PutRuntimeManagementConfig
	lambda:RemoveLayerVersionPermission



サービスプレフィックス	アクション
	lambda:RemoveLayerVersionPermission
	lambda:RemovePermission
	lambda:RemovePermission
	lambda:RemovePermission
	lambda:UpdateAlias
	lambda:UpdateAlias
	lambda:UpdateCodeSigningConfig
	lambda:UpdateEventSourceMapping
	lambda:UpdateEventSourceMapping
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionEventInvokeConfig
	lambda:UpdateFunctionUrlConfig

サービスプレフィックス	アクション
lex	lex:BatchCreateCustomVocabularyItem
	lex:BatchDeleteCustomVocabularyItem
	lex:BatchUpdateCustomVocabularyItem
	lex:BuildBotLocale
	lex:CreateBotAlias
	lex:CreateBotVersion
	lex:CreateExport
	lex:CreateIntentVersion
	lex:CreateResourcePolicy
	lex:CreateSlotTypeVersion
	lex:CreateTestSetDiscrepancyReport
	lex:CreateUploadUrl
	lex>DeleteBot
	lex>DeleteBotChannelAssociation
	lex>DeleteExport
	lex>DeleteImport
	lex>DeleteIntentVersion
	lex>DeleteResourcePolicy
	lex>DeleteSlotTypeVersion
	lex>DeleteTestSet
	lex>DeleteUtterances

サービスプレフィックス	アクション
	lex:DescribeBotAlias
	lex:DescribeBotRecommendation
	lex:DescribeBotResourceGeneration
	lex:DescribeBotVersion
	lex:DescribeCustomVocabularyMetadata
	lex:DescribeExport
	lex:DescribeImport
	lex:DescribeResourcePolicy
	lex:DescribeTestExecution
	lex:DescribeTestSet
	lex:DescribeTestSetDiscrepancyReport
	lex:DescribeTestSetGeneration
	lex:GenerateBotElement
	lex:GetBot
	lex:GetBotAlias
	lex:GetBotAliases
	lex:GetBotChannelAssociation
	lex:GetBotChannelAssociations
	lex:GetBots
	lex:GetBotVersions
	lex:GetBuiltinIntent

サービスプレフィックス	アクション
	lex:GetBuiltinIntents
	lex:GetBuiltinSlotTypes
	lex:GetExport
	lex:GetImport
	lex:GetIntent
	lex:GetIntents
	lex:GetIntentVersions
	lex:GetMigration
	lex:GetMigrations
	lex:GetSlotType
	lex:GetSlotTypes
	lex:GetSlotTypeVersions
	lex:GetTestExecutionArtifactsUrl
	lex:GetUtterancesView
	lex:ListBotAliases
	lex:ListBotRecommendations
	lex:ListBotResourceGenerations
	lex:ListBots
	lex:ListBotVersions
	lex:ListBuiltinIntents
	lex:ListBuiltinSlotTypes

サービスプレフィックス	アクション
	lex:ListCustomVocabularyItems
	lex:ListExports
	lex:ListImports
	lex:ListIntentMetrics
	lex:ListIntentPaths
	lex:ListRecommendedIntents
	lex:ListSessionAnalyticsData
	lex:ListSessionMetrics
	lex:ListTestExecutionResultItems
	lex:ListTestExecutions
	lex:ListTestSets
	lex:PutBot
	lex:PutBotAlias
	lex:PutIntent
	lex:PutSlotType
	lex:SearchAssociatedTranscripts
	lex:StartBotRecommendation
	lex:StartImport
	lex:StartMigration
	lex:StartTestExecution
	lex:StartTestSetGeneration

サービスプレフィックス	アクション
	lex:StopBotRecommendation lex:UpdateBotAlias lex:UpdateBotRecommendation lex:UpdateExport lex:UpdateResourcePolicy
license-manager-linux-subscriptions	license-manager-linux-subscriptions:GetServiceSettings license-manager-linux-subscriptions:ListLinuxSubscriptionInstances license-manager-linux-subscriptions:ListLinuxSubscriptions license-manager-linux-subscriptions:UpdateServiceSettings

サービスプレフィックス	アクション
lightsail	lightsail:AllocateStaticIp
	lightsail:AttachCertificateToDistribution
	lightsail:AttachDisk
	lightsail:AttachInstancesToLoadBalancer
	lightsail:AttachLoadBalancerTlsCertificate
	lightsail:AttachStaticIp
	lightsail:CloseInstancePublicPorts
	lightsail:CopySnapshot
	lightsail>CreateBucket
	lightsail>CreateBucketAccessKey
	lightsail>CreateCertificate
	lightsail>CreateCloudFormationStack
	lightsail>CreateContactMethod
	lightsail>CreateContainerService
	lightsail>CreateContainerServiceDeployment
	lightsail>CreateContainerServiceRegistryLogin
	lightsail:CreateDisk
	lightsail:CreateDiskFromSnapshot
	lightsail:CreateDiskSnapshot
	lightsail:CreateDistribution
	lightsail:CreateDomain

サービスプレフィックス	アクション
	lightsail:CreateGUISessionAccessDetails
	lightsail:CreateInstances
	lightsail:CreateInstancesFromSnapshot
	lightsail:CreateInstanceSnapshot
	lightsail:CreateKeyPair
	lightsail:CreateLoadBalancer
	lightsail:CreateLoadBalancerTlsCertificate
	lightsail:CreateRelationalDatabase
	lightsail:CreateRelationalDatabaseFromSnapshot
	lightsail:CreateRelationalDatabaseSnapshot
	lightsail>DeleteAlarm
	lightsail>DeleteAutoSnapshot
	lightsail>DeleteBucket
	lightsail>DeleteBucketAccessKey
	lightsail>DeleteCertificate
	lightsail>DeleteContactMethod
	lightsail>DeleteContainerImage
	lightsail>DeleteContainerService
	lightsail>DeleteDisk
	lightsail>DeleteDiskSnapshot
	lightsail>DeleteDistribution



サービスプレフィックス	アクション
	lightsail:DeleteDomain
	lightsail:DeleteDomainEntry
	lightsail:DeleteInstance
	lightsail:DeleteInstanceSnapshot
	lightsail:DeleteKeyPair
	lightsail:DeleteKnownHostKeys
	lightsail:DeleteLoadBalancer
	lightsail:DeleteLoadBalancerTlsCertificate
	lightsail:DeleteRelationalDatabase
	lightsail:DeleteRelationalDatabaseSnapshot
	lightsail:DetachCertificateFromDistribution
	lightsail:DetachDisk
	lightsail:DetachInstancesFromLoadBalancer
	lightsail:DetachStaticIp
	lightsail:DisableAddOn
	lightsail:DownloadDefaultKeyPair
	lightsail:EnableAddOn
	lightsail:ExportSnapshot
	lightsail:GetActiveNames
	lightsail:GetAlarms
	lightsail:GetAutoSnapshots

サービスプレフィックス	アクション
	lightsail:GetBlueprints
	lightsail:GetBucketAccessKeys
	lightsail:GetBucketBundles
	lightsail:GetBucketMetricData
	lightsail:GetBuckets
	lightsail:GetBundles
	lightsail:GetCertificates
	lightsail:GetCloudFormationStackRecords
	lightsail:GetContactMethods
	lightsail:GetContainerAPIMetadata
	lightsail:GetContainerImages
	lightsail:GetContainerLog
	lightsail:GetContainerServiceDeployments
	lightsail:GetContainerServiceMetricData
	lightsail:GetContainerServicePowers
	lightsail:GetContainerServices
	lightsail:GetCostEstimate
	lightsail:GetDisk
	lightsail:GetDisks
	lightsail:GetDiskSnapshot
	lightsail:GetDiskSnapshots

サービスプレフィックス	アクション
	lightsail:GetDistributionBundles
	lightsail:GetDistributionLatestCacheReset
	lightsail:GetDistributionMetricData
	lightsail:GetDistributions
	lightsail:GetDomain
	lightsail:GetExportSnapshotRecords
	lightsail:GetInstance
	lightsail:GetInstanceMetricData
	lightsail:GetInstancePortStates
	lightsail:GetInstances
	lightsail:GetInstanceSnapshot
	lightsail:GetInstanceSnapshots
	lightsail:GetInstanceState
	lightsail:GetKeyPair
	lightsail:GetKeyPairs
	lightsail:GetLoadBalancer
	lightsail:GetLoadBalancerMetricData
	lightsail:GetLoadBalancers
	lightsail:GetLoadBalancerTlsCertificates
	lightsail:GetLoadBalancerTlsPolicies
	lightsail:GetOperation

サービスプレフィックス	アクション
	lightsail:GetOperations
	lightsail:GetOperationsForResource
	lightsail:GetRegions
	lightsail:GetRelationalDatabase
	lightsail:GetRelationalDatabaseBlueprints
	lightsail:GetRelationalDatabaseBundles
	lightsail:GetRelationalDatabaseEvents
	lightsail:GetRelationalDatabaseLogEvents
	lightsail:GetRelationalDatabaseLogStreams
	lightsail:GetRelationalDatabaseMasterUserPassword
	lightsail:GetRelationalDatabaseMetricData
	lightsail:GetRelationalDatabaseParameters
	lightsail:GetRelationalDatabases
	lightsail:GetRelationalDatabaseSnapshot
	lightsail:GetRelationalDatabaseSnapshots
	lightsail:GetSetupHistory
	lightsail:GetStaticIp
	lightsail:GetStaticIps
	lightsail:ImportKeyPair
	lightsail:IsVpcPeered
	lightsail:OpenInstancePublicPorts

サービスプレフィックス	アクション
	lightsail:PeerVpc
	lightsail:PutAlarm
	lightsail:PutInstancePublicPorts
	lightsail:RebootInstance
	lightsail:RebootRelationalDatabase
	lightsail:RegisterContainerImage
	lightsail:ReleaseStaticIp
	lightsail:ResetDistributionCache
	lightsail:SendContactMethodVerification
	lightsail:SetIpAddressType
	lightsail:SetResourceAccessForBucket
	lightsail:SetupInstanceHttps
	lightsail:StartGUISession
	lightsail:StartInstance
	lightsail:StartRelationalDatabase
	lightsail:StopGUISession
	lightsail:StopInstance
	lightsail:StopRelationalDatabase
	lightsail:TestAlarm
	lightsail:UnpeerVpc
	lightsail:UpdateBucket

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>lightsail:UpdateBucketBundle</li><li>lightsail:UpdateContainerService</li><li>lightsail:UpdateDistribution</li><li>lightsail:UpdateDistributionBundle</li><li>lightsail:UpdateDomainEntry</li><li>lightsail:UpdateInstanceMetadataOptions</li><li>lightsail:UpdateLoadBalancerAttribute</li><li>lightsail:UpdateRelationalDatabase</li><li>lightsail:UpdateRelationalDatabaseParameters</li></ul>

サービスプレフィックス	アクション
ログ	logs:AssociateKmsKey
	logs:CancelExportTask
	logs:CreateExportTask
	logs:CreateLogAnomalyDetector
	logs:CreateLogGroup
	logs:CreateLogStream
	logs>DeleteDataProtectionPolicy
	logs>DeleteDelivery
	logs>DeleteDeliveryDestination
	logs>DeleteDeliveryDestinationPolicy
	logs>DeleteDeliverySource
	logs>DeleteDestination
	logs>DeleteLogGroup
	logs>DeleteLogStream
	logs>DeleteMetricFilter
	logs>DeleteQueryDefinition
	logs>DeleteResourcePolicy
	logs>DeleteRetentionPolicy
	logs>DeleteSubscriptionFilter
	logs:DescribeAccountPolicies
	logs:DescribeDeliveries

サービスプレフィックス	アクション
	logs:DescribeDeliveryDestinations
	logs:DescribeDeliverySources
	logs:DescribeDestinations
	logs:DescribeExportTasks
	logs:DescribeLogGroups
	logs:DescribeLogStreams
	logs:DescribeMetricFilters
	logs:DescribeQueries
	logs:DescribeQueryDefinitions
	logs:DescribeResourcePolicies
	logs:DescribeSubscriptionFilters
	logs:DisassociateKmsKey
	logs:GetDataProtectionPolicy
	logs:GetDelivery
	logs:GetDeliveryDestination
	logs:GetDeliveryDestinationPolicy
	logs:GetDeliverySource
	logs:GetLogGroupFields
	logs:GetLogRecord
	logs:GetQueryResults
	logs>ListAnomalies



サービスプレフィックス	アクション
	logs:ListLogAnomalyDetectors
	logs:PutDataProtectionPolicy
	logs:PutDeliveryDestination
	logs:PutDeliveryDestinationPolicy
	logs:PutDeliverySource
	logs:PutDestination
	logs:PutDestinationPolicy
	logs:PutMetricFilter
	logs:PutQueryDefinition
	logs:PutResourcePolicy
	logs:PutRetentionPolicy
	logs:PutSubscriptionFilter
	logs:StartLiveTail
	logs:StartQuery
	logs:StopQuery
	logs:TestMetricFilter

サービスプレフィックス	アクション
lookoutequipment	lookoutequipment:CreateDataset
	lookoutequipment:CreateInferenceScheduler
	lookoutequipment:CreateLabel
	lookoutequipment:CreateLabelGroup
	lookoutequipment:CreateModel
	lookoutequipment>DeleteDataset
	lookoutequipment>DeleteInferenceScheduler
	lookoutequipment>DeleteLabel
	lookoutequipment>DeleteLabelGroup
	lookoutequipment>DeleteModel
	lookoutequipment>DeleteResourcePolicy
	lookoutequipment>DeleteRetrainingScheduler
	lookoutequipment:DescribeDataIngestionJob
	lookoutequipment:DescribeDataset
	lookoutequipment:DescribeInferenceScheduler
	lookoutequipment:DescribeLabel
	lookoutequipment:DescribeLabelGroup
	lookoutequipment:DescribeModel
	lookoutequipment:DescribeModelVersion
	lookoutequipment:DescribeResourcePolicy
lookoutequipment:DescribeRetrainingScheduler	

サービスプレフィックス	アクション
	lookoutequipment:ImportDataset
	lookoutequipment:ImportModelVersion
	lookoutequipment:ListDataIngestionJobs
	lookoutequipment:ListDatasets
	lookoutequipment:ListInferenceEvents
	lookoutequipment:ListInferenceExecutions
	lookoutequipment:ListInferenceSchedulers
	lookoutequipment:ListLabelGroups
	lookoutequipment:ListLabels
	lookoutequipment:ListModels
	lookoutequipment:ListModelVersions
	lookoutequipment:ListRetrainingSchedulers
	lookoutequipment:ListSensorStatistics
	lookoutequipment:PutResourcePolicy
	lookoutequipment:StartDataIngestionJob
	lookoutequipment:StartInferenceScheduler
	lookoutequipment:StartRetrainingScheduler
	lookoutequipment:StopInferenceScheduler
	lookoutequipment:StopRetrainingScheduler
	lookoutequipment:UpdateActiveModelVersion
	lookoutequipment:UpdateInferenceScheduler

サービスプレフィックス	アクション
	lookoutequipment:UpdateLabelGroup lookoutequipment:UpdateModel lookoutequipment:UpdateRetrainingScheduler

サービスプレフィックス	アクション
lookoutmetrics	lookoutmetrics:ActivateAnomalyDetector
	lookoutmetrics:BackTestAnomalyDetector
	lookoutmetrics:CreateAlert
	lookoutmetrics:CreateAnomalyDetector
	lookoutmetrics:CreateMetricSet
	lookoutmetrics:DeactivateAnomalyDetector
	lookoutmetrics>DeleteAlert
	lookoutmetrics>DeleteAnomalyDetector
	lookoutmetrics:DescribeAlert
	lookoutmetrics:DescribeAnomalyDetectionExecutions
	lookoutmetrics:DescribeAnomalyDetector
	lookoutmetrics:DescribeMetricSet
	lookoutmetrics:DetectMetricSetConfig
	lookoutmetrics:GetAnomalyGroup
	lookoutmetrics:GetDataQualityMetrics
	lookoutmetrics:GetFeedback
	lookoutmetrics:GetSampleData
	lookoutmetrics:ListAlerts
	lookoutmetrics:ListAnomalyDetectors
	lookoutmetrics:ListAnomalyGroupRelatedMetrics
	lookoutmetrics:ListAnomalyGroupSummaries

サービスプレフィックス	アクション
	lookoutmetrics:ListAnomalyGroupTimeSeries
	lookoutmetrics:ListMetricSets
	lookoutmetrics:PutFeedback
	lookoutmetrics:UpdateAlert
	lookoutmetrics:UpdateAnomalyDetector
	lookoutmetrics:UpdateMetricSet

サービスプレフィックス	アクション
lookoutvision	lookoutvision:CreateDataset
	lookoutvision:CreateModel
	lookoutvision:CreateProject
	lookoutvision>DeleteDataset
	lookoutvision>DeleteModel
	lookoutvision>DeleteProject
	lookoutvision:DescribeDataset
	lookoutvision:DescribeModel
	lookoutvision:DescribeModelPackagingJob
	lookoutvision:DescribeProject
	lookoutvision:DetectAnomalies
	lookoutvision:ListDatasetEntries
	lookoutvision:ListModelPackagingJobs
	lookoutvision:ListModels
	lookoutvision:ListProjects
	lookoutvision:StartModel
	lookoutvision:StartModelPackagingJob
	lookoutvision:StopModel
	lookoutvision:UpdateDatasetEntries

サービスプレフィックス	アクション
m2	m2:CancelBatchJobExecution
	m2:CreateApplication
	m2:CreateDataSetImportTask
	m2:CreateDeployment
	m2:CreateEnvironment
	m2>DeleteApplication
	m2>DeleteApplicationFromEnvironment
	m2>DeleteEnvironment
	m2:GetApplication
	m2:GetApplicationVersion
	m2:GetBatchJobExecution
	m2:GetDataSetDetails
	m2:GetDataSetImportTask
	m2:GetDeployment
	m2:GetEnvironment
	m2:GetSignedBluinsightsUrl
	m2:ListApplications
	m2:ListApplicationVersions
	m2:ListBatchJobDefinitions
	m2:ListBatchJobExecutions
	m2:ListBatchJobRestartPoints



サービスプレフィックス	アクション
	m2:ListDataSetImportHistory
	m2:ListDataSets
	m2:ListDeployments
	m2:ListEngineVersions
	m2:ListEnvironments
	m2:StartApplication
	m2:StartBatchJob
	m2:StopApplication
	m2:UpdateApplication
	m2:UpdateEnvironment

サービスプレフィックス	アクション
managedblockchain	managedblockchain:CreateAccessor
	managedblockchain:CreateMember
	managedblockchain:CreateNetwork
	managedblockchain:CreateNode
	managedblockchain:CreateProposal
	managedblockchain>DeleteAccessor
	managedblockchain>DeleteMember
	managedblockchain>DeleteNode
	managedblockchain:GetAccessor
	managedblockchain:GetMember
	managedblockchain:GetNetwork
	managedblockchain:GetNode
	managedblockchain:GetProposal
	managedblockchain:InvokeRpcPolygonMainnet
	managedblockchain:InvokeRpcPolygonMumbaiTestnet
	managedblockchain:ListAccessors
	managedblockchain:ListInvitations
	managedblockchain:ListMembers
	managedblockchain:ListNetworks
	managedblockchain:ListNodes
	managedblockchain:ListProposals

サービスプレフィックス	アクション
	managedblockchain:ListProposalVotes managedblockchain:RejectInvitation managedblockchain:UpdateMember managedblockchain:UpdateNode managedblockchain:VoteOnProposal

サービスプレフィックス	アクション
mediacconnect	mediacconnect:AddBridgeOutputs
	mediacconnect:AddBridgeSources
	mediacconnect:AddFlowMediaStreams
	mediacconnect:AddFlowOutputs
	mediacconnect:AddFlowSources
	mediacconnect:AddFlowVpcInterfaces
	mediacconnect:CreateBridge
	mediacconnect:CreateFlow
	mediacconnect:CreateGateway
	mediacconnect>DeleteBridge
	mediacconnect>DeleteFlow
	mediacconnect>DeleteGateway
	mediacconnect:DeregisterGatewayInstance
	mediacconnect:DescribeBridge
	mediacconnect:DescribeFlow
	mediacconnect:DescribeFlowSourceMetadata
	mediacconnect:DescribeGateway
	mediacconnect:DescribeGatewayInstance
	mediacconnect:DescribeOffering
	mediacconnect:DescribeReservation
	mediacconnect:GrantFlowEntitlements

サービスプレフィックス	アクション
	mediacconnect:ListBridges
	mediacconnect:ListEntitlements
	mediacconnect:ListFlows
	mediacconnect:ListGatewayInstances
	mediacconnect:ListGateways
	mediacconnect:ListOfferings
	mediacconnect:ListReservations
	mediacconnect:PurchaseOffering
	mediacconnect:RemoveBridgeOutput
	mediacconnect:RemoveBridgeSource
	mediacconnect:RemoveFlowMediaStream
	mediacconnect:RemoveFlowOutput
	mediacconnect:RemoveFlowSource
	mediacconnect:RemoveFlowVpcInterface
	mediacconnect:RevokeFlowEntitlement
	mediacconnect:StartFlow
	mediacconnect:StopFlow
	mediacconnect:UpdateBridge
	mediacconnect:UpdateBridgeOutput
	mediacconnect:UpdateBridgeSource
	mediacconnect:UpdateBridgeState

サービスプレフィックス	アクション
	mediacconnect:UpdateFlow
	mediacconnect:UpdateFlowEntitlement
	mediacconnect:UpdateFlowMediaStream
	mediacconnect:UpdateFlowOutput
	mediacconnect:UpdateFlowSource
	mediacconnect:UpdateGatewayInstance

サービスプレフィックス	アクション
mediaconvert	mediaconvert:AssociateCertificate
	mediaconvert:CancelJob
	mediaconvert:CreateJob
	mediaconvert:CreateJobTemplate
	mediaconvert:CreatePreset
	mediaconvert:CreateQueue
	mediaconvert>DeleteJobTemplate
	mediaconvert>DeletePolicy
	mediaconvert>DeletePreset
	mediaconvert>DeleteQueue
	mediaconvert:DescribeEndpoints
	mediaconvert:DisassociateCertificate
	mediaconvert:GetJob
	mediaconvert:GetJobTemplate
	mediaconvert:GetPolicy
	mediaconvert:GetPreset
	mediaconvert:GetQueue
	mediaconvert:ListJobs
	mediaconvert:ListJobTemplates
	mediaconvert:ListPresets
	mediaconvert:ListQueues

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>mediaconvert:PutPolicy</li><li>mediaconvert:UpdateJobTemplate</li><li>mediaconvert:UpdatePreset</li><li>mediaconvert:UpdateQueue</li></ul>



サービスプレフィックス	アクション
medialive	medialive:AcceptInputDeviceTransfer
	medialive:BatchDelete
	medialive:BatchStart
	medialive:BatchStop
	medialive:BatchUpdateSchedule
	medialive:CancelInputDeviceTransfer
	medialive:ClaimDevice
	medialive:CreateChannel
	medialive:CreateCloudWatchAlarmTemplate
	medialive:CreateCloudWatchAlarmTemplateGroup
	medialive:CreateEventBridgeRuleTemplate
	medialive:CreateEventBridgeRuleTemplateGroup
	medialive:CreateInput
	medialive:CreateInputSecurityGroup
	medialive:CreateMultiplex
	medialive:CreateMultiplexProgram
	medialive:CreatePartnerInput
	medialive:CreateSignalMap
	medialive>DeleteChannel
	medialive>DeleteCloudWatchAlarmTemplate
	medialive>DeleteCloudWatchAlarmTemplateGroup

サービスプレフィックス	アクション
	medialive:DeleteEventBridgeRuleTemplate
	medialive:DeleteEventBridgeRuleTemplateGroup
	medialive:DeleteInput
	medialive:DeleteInputSecurityGroup
	medialive:DeleteMultiplex
	medialive:DeleteMultiplexProgram
	medialive:DeleteReservation
	medialive:DeleteSchedule
	medialive:DeleteSignalMap
	medialive:DescribeAccountConfiguration
	medialive:DescribeChannel
	medialive:DescribeInput
	medialive:DescribeInputDevice
	medialive:DescribeInputDeviceThumbnail
	medialive:DescribeInputSecurityGroup
	medialive:DescribeMultiplex
	medialive:DescribeMultiplexProgram
	medialive:DescribeOffering
	medialive:DescribeReservation
	medialive:DescribeSchedule
	medialive:DescribeThumbnails

サービスプレフィックス	アクション
	medialive:GetCloudWatchAlarmTemplate
	medialive:GetCloudWatchAlarmTemplateGroup
	medialive:GetEventBridgeRuleTemplate
	medialive:GetEventBridgeRuleTemplateGroup
	medialive:GetSignalMap
	medialive:ListChannels
	medialive:ListCloudWatchAlarmTemplateGroups
	medialive:ListCloudWatchAlarmTemplates
	medialive:ListEventBridgeRuleTemplateGroups
	medialive:ListEventBridgeRuleTemplates
	medialive:ListInputDevices
	medialive:ListInputDeviceTransfers
	medialive:ListInputs
	medialive:ListInputSecurityGroups
	medialive:ListMultiplexes
	medialive:ListMultiplexPrograms
	medialive:ListOfferings
	medialive:ListReservations
	medialive:ListSignalMaps
	medialive:PurchaseOffering
	medialive:RebootInputDevice

サービスプレフィックス	アクション
	medialive:RejectInputDeviceTransfer
	medialive:RestartChannelPipelines
	medialive:StartChannel
	medialive:StartDeleteMonitorDeployment
	medialive:StartInputDevice
	medialive:StartInputDeviceMaintenanceWindow
	medialive:StartMonitorDeployment
	medialive:StartMultiplex
	medialive:StartUpdateSignalMap
	medialive:StopChannel
	medialive:StopInputDevice
	medialive:StopMultiplex
	medialive:TransferInputDevice
	medialive:UpdateAccountConfiguration
	medialive:UpdateChannel
	medialive:UpdateChannelClass
	medialive:UpdateCloudWatchAlarmTemplate
	medialive:UpdateCloudWatchAlarmTemplateGroup
	medialive:UpdateEventBridgeRuleTemplate
	medialive:UpdateEventBridgeRuleTemplateGroup
	medialive:UpdateInput

サービスプレフィックス	アクション
	medialive:UpdateInputDevice
	medialive:UpdateInputSecurityGroup
	medialive:UpdateMultiplex
	medialive:UpdateMultiplexProgram
	medialive:UpdateReservation

サービスプレフィックス	アクション
mediapackage	mediapackage:ConfigureLogs
	mediapackage:CreateChannel
	mediapackage:CreateHarvestJob
	mediapackage:CreateOriginEndpoint
	mediapackage>DeleteChannel
	mediapackage>DeleteOriginEndpoint
	mediapackage:DescribeChannel
	mediapackage:DescribeHarvestJob
	mediapackage:DescribeOriginEndpoint
	mediapackage:ListChannels
	mediapackage:ListHarvestJobs
	mediapackage:ListOriginEndpoints
	mediapackage:ListTagsForResource
	mediapackage:RotateChannelCredentials
	mediapackage:RotateIngestEndpointCredentials
	mediapackage:TagResource
	mediapackage:UntagResource
	mediapackage:UpdateChannel
	mediapackage:UpdateOriginEndpoint

サービスプレフィックス	アクション
mediapackage-vod	mediapackage-vod:ConfigureLogs
	mediapackage-vod:CreateAsset
	mediapackage-vod:CreatePackagingConfiguration
	mediapackage-vod:CreatePackagingGroup
	mediapackage-vod>DeleteAsset
	mediapackage-vod>DeletePackagingConfiguration
	mediapackage-vod>DeletePackagingGroup
	mediapackage-vod:DescribeAsset
	mediapackage-vod:DescribePackagingConfiguration
	mediapackage-vod:DescribePackagingGroup
	mediapackage-vod:ListAssets
	mediapackage-vod:ListPackagingConfigurations
	mediapackage-vod:ListPackagingGroups
	mediapackage-vod:ListTagsForResource
	mediapackage-vod:TagResource
	mediapackage-vod:UntagResource
	mediapackage-vod:UpdatePackagingGroup

サービスプレフィックス	アクション
mediastore	mediastore:CreateContainer
	mediastore>DeleteContainer
	mediastore>DeleteContainerPolicy
	mediastore>DeleteCorsPolicy
	mediastore>DeleteLifecyclePolicy
	mediastore>DeleteMetricPolicy
	mediastore:DescribeContainer
	mediastore:GetContainerPolicy
	mediastore:GetCorsPolicy
	mediastore:GetLifecyclePolicy
	mediastore:GetMetricPolicy
	mediastore:ListContainers
	mediastore:PutContainerPolicy
	mediastore:PutCorsPolicy
	mediastore:PutLifecyclePolicy
	mediastore:PutMetricPolicy
	mediastore:StartAccessLogging
	mediastore:StopAccessLogging



サービスプレフィックス	アクション
mediatailor	mediatailor:ConfigureLogsForPlaybackConfiguration
	mediatailor:CreateChannel
	mediatailor:CreateLiveSource
	mediatailor:CreatePrefetchSchedule
	mediatailor:CreateProgram
	mediatailor:CreateSourceLocation
	mediatailor:CreateVodSource
	mediatailor>DeleteChannel
	mediatailor>DeleteChannelPolicy
	mediatailor>DeleteLiveSource
	mediatailor>DeletePlaybackConfiguration
	mediatailor>DeletePrefetchSchedule
	mediatailor>DeleteProgram
	mediatailor>DeleteSourceLocation
	mediatailor>DeleteVodSource
	mediatailor:DescribeChannel
	mediatailor:DescribeLiveSource
	mediatailor:DescribeProgram
	mediatailor:DescribeSourceLocation
	mediatailor:DescribeVodSource
	mediatailor:GetChannelPolicy

サービスプレフィックス	アクション
	mediatailor:GetChannelSchedule
	mediatailor:GetPlaybackConfiguration
	mediatailor:GetPrefetchSchedule
	mediatailor:ListAlerts
	mediatailor:ListChannels
	mediatailor:ListLiveSources
	mediatailor:ListPlaybackConfigurations
	mediatailor:ListPrefetchSchedules
	mediatailor:ListSourceLocations
	mediatailor:ListVodSources
	mediatailor:PutChannelPolicy
	mediatailor:PutPlaybackConfiguration
	mediatailor:StartChannel
	mediatailor:StopChannel
	mediatailor:UpdateChannel
	mediatailor:UpdateLiveSource
	mediatailor:UpdateProgram
	mediatailor:UpdateSourceLocation
	mediatailor:UpdateVodSource

サービスプレフィックス	アクション
memorydb	memorydb:BatchUpdateCluster
	memorydb:CopySnapshot
	memorydb>CreateAcl
	memorydb>CreateCluster
	memorydb>CreateParameterGroup
	memorydb>CreateSnapshot
	memorydb>CreateSubnetGroup
	memorydb>CreateUser
	memorydb>DeleteAcl
	memorydb>DeleteCluster
	memorydb>DeleteParameterGroup
	memorydb>DeleteSnapshot
	memorydb>DeleteSubnetGroup
	memorydb>DeleteUser
	memorydb:DescribeAcls
	memorydb:DescribeClusters
	memorydb:DescribeEngineVersions
	memorydb:DescribeEvents
	memorydb:DescribeParameterGroups
	memorydb:DescribeParameters
memorydb:DescribeReservedNodes	

サービスプレフィックス	アクション
	memorydb:DescribeReservedNodesOfferings
	memorydb:DescribeServiceUpdates
	memorydb:DescribeSnapshots
	memorydb:DescribeSubnetGroups
	memorydb:DescribeUsers
	memorydb:FailoverShard
	memorydb:ListAllowedNodeTypeUpdates
	memorydb:PurchaseReservedNodesOffering
	memorydb:ResetParameterGroup
	memorydb:UpdateAcl
	memorydb:UpdateCluster
	memorydb:UpdateParameterGroup
	memorydb:UpdateSubnetGroup
	memorydb:UpdateUser

サービスプレフィックス	アクション
mgh	mgh:AssociateCreatedArtifact
	mgh:AssociateDiscoveredResource
	mgh>CreateHomeRegionControl
	mgh>CreateProgressUpdateStream
	mgh>DeleteHomeRegionControl
	mgh>DeleteProgressUpdateStream
	mgh:DescribeApplicationState
	mgh:DescribeHomeRegionControls
	mgh:DescribeMigrationTask
	mgh:DisassociateCreatedArtifact
	mgh:DisassociateDiscoveredResource
	mgh:GetHomeRegion
	mgh:ImportMigrationTask
	mgh>ListApplicationStates
	mgh>ListCreatedArtifacts
	mgh>ListDiscoveredResources
	mgh>ListMigrationTasks
	mgh>ListProgressUpdateStreams
	mgh:NotifyApplicationState
	mgh:NotifyMigrationTaskState
	mgh:PutResourceAttributes

サービスプレフィックス	アクション
mgn	mgn:ArchiveApplication
	mgn:ArchiveWave
	mgn:AssociateApplications
	mgn:AssociateSourceServers
	mgn:ChangeServerLifeCycleState
	mgn:CreateApplication
	mgn:CreateConnector
	mgn:CreateLaunchConfigurationTemplate
	mgn:CreateReplicationConfigurationTemplate
	mgn:CreateWave
	mgn>DeleteApplication
	mgn>DeleteConnector
	mgn>DeleteJob
	mgn>DeleteLaunchConfigurationTemplate
	mgn>DeleteReplicationConfigurationTemplate
	mgn>DeleteSourceServer
	mgn>DeleteVcenterClient
	mgn>DeleteWave
	mgn:DescribeJobLogItems
	mgn:DescribeJobs
	mgn:DescribeLaunchConfigurationTemplates

サービスプレフィックス	アクション
	mgn:DescribeReplicationConfigurationTemplates
	mgn:DescribeVcenterClients
	mgn:DisassociateApplications
	mgn:DisassociateSourceServers
	mgn:DisconnectFromService
	mgn:FinalizeCutover
	mgn:GetReplicationConfiguration
	mgn:InitializeService
	mgn:ListConnectors
	mgn:ListExportErrors
	mgn:ListExports
	mgn:ListImportErrors
	mgn:ListImports
	mgn:ListManagedAccounts
	mgn:ListSourceServerActions
	mgn:ListTemplateActions
	mgn:MarkAsArchived
	mgn:PauseReplication
	mgn:PutSourceServerAction
	mgn:PutTemplateAction
	mgn:RemoveSourceServerAction

サービスプレフィックス	アクション
	mgn:RemoveTemplateAction
	mgn:ResumeReplication
	mgn:RetryDataReplication
	mgn:StartCutover
	mgn:StartExport
	mgn:StartImport
	mgn:StartReplication
	mgn:StartTest
	mgn:StopReplication
	mgn:TerminateTargetInstances
	mgn:UnarchiveApplication
	mgn:UnarchiveWave
	mgn:UpdateApplication
	mgn:UpdateConnector
	mgn:UpdateLaunchConfigurationTemplate
	mgn:UpdateReplicationConfiguration
	mgn:UpdateReplicationConfigurationTemplate
	mgn:UpdateSourceServer
	mgn:UpdateSourceServerReplicationType
	mgn:UpdateWave



サービスプレフィックス	アクション
migrationhub-strategy	migrationhub-strategy:GetAntiPattern
	migrationhub-strategy:GetApplicationComponentDetails
	migrationhub-strategy:GetApplicationComponentStrategies
	migrationhub-strategy:GetAssessment
	migrationhub-strategy:GetImportFileTask
	migrationhub-strategy:GetLatestAssessmentId
	migrationhub-strategy:GetMessage
	migrationhub-strategy:GetPortfolioPreferences
	migrationhub-strategy:GetPortfolioSummary
	migrationhub-strategy:GetRecommendationReportDetails
	migrationhub-strategy:GetServerDetails
	migrationhub-strategy:GetServerStrategies
	migrationhub-strategy:ListAnalyzableServers
	migrationhub-strategy:ListAntiPatterns
	migrationhub-strategy:ListApplicationComponents
	migrationhub-strategy:ListCollectors
	migrationhub-strategy:ListImportFileTask
	migrationhub-strategy:ListJarArtifacts
	migrationhub-strategy:ListServers
	migrationhub-strategy:PutPortfolioPreferences
	migrationhub-strategy:RegisterCollector

サービスプレフィックス	アクション
	migrationhub-strategy:SendMessage
	migrationhub-strategy:StartAssessment
	migrationhub-strategy:StartImportFileTask
	migrationhub-strategy:StartRecommendationReportGeneration
	migrationhub-strategy:StopAssessment
	migrationhub-strategy:UpdateApplicationComponentConfig
	migrationhub-strategy:UpdateCollectorConfiguration
	migrationhub-strategy:UpdateServerConfig

サービスプレフィックス	アクション
mobiletargeting	mobiletargeting:CreateApp
	mobiletargeting:CreateCampaign
	mobiletargeting:CreateEmailTemplate
	mobiletargeting:CreateExportJob
	mobiletargeting:CreateImportJob
	mobiletargeting:CreateInAppTemplate
	mobiletargeting:CreateJourney
	mobiletargeting:CreatePushTemplate
	mobiletargeting:CreateRecommenderConfiguration
	mobiletargeting:CreateSegment
	mobiletargeting:CreateSmsTemplate
	mobiletargeting:CreateVoiceTemplate
	mobiletargeting>DeleteAdmChannel
	mobiletargeting>DeleteApnsChannel
	mobiletargeting>DeleteApnsSandboxChannel
	mobiletargeting>DeleteApnsVoipChannel
	mobiletargeting>DeleteApnsVoipSandboxChannel
	mobiletargeting>DeleteApp
	mobiletargeting>DeleteBaiduChannel
	mobiletargeting>DeleteCampaign
	mobiletargeting>DeleteEmailChannel

サービスプレフィックス	アクション
	mobiletargeting:DeleteEmailTemplate
	mobiletargeting:DeleteEndpoint
	mobiletargeting:DeleteEventStream
	mobiletargeting:DeleteGcmChannel
	mobiletargeting:DeleteInAppTemplate
	mobiletargeting:DeleteJourney
	mobiletargeting:DeletePushTemplate
	mobiletargeting:DeleteRecommenderConfiguration
	mobiletargeting:DeleteSegment
	mobiletargeting:DeleteSmsChannel
	mobiletargeting:DeleteSmsTemplate
	mobiletargeting:DeleteUserEndpoints
	mobiletargeting:DeleteVoiceChannel
	mobiletargeting:DeleteVoiceTemplate
	mobiletargeting:GetAdmChannel
	mobiletargeting:GetApnsChannel
	mobiletargeting:GetApnsSandboxChannel
	mobiletargeting:GetApnsVoipChannel
	mobiletargeting:GetApnsVoipSandboxChannel
	mobiletargeting:GetApp
	mobiletargeting:GetApplicationDateRangeKpi

サービスプレフィックス	アクション
	mobiletargeting:GetApplicationSettings
	mobiletargeting:GetApps
	mobiletargeting:GetBaiduChannel
	mobiletargeting:GetCampaign
	mobiletargeting:GetCampaignActivities
	mobiletargeting:GetCampaignDateRangeKpi
	mobiletargeting:GetCampaigns
	mobiletargeting:GetCampaignVersion
	mobiletargeting:GetCampaignVersions
	mobiletargeting:GetChannels
	mobiletargeting:GetEmailChannel
	mobiletargeting:GetEmailTemplate
	mobiletargeting:GetEndpoint
	mobiletargeting:GetEventStream
	mobiletargeting:GetExportJob
	mobiletargeting:GetExportJobs
	mobiletargeting:GetGcmChannel
	mobiletargeting:GetImportJob
	mobiletargeting:GetImportJobs
	mobiletargeting:GetInAppMessages
	mobiletargeting:GetInAppTemplate

サービスプレフィックス	アクション
	mobiletargeting:GetJourney
	mobiletargeting:GetJourneyDateRangeKpi
	mobiletargeting:GetJourneyExecutionActivityMetrics
	mobiletargeting:GetJourneyExecutionMetrics
	mobiletargeting:GetJourneyRunExecutionActivityMetrics
	mobiletargeting:GetJourneyRunExecutionMetrics
	mobiletargeting:GetJourneyRuns
	mobiletargeting:GetPushTemplate
	mobiletargeting:GetRecommenderConfiguration
	mobiletargeting:GetRecommenderConfigurations
	mobiletargeting:GetSegment
	mobiletargeting:GetSegmentExportJobs
	mobiletargeting:GetSegmentImportJobs
	mobiletargeting:GetSegments
	mobiletargeting:GetSegmentVersion
	mobiletargeting:GetSegmentVersions
	mobiletargeting:GetSmsChannel
	mobiletargeting:GetSmsTemplate
	mobiletargeting:GetUserEndpoints
	mobiletargeting:GetVoiceChannel
	mobiletargeting:GetVoiceTemplate

サービスプレフィックス	アクション
	mobiletargeting:ListJourneys
	mobiletargeting:ListTemplates
	mobiletargeting:ListTemplateVersions
	mobiletargeting:PhoneNumberValidate
	mobiletargeting:PutEventStream
	mobiletargeting:RemoveAttributes
	mobiletargeting:UpdateAdmChannel
	mobiletargeting:UpdateApnsChannel
	mobiletargeting:UpdateApnsSandboxChannel
	mobiletargeting:UpdateApnsVoipChannel
	mobiletargeting:UpdateApnsVoipSandboxChannel
	mobiletargeting:UpdateApplicationSettings
	mobiletargeting:UpdateBaiduChannel
	mobiletargeting:UpdateCampaign
	mobiletargeting:UpdateEmailChannel
	mobiletargeting:UpdateEmailTemplate
	mobiletargeting:UpdateEndpoint
	mobiletargeting:UpdateEndpointsBatch
	mobiletargeting:UpdateGcmChannel
	mobiletargeting:UpdateInAppTemplate
	mobiletargeting:UpdateJourney

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>mobiletargeting:UpdateJourneyState</li><li>mobiletargeting:UpdatePushTemplate</li><li>mobiletargeting:UpdateRecommenderConfiguration</li><li>mobiletargeting:UpdateSegment</li><li>mobiletargeting:UpdateSmsChannel</li><li>mobiletargeting:UpdateSmsTemplate</li><li>mobiletargeting:UpdateTemplateActiveVersion</li><li>mobiletargeting:UpdateVoiceChannel</li><li>mobiletargeting:UpdateVoiceTemplate</li><li>mobiletargeting:VerifyOTPMessage</li></ul>



サービスプレフィックス	アクション
mq	mq:CreateBroker
	mq:CreateConfiguration
	mq:CreateUser
	mq>DeleteBroker
	mq>DeleteUser
	mq:DescribeBroker
	mq:DescribeBrokerEngineTypes
	mq:DescribeBrokerInstanceOptions
	mq:DescribeConfiguration
	mq:DescribeConfigurationRevision
	mq:DescribeUser
	mq:ListBrokers
	mq:ListConfigurationRevisions
	mq:ListConfigurations
	mq:ListUsers
	mq:Promote
	mq:RebootBroker
	mq:UpdateBroker
	mq:UpdateConfiguration
	mq:UpdateUser

サービスプレフィックス	アクション
networkmanager	networkmanager:AcceptAttachment
	networkmanager:AssociateConnectPeer
	networkmanager:AssociateCustomerGateway
	networkmanager:AssociateLink
	networkmanager:AssociateTransitGatewayConnectPeer
	networkmanager:CreateConnectAttachment
	networkmanager:CreateConnection
	networkmanager:CreateConnectPeer
	networkmanager:CreateCoreNetwork
	networkmanager:CreateDevice
	networkmanager:CreateGlobalNetwork
	networkmanager:CreateLink
	networkmanager:CreateSite
	networkmanager:CreateSiteToSiteVpnAttachment
	networkmanager:CreateTransitGatewayPeering
	networkmanager:CreateTransitGatewayRouteTableAttachment
	networkmanager:CreateVpcAttachment
	networkmanager>DeleteAttachment
	networkmanager>DeleteConnection
	networkmanager>DeleteConnectPeer
networkmanager>DeleteCoreNetwork	

サービスプレフィックス	アクション
	networkmanager:DeleteCoreNetworkPolicyVersion
	networkmanager:DeleteDevice
	networkmanager:DeleteGlobalNetwork
	networkmanager:DeleteLink
	networkmanager:DeletePeering
	networkmanager:DeleteResourcePolicy
	networkmanager:DeleteSite
	networkmanager:DeregisterTransitGateway
	networkmanager:DescribeGlobalNetworks
	networkmanager:DisassociateConnectPeer
	networkmanager:DisassociateCustomerGateway
	networkmanager:DisassociateLink
	networkmanager:DisassociateTransitGatewayConnectPeer
	networkmanager:ExecuteCoreNetworkChangeSet
	networkmanager:GetConnectAttachment
	networkmanager:GetConnections
	networkmanager:GetConnectPeer
	networkmanager:GetConnectPeerAssociations
	networkmanager:GetCoreNetwork
	networkmanager:GetCoreNetworkChangeEvents
	networkmanager:GetCoreNetworkChangeSet

サービスプレフィックス	アクション
	<p>networkmanager:GetCoreNetworkPolicy</p> <p>networkmanager:GetCustomerGatewayAssociations</p> <p>networkmanager:GetDevices</p> <p>networkmanager:GetLinkAssociations</p> <p>networkmanager:GetLinks</p> <p>networkmanager:GetNetworkResourceCounts</p> <p>networkmanager:GetNetworkResourceRelationships</p> <p>networkmanager:GetNetworkResources</p> <p>networkmanager:GetNetworkRoutes</p> <p>networkmanager:GetNetworkTelemetry</p> <p>networkmanager:GetResourcePolicy</p> <p>networkmanager:GetRouteAnalysis</p> <p>networkmanager:GetSites</p> <p>networkmanager:GetSiteToSiteVpnAttachment</p> <p>networkmanager:GetTransitGatewayConnectPeerAssociations</p> <p>networkmanager:GetTransitGatewayPeering</p> <p>networkmanager:GetTransitGatewayRegistrations</p> <p>networkmanager:GetTransitGatewayRouteTableAttachment</p> <p>networkmanager:GetVpcAttachment</p> <p>networkmanager:ListAttachments</p> <p>networkmanager:ListConnectPeers</p>

サービスプレフィックス	アクション
	<p>networkmanager:ListCoreNetworkPolicyVersions</p> <p>networkmanager:ListCoreNetworks</p> <p>networkmanager:ListOrganizationServiceAccessStatus</p> <p>networkmanager:ListPeerings</p> <p>networkmanager:PutCoreNetworkPolicy</p> <p>networkmanager:PutResourcePolicy</p> <p>networkmanager:RegisterTransitGateway</p> <p>networkmanager:RejectAttachment</p> <p>networkmanager:RestoreCoreNetworkPolicyVersion</p> <p>networkmanager:StartOrganizationServiceAccessUpdate</p> <p>networkmanager:StartRouteAnalysis</p> <p>networkmanager:UpdateConnection</p> <p>networkmanager:UpdateCoreNetwork</p> <p>networkmanager:UpdateDevice</p> <p>networkmanager:UpdateGlobalNetwork</p> <p>networkmanager:UpdateLink</p> <p>networkmanager:UpdateNetworkResourceMetadata</p> <p>networkmanager:UpdateSite</p> <p>networkmanager:UpdateVpcAttachment</p>

サービスプレフィックス	アクション
nimble	nimble:AcceptEulas
	nimble:CreateLaunchProfile
	nimble:CreateStreamingImage
	nimble:CreateStreamingSession
	nimble:CreateStreamingSessionStream
	nimble:CreateStudio
	nimble:CreateStudioComponent
	nimble>DeleteLaunchProfile
	nimble>DeleteLaunchProfileMember
	nimble>DeleteStreamingImage
	nimble>DeleteStreamingSession
	nimble>DeleteStudio
	nimble>DeleteStudioComponent
	nimble>DeleteStudioMember
	nimble:GetEula
	nimble:GetLaunchProfileDetails
	nimble:GetStreamingImage
	nimble:GetStreamingSession
	nimble:GetStreamingSessionBackup
	nimble:GetStreamingSessionStream
nimble:GetStudio	

サービスプレフィックス	アクション
	<code>nimble:GetStudioComponent</code>
	<code>nimble:GetStudioMember</code>
	<code>nimble:ListEulas</code>
	<code>nimble:ListLaunchProfileMembers</code>
	<code>nimble:ListLaunchProfiles</code>
	<code>nimble:ListStreamingImages</code>
	<code>nimble:ListStreamingSessionBackups</code>
	<code>nimble:ListStreamingSessions</code>
	<code>nimble:ListStudioComponents</code>
	<code>nimble:ListStudioMembers</code>
	<code>nimble:ListStudios</code>
	<code>nimble:PutLaunchProfileMembers</code>
	<code>nimble:PutStudioMembers</code>
	<code>nimble:StartStreamingSession</code>
	<code>nimble:StartStudioSSOConfigurationRepair</code>
	<code>nimble:StopStreamingSession</code>
	<code>nimble:UpdateLaunchProfile</code>
	<code>nimble:UpdateLaunchProfileMember</code>
	<code>nimble:UpdateStreamingImage</code>
	<code>nimble:UpdateStudio</code>
	<code>nimble:UpdateStudioComponent</code>

サービスプレフィックス	アクション
omics	omics:AbortMultipartReadSetUpload
	omics:BatchDeleteReadSet
	omics:CancelAnnotationImportJob
	omics:CancelRun
	omics:CancelVariantImportJob
	omics:CompleteMultipartReadSetUpload
	omics:CreateAnnotationStore
	omics:CreateMultipartReadSetUpload
	omics:CreateReferenceStore
	omics:CreateRunGroup
	omics:CreateSequenceStore
	omics:CreateVariantStore
	omics:CreateWorkflow
	omics>DeleteAnnotationStore
	omics>DeleteReference
	omics>DeleteReferenceStore
	omics>DeleteRun
	omics>DeleteRunGroup
	omics>DeleteSequenceStore
	omics>DeleteVariantStore
	omics>DeleteWorkflow



サービスプレフィックス	アクション
	omics:GetAnnotationImportJob
	omics:GetAnnotationStore
	omics:GetReadSet
	omics:GetReadSetActivationJob
	omics:GetReadSetExportJob
	omics:GetReadSetImportJob
	omics:GetReadSetMetadata
	omics:GetReference
	omics:GetReferenceImportJob
	omics:GetReferenceMetadata
	omics:GetReferenceStore
	omics:GetRun
	omics:GetRunGroup
	omics:GetRunTask
	omics:GetSequenceStore
	omics:GetVariantImportJob
	omics:GetVariantStore
	omics:GetWorkflow
	omics>ListAnnotationImportJobs
	omics>ListAnnotationStores
	omics>ListMultipartReadSetUploads

サービスプレフィックス	アクション
	omics:ListReadSetActivationJobs
	omics:ListReadSetExportJobs
	omics:ListReadSetImportJobs
	omics:ListReadSets
	omics:ListReadSetUploadParts
	omics:ListReferenceImportJobs
	omics:ListReferences
	omics:ListReferenceStores
	omics:ListRunGroups
	omics:ListRuns
	omics:ListRunTasks
	omics:ListSequenceStores
	omics:ListVariantImportJobs
	omics:ListVariantStores
	omics:ListWorkflows
	omics:StartAnnotationImportJob
	omics:StartReadSetActivationJob
	omics:StartReadSetExportJob
	omics:StartReadSetImportJob
	omics:StartReferenceImportJob
	omics:StartRun

サービスプレフィックス	アクション
	omics:StartVariantImportJob
	omics:UpdateAnnotationStore
	omics:UpdateRunGroup
	omics:UpdateVariantStore
	omics:UpdateWorkflow
	omics:UploadReadSetPart

サービスプレフィックス	アクション
opsworks	opsworks:AssignInstance
	opsworks:AssignVolume
	opsworks:AssociateElasticIp
	opsworks:AttachElasticLoadBalancer
	opsworks:CloneStack
	opsworks:CreateApp
	opsworks:CreateDeployment
	opsworks:CreateInstance
	opsworks:CreateLayer
	opsworks:CreateStack
	opsworks:CreateUserProfile
	opsworks>DeleteApp
	opsworks>DeleteInstance
	opsworks>DeleteLayer
	opsworks>DeleteStack
	opsworks>DeleteUserProfile
	opsworks:DeregisterEcsCluster
	opsworks:DeregisterElasticIp
	opsworks:DeregisterInstance
	opsworks:DeregisterRdsDbInstance
	opsworks:DeregisterVolume

サービスプレフィックス	アクション
	opsworks:DescribeAgentVersions
	opsworks:DescribeApps
	opsworks:DescribeCommands
	opsworks:DescribeDeployments
	opsworks:DescribeEcsClusters
	opsworks:DescribeElasticIps
	opsworks:DescribeElasticLoadBalancers
	opsworks:DescribeInstances
	opsworks:DescribeLayers
	opsworks:DescribeLoadBasedAutoScaling
	opsworks:DescribeMyUserProfile
	opsworks:DescribeOperatingSystems
	opsworks:DescribePermissions
	opsworks:DescribeRaidArrays
	opsworks:DescribeRdsDbInstances
	opsworks:DescribeServiceErrors
	opsworks:DescribeStackProvisioningParameters
	opsworks:DescribeStacks
	opsworks:DescribeStackSummary
	opsworks:DescribeTimeBasedAutoScaling
	opsworks:DescribeUserProfiles

サービスプレフィックス	アクション
	opsworks:DescribeVolumes
	opsworks:DetachElasticLoadBalancer
	opsworks:DisassociateElasticIp
	opsworks:GetHostnameSuggestion
	opsworks:GrantAccess
	opsworks:RebootInstance
	opsworks:RegisterEcsCluster
	opsworks:RegisterElasticIp
	opsworks:RegisterInstance
	opsworks:RegisterRdsDbInstance
	opsworks:RegisterVolume
	opsworks:SetLoadBasedAutoScaling
	opsworks:SetPermission
	opsworks:SetTimeBasedAutoScaling
	opsworks:StartInstance
	opsworks:StartStack
	opsworks:StopInstance
	opsworks:StopStack
	opsworks:UnassignInstance
	opsworks:UnassignVolume
	opsworks:UpdateApp

サービスプレフィックス	アクション
	opsworks:UpdateElasticIp
	opsworks:UpdateInstance
	opsworks:UpdateLayer
	opsworks:UpdateMyUserProfile
	opsworks:UpdateRdsDbInstance
	opsworks:UpdateStack
	opsworks:UpdateUserProfile
	opsworks:UpdateVolume

サービスプレフィックス	アクション
opsworks-cm	opsworks-cm:AssociateNode
	opsworks-cm:CreateBackup
	opsworks-cm:CreateServer
	opsworks-cm>DeleteBackup
	opsworks-cm>DeleteServer
	opsworks-cm:DescribeAccountAttributes
	opsworks-cm:DescribeBackups
	opsworks-cm:DescribeEvents
	opsworks-cm:DescribeNodeAssociationStatus
	opsworks-cm:DescribeServers
	opsworks-cm:DisassociateNode
	opsworks-cm:ExportServerEngineAttribute
	opsworks-cm:RestoreServer
	opsworks-cm:StartMaintenance
	opsworks-cm:UpdateServer
	opsworks-cm:UpdateServerEngineAttributes



サービスプレフィックス	アクション
組織	organizations:AcceptHandshake
	organizations:AttachPolicy
	organizations:CancelHandshake
	organizations:CloseAccount
	organizations:CreateAccount
	organizations:CreateGovCloudAccount
	organizations:CreateOrganization
	organizations:CreateOrganizationalUnit
	organizations:CreatePolicy
	organizations:DeclineHandshake
	organizations>DeleteOrganization
	organizations>DeleteOrganizationalUnit
	organizations>DeletePolicy
	organizations>DeleteResourcePolicy
	organizations:DeregisterDelegatedAdministrator
	organizations:DescribeAccount
	organizations:DescribeCreateAccountStatus
	organizations:DescribeEffectivePolicy
	organizations:DescribeHandshake
	organizations:DescribeOrganization
	organizations:DescribeOrganizationalUnit

サービスプレフィックス	アクション
	organizations:DescribePolicy
	organizations:DescribeResourcePolicy
	organizations:DetachPolicy
	organizations:DisableAWSServiceAccess
	organizations:DisablePolicyType
	organizations:EnableAllFeatures
	organizations:EnableAWSServiceAccess
	organizations:EnablePolicyType
	organizations:InviteAccountToOrganization
	organizations:LeaveOrganization
	organizations:ListAccounts
	organizations:ListAccountsForParent
	organizations:ListAWSServiceAccessForOrganization
	organizations:ListChildren
	organizations:ListCreateAccountStatus
	organizations:ListDelegatedAdministrators
	organizations:ListDelegatedServicesForAccount
	organizations:ListHandshakesForAccount
	organizations:ListHandshakesForOrganization
	organizations:ListOrganizationalUnitsForParent
	organizations:ListParents

サービスプレフィックス	アクション
	<code>organizations:ListPolicies</code>
	<code>organizations:ListPoliciesForTarget</code>
	<code>organizations:ListRoots</code>
	<code>organizations:ListTargetsForPolicy</code>
	<code>organizations:MoveAccount</code>
	<code>organizations:PutResourcePolicy</code>
	<code>organizations:RegisterDelegatedAdministrator</code>
	<code>organizations:RemoveAccountFromOrganization</code>
	<code>organizations:UpdateOrganizationalUnit</code>
	<code>organizations:UpdatePolicy</code>

サービスプレフィックス	アクション
outposts	outposts:CancelCapacityTask
	outposts:CancelOrder
	outposts:CreateOrder
	outposts:CreateOutpost
	outposts:CreatePrivateConnectivityConfig
	outposts:CreateSite
	outposts>DeleteOutpost
	outposts>DeleteSite
	outposts:GetCapacityTask
	outposts:GetCatalogItem
	outposts:GetConnection
	outposts:GetOrder
	outposts:GetOutpost
	outposts:GetOutpostInstanceTypes
	outposts:GetOutpostSupportedInstanceTypes
	outposts:GetPrivateConnectivityConfig
	outposts:GetSite
	outposts:GetSiteAddress
	outposts:ListAssets
	outposts:ListCapacityTasks
	outposts:ListCatalogItems

サービスプレフィックス	アクション
	<ul style="list-style-type: none"><li>outposts:ListOrders</li><li>outposts:ListOutposts</li><li>outposts:ListSites</li><li>outposts:StartCapacityTask</li><li>outposts:StartConnection</li><li>outposts:UpdateOutpost</li><li>outposts:UpdateSite</li><li>outposts:UpdateSiteAddress</li><li>outposts:UpdateSiteRackPhysicalProperties</li></ul>

サービスプレフィックス	アクション
panorama	panorama:CreateApplicationInstance
	panorama:CreateJobForDevices
	panorama:CreateNodeFromTemplateJob
	panorama:CreatePackage
	panorama:CreatePackageImportJob
	panorama>DeleteDevice
	panorama>DeletePackage
	panorama:DeregisterPackageVersion
	panorama:DescribeApplicationInstance
	panorama:DescribeApplicationInstanceDetails
	panorama:DescribeDevice
	panorama:DescribeDeviceJob
	panorama:DescribeNode
	panorama:DescribeNodeFromTemplateJob
	panorama:DescribePackage
	panorama:DescribePackageImportJob
	panorama:DescribePackageVersion
	panorama:ListApplicationInstanceDependencies
	panorama:ListApplicationInstanceNodeInstances
	panorama:ListApplicationInstances
panorama:ListDevices	

サービスプレフィックス	アクション
	panorama:ListDevicesJobs panorama:ListNodeFromTemplateJobs panorama:ListNodes panorama:ListPackageImportJobs panorama:ListPackages panorama:ProvisionDevice panorama:RegisterPackageVersion panorama:RemoveApplicationInstance panorama:SignalApplicationInstanceNodeInstances panorama:UpdateDeviceMetadata
pi	pi:CreatePerformanceAnalysisReport pi>DeletePerformanceAnalysisReport pi:DescribeDimensionKeys pi:GetDimensionKeyDetails pi:GetPerformanceAnalysisReport pi:GetResourceMetadata pi:GetResourceMetrics pi>ListAvailableResourceDimensions pi>ListAvailableResourceMetrics pi>ListPerformanceAnalysisReports

サービスプレフィックス	アクション
pipes	pipes:CreatePipe
	pipes>DeletePipe
	pipes:DescribePipe
	pipes:ListPipes
	pipes:StartPipe
	pipes:StopPipe
	pipes:UpdatePipe
polly	polly>DeleteLexicon
	polly:DescribeVoices
	polly:GetLexicon
	polly:GetSpeechSynthesisTask
	polly:ListLexicons
	polly:ListSpeechSynthesisTasks
	polly:PutLexicon
	polly:StartSpeechSynthesisTask
	polly:SynthesizeSpeech



サービスプレフィックス	アクション
profile	profile:AddProfileKey
	profile:CreateCalculatedAttributeDefinition
	profile:CreateDomain
	profile:CreateEventStream
	profile:CreateProfile
	profile>DeleteCalculatedAttributeDefinition
	profile>DeleteDomain
	profile>DeleteEventStream
	profile>DeleteIntegration
	profile>DeleteProfile
	profile>DeleteProfileKey
	profile>DeleteProfileObject
	profile>DeleteProfileObjectType
	profile>DeleteWorkflow
	profile:DetectProfileObjectType
	profile:GetAutoMergingPreview
	profile:GetCalculatedAttributeDefinition
	profile:GetCalculatedAttributeForProfile
	profile:GetDomain
	profile:GetEventStream
	profile:GetIdentityResolutionJob

サービスプレフィックス	アクション
	profile:GetIntegration
	profile:GetMatches
	profile:GetProfileObjectType
	profile:GetProfileObjectTypeTemplate
	profile:GetSimilarProfiles
	profile:GetWorkflow
	profile:GetWorkflowSteps
	profile:ListAccountIntegrations
	profile:ListCalculatedAttributeDefinitions
	profile:ListCalculatedAttributesForProfile
	profile:ListDomains
	profile:ListEventStreams
	profile:ListIdentityResolutionJobs
	profile:ListIntegrations
	profile:ListProfileObjects
	profile:ListProfileObjectTypes
	profile:ListProfileObjectTypeTemplates
	profile:ListRuleBasedMatches
	profile:ListWorkflows
	profile:MergeProfiles
	profile:PutIntegration

サービスプレフィックス	アクション
	profile:PutProfileObject
	profile:PutProfileObjectType
	profile:SearchProfiles
	profile:UpdateCalculatedAttributeDefinition
	profile:UpdateDomain
	profile:UpdateProfile

サービスプレフィックス	アクション
qldb	qldb:CancelJournalKinesisStream
	qldb:CreateLedger
	qldb>DeleteLedger
	qldb:DescribeJournalKinesisStream
	qldb:DescribeJournalS3Export
	qldb:DescribeLedger
	qldb:ExportJournalToS3
	qldb:GetBlock
	qldb:GetDigest
	qldb:GetRevision
	qldb:ListJournalKinesisStreamsForLedger
	qldb:ListJournalS3Exports
	qldb:ListJournalS3ExportsForLedger
	qldb:ListLedgers
	qldb:StreamJournalToKinesis
	qldb:UpdateLedger
	qldb:UpdateLedgerPermissionsMode

サービスプレフィックス	アクション
ram	ram:AcceptResourceShareInvitation
	ram:AssociateResourceShare
	ram:AssociateResourceSharePermission
	ram:CreatePermission
	ram:CreatePermissionVersion
	ram:CreateResourceShare
	ram>DeletePermission
	ram>DeletePermissionVersion
	ram>DeleteResourceShare
	ram:DisassociateResourceShare
	ram:DisassociateResourceSharePermission
	ram:EnableSharingWithAwsOrganization
	ram:GetPermission
	ram:GetResourcePolicies
	ram:GetResourceShareAssociations
	ram:GetResourceShareInvitations
	ram:GetResourceShares
	ram:ListPendingInvitationResources
	ram:ListPermissionAssociations
	ram:ListPermissions
	ram:ListPermissionVersions

サービスプレフィックス	アクション
	ram:ListPrincipals
	ram:ListReplacePermissionAssociationsWork
	ram:ListResources
	ram:ListResourceSharePermissions
	ram:ListResourceTypes
	ram:PromotePermissionCreatedFromPolicy
	ram:PromoteResourceShareCreatedFromPolicy
	ram:RejectResourceShareInvitation
	ram:ReplacePermissionAssociations
	ram:SetDefaultPermissionVersion
	ram:UpdateResourceShare
rbin	rbin:CreateRule
	rbin>DeleteRule
	rbin:GetRule
	rbin:ListRules
	rbin:LockRule
	rbin:UnlockRule
	rbin:UpdateRule

サービスプレフィックス	アクション
rds	rds:AddRoleToDBCluster
	rds:AddRoleToDBInstance
	rds:AddSourceIdentifierToSubscription
	rds:ApplyPendingMaintenanceAction
	rds:AuthorizeDBSecurityGroupIngress
	rds:BacktrackDBCluster
	rds:CancelExportTask
	rds:CopyDBClusterParameterGroup
	rds:CopyDBClusterSnapshot
	rds:CopyDBParameterGroup
	rds:CopyDBSnapshot
	rds:CopyOptionGroup
	rds>CreateCustomDBEngineVersion
	rds>CreateDBClusterParameterGroup
	rds>CreateDBClusterSnapshot
	rds>CreateDBParameterGroup
	rds>CreateDBProxy
	rds>CreateDBProxyEndpoint
	rds>CreateDBSecurityGroup
	rds>CreateDBSnapshot
	rds>CreateDBSubnetGroup

サービスプレフィックス	アクション
	rds:CreateEventSubscription
	rds:CreateGlobalCluster
	rds:CreateOptionGroup
	rds>DeleteBlueGreenDeployment
	rds>DeleteDBClusterAutomatedBackup
	rds>DeleteDBClusterParameterGroup
	rds>DeleteDBClusterSnapshot
	rds>DeleteDBInstanceAutomatedBackup
	rds>DeleteDBParameterGroup
	rds>DeleteDBProxy
	rds>DeleteDBProxyEndpoint
	rds>DeleteDBSecurityGroup
	rds>DeleteDBSnapshot
	rds>DeleteDBSubnetGroup
	rds>DeleteEventSubscription
	rds>DeleteGlobalCluster
	rds>DeleteOptionGroup
	rds:DeregisterDBProxyTargets
	rds:DescribeAccountAttributes
	rds:DescribeBlueGreenDeployments
	rds:DescribeCertificates



サービスプレフィックス	アクション
	rds:DescribeDBClusterAutomatedBackups
	rds:DescribeDBClusterBacktracks
	rds:DescribeDBClusterEndpoints
	rds:DescribeDBClusterParameterGroups
	rds:DescribeDBClusterParameters
	rds:DescribeDBClusters
	rds:DescribeDBClusterSnapshotAttributes
	rds:DescribeDBClusterSnapshots
	rds:DescribeDBEngineVersions
	rds:DescribeDBInstanceAutomatedBackups
	rds:DescribeDBInstances
	rds:DescribeDBLogFiles
	rds:DescribeDBParameterGroups
	rds:DescribeDBParameters
	rds:DescribeDBProxies
	rds:DescribeDBProxyEndpoints
	rds:DescribeDBProxyTargetGroups
	rds:DescribeDBProxyTargets
	rds:DescribeDBRecommendations
	rds:DescribeDBSecurityGroups
	rds:DescribeDBSnapshotAttributes

サービスプレフィックス	アクション
	rds:DescribeDBSnapshots
	rds:DescribeDbSnapshotTenantDatabases
	rds:DescribeDBSubnetGroups
	rds:DescribeEngineDefaultClusterParameters
	rds:DescribeEngineDefaultParameters
	rds:DescribeEventCategories
	rds:DescribeEvents
	rds:DescribeEventSubscriptions
	rds:DescribeExportTasks
	rds:DescribeGlobalClusters
	rds:DescribeIntegrations
	rds:DescribeOptionGroupOptions
	rds:DescribeOptionGroups
	rds:DescribeOrderableDBInstanceOptions
	rds:DescribePendingMaintenanceActions
	rds:DescribeReservedDBInstances
	rds:DescribeReservedDBInstancesOfferings
	rds:DescribeSourceRegions
	rds:DescribeTenantDatabases
	rds:DescribeValidDBInstanceModifications
	rds:DownloadCompleteDBLogFile

サービスプレフィックス	アクション
	rds:DownloadDBLogFilePortion
	rds:FailoverDBCluster
	rds:FailoverGlobalCluster
	rds:ModifyActivityStream
	rds:ModifyCertificates
	rds:ModifyCurrentDBClusterCapacity
	rds:ModifyDBClusterEndpoint
	rds:ModifyDBClusterParameterGroup
	rds:ModifyDBClusterSnapshotAttribute
	rds:ModifyDBParameterGroup
	rds:ModifyDBProxy
	rds:ModifyDBProxyEndpoint
	rds:ModifyDBProxyTargetGroup
	rds:ModifyDBRecommendation
	rds:ModifyDBSnapshot
	rds:ModifyDBSnapshotAttribute
	rds:ModifyDBSubnetGroup
	rds:ModifyEventSubscription
	rds:ModifyGlobalCluster
	rds:ModifyOptionGroup
	rds:ModifyTenantDatabase

サービスプレフィックス	アクション
	rds:PurchaseReservedDBInstancesOffering
	rds:RebootDBCluster
	rds:RegisterDBProxyTargets
	rds:RemoveFromGlobalCluster
	rds:RemoveRoleFromDBCluster
	rds:RemoveRoleFromDBInstance
	rds:RemoveSourceIdentifierFromSubscription
	rds:ResetDBClusterParameterGroup
	rds:ResetDBParameterGroup
	rds:RestoreDBClusterFromS3
	rds:RestoreDBClusterFromSnapshot
	rds:RestoreDBClusterToPointInTime
	rds:RestoreDBInstanceFromDBSnapshot
	rds:RestoreDBInstanceFromS3
	rds:RestoreDBInstanceToPointInTime
	rds:RevokeDBSecurityGroupIngress
	rds:StartActivityStream
	rds:StartDBCluster
	rds:StartDBInstance
	rds:StartDBInstanceAutomatedBackupsReplication
	rds:StartExportTask

サービスプレフィックス	アクション
	rds:StopActivityStream
	rds:StopDBCluster
	rds:StopDBInstance
	rds:StopDBInstanceAutomatedBackupsReplication
	rds:SwitchoverBlueGreenDeployment
	rds:SwitchoverGlobalCluster
	rds:SwitchoverReadReplica

サービスプレフィックス	アクション
redshift	redshift:AcceptReservedNodeExchange
	redshift:AddPartner
	redshift:AssociateDataShareConsumer
	redshift:AuthorizeClusterSecurityGroupIngress
	redshift:AuthorizeDataShare
	redshift:AuthorizeEndpointAccess
	redshift:AuthorizeSnapshotAccess
	redshift:BatchDeleteClusterSnapshots
	redshift:BatchModifyClusterSnapshots
	redshift:CancelResize
	redshift:CopyClusterSnapshot
	redshift>CreateAuthenticationProfile
	redshift>CreateCluster
	redshift>CreateClusterParameterGroup
	redshift>CreateClusterSecurityGroup
	redshift>CreateClusterSnapshot
	redshift>CreateClusterSubnetGroup
	redshift>CreateCustomDomainAssociation
	redshift>CreateEndpointAccess
	redshift>CreateEventSubscription
	redshift>CreateHsmClientCertificate

サービスプレフィックス	アクション
	redshift:CreateHsmConfiguration
	redshift:CreateRedshiftIdcApplication
	redshift:CreateScheduledAction
	redshift:CreateSnapshotCopyGrant
	redshift:CreateSnapshotSchedule
	redshift:CreateUsageLimit
	redshift:DeauthorizeDataShare
	redshift>DeleteAuthenticationProfile
	redshift>DeleteCluster
	redshift>DeleteClusterParameterGroup
	redshift>DeleteClusterSecurityGroup
	redshift>DeleteClusterSnapshot
	redshift>DeleteClusterSubnetGroup
	redshift>DeleteCustomDomainAssociation
	redshift>DeleteEndpointAccess
	redshift>DeleteEventSubscription
	redshift>DeleteHsmClientCertificate
	redshift>DeleteHsmConfiguration
	redshift>DeletePartner
	redshift>DeleteScheduledAction
	redshift>DeleteSnapshotCopyGrant

サービスプレフィックス	アクション
	redshift:DeleteSnapshotSchedule
	redshift:DeleteUsageLimit
	redshift:DescribeAccountAttributes
	redshift:DescribeAuthenticationProfiles
	redshift:DescribeClusterDbRevisions
	redshift:DescribeClusterParameterGroups
	redshift:DescribeClusterParameters
	redshift:DescribeClusters
	redshift:DescribeClusterSecurityGroups
	redshift:DescribeClusterSnapshots
	redshift:DescribeClusterSubnetGroups
	redshift:DescribeClusterTracks
	redshift:DescribeClusterVersions
	redshift:DescribeCustomDomainAssociations
	redshift:DescribeDataShares
	redshift:DescribeDataSharesForConsumer
	redshift:DescribeDataSharesForProducer
	redshift:DescribeDefaultClusterParameters
	redshift:DescribeEndpointAccess
	redshift:DescribeEndpointAuthorization
	redshift:DescribeEventCategories



サービスプレフィックス	アクション
	redshift:DescribeEvents
	redshift:DescribeEventSubscriptions
	redshift:DescribeHsmClientCertificates
	redshift:DescribeHsmConfigurations
	redshift:DescribeInboundIntegrations
	redshift:DescribeLoggingStatus
	redshift:DescribeNodeConfigurationOptions
	redshift:DescribeOrderableClusterOptions
	redshift:DescribePartners
	redshift:DescribeRedshiftIadcApplications
	redshift:DescribeReservedNodeExchangeStatus
	redshift:DescribeReservedNodeOfferings
	redshift:DescribeReservedNodes
	redshift:DescribeResize
	redshift:DescribeScheduledActions
	redshift:DescribeSnapshotCopyGrants
	redshift:DescribeSnapshotSchedules
	redshift:DescribeStorage
	redshift:DescribeTableRestoreStatus
	redshift:DescribeUsageLimits
	redshift:DisableLogging

サービスプレフィックス	アクション
	<p>redshift:DisableSnapshotCopy</p> <p>redshift:DisassociateDataShareConsumer</p> <p>redshift:EnableLogging</p> <p>redshift:EnableSnapshotCopy</p> <p>redshift:FailoverPrimaryCompute</p> <p>redshift:GetClusterCredentials</p> <p>redshift:GetClusterCredentialsWithIAM</p> <p>redshift:GetReservedNodeExchangeConfigurationOptions</p> <p>redshift:GetReservedNodeExchangeOfferings</p> <p>redshift:ListRecommendations</p> <p>redshift:ModifyAquaConfiguration</p> <p>redshift:ModifyAuthenticationProfile</p> <p>redshift:ModifyCluster</p> <p>redshift:ModifyClusterDbRevision</p> <p>redshift:ModifyClusterIamRoles</p> <p>redshift:ModifyClusterMaintenance</p> <p>redshift:ModifyClusterParameterGroup</p> <p>redshift:ModifyClusterSnapshot</p> <p>redshift:ModifyClusterSnapshotSchedule</p> <p>redshift:ModifyClusterSubnetGroup</p> <p>redshift:ModifyCustomDomainAssociation</p>

サービスプレフィックス	アクション
	redshift:ModifyEndpointAccess
	redshift:ModifyEventSubscription
	redshift:ModifyScheduledAction
	redshift:ModifySnapshotCopyRetentionPeriod
	redshift:ModifySnapshotSchedule
	redshift:ModifyUsageLimit
	redshift:PauseCluster
	redshift:PurchaseReservedNodeOffering
	redshift:RebootCluster
	redshift:RejectDataShare
	redshift:ResetClusterParameterGroup
	redshift:ResizeCluster
	redshift:RestoreFromClusterSnapshot
	redshift:RestoreTableFromClusterSnapshot
	redshift:ResumeCluster
	redshift:RevokeClusterSecurityGroupIngress
	redshift:RevokeEndpointAccess
	redshift:RevokeSnapshotAccess
	redshift:RotateEncryptionKey
	redshift:UpdatePartnerStatus

サービスプレフィックス	アクション
redshift-data	redshift-data:BatchExecuteStatement
	redshift-data:CancelStatement
	redshift-data:DescribeStatement
	redshift-data:DescribeTable
	redshift-data:ExecuteStatement
	redshift-data:GetStatementResult
	redshift-data:ListDatabases
	redshift-data:ListSchemas
	redshift-data:ListStatements
	redshift-data:ListTables

サービスプレフィックス	アクション
refactor-spaces	refactor-spaces:CreateApplication
	refactor-spaces:CreateEnvironment
	refactor-spaces:CreateRoute
	refactor-spaces:CreateService
	refactor-spaces>DeleteApplication
	refactor-spaces>DeleteEnvironment
	refactor-spaces>DeleteResourcePolicy
	refactor-spaces>DeleteRoute
	refactor-spaces>DeleteService
	refactor-spaces:GetApplication
	refactor-spaces:GetEnvironment
	refactor-spaces:GetResourcePolicy
	refactor-spaces:GetRoute
	refactor-spaces:GetService
	refactor-spaces:ListApplications
	refactor-spaces:ListEnvironments
	refactor-spaces:ListEnvironmentVpcs
	refactor-spaces:ListRoutes
	refactor-spaces:ListServices
	refactor-spaces:PutResourcePolicy
refactor-spaces:UpdateRoute	

サービスプレフィックス	アクション
rekognition	rekognition:AssociateFaces
	rekognition:CompareFaces
	rekognition:CopyProjectVersion
	rekognition:CreateCollection
	rekognition:CreateDataset
	rekognition:CreateFaceLivenessSession
	rekognition:CreateProject
	rekognition:CreateProjectVersion
	rekognition:CreateStreamProcessor
	rekognition:CreateUser
	rekognition>DeleteCollection
	rekognition>DeleteDataset
	rekognition>DeleteFaces
	rekognition>DeleteProject
	rekognition>DeleteProjectPolicy
	rekognition>DeleteProjectVersion
	rekognition>DeleteStreamProcessor
	rekognition>DeleteUser
	rekognition:DescribeCollection
	rekognition:DescribeDataset
	rekognition:DescribeProjects

サービスプレフィックス	アクション
	rekognition:DescribeProjectVersions
	rekognition:DescribeStreamProcessor
	rekognition:DetectCustomLabels
	rekognition:DetectFaces
	rekognition:DetectLabels
	rekognition:DetectModerationLabels
	rekognition:DetectProtectiveEquipment
	rekognition:DetectText
	rekognition:DisassociateFaces
	rekognition:DistributeDatasetEntries
	rekognition:GetCelebrityInfo
	rekognition:GetCelebrityRecognition
	rekognition:GetContentModeration
	rekognition:GetFaceDetection
	rekognition:GetFaceLivenessSessionResults
	rekognition:GetFaceSearch
	rekognition:GetLabelDetection
	rekognition:GetMediaAnalysisJob
	rekognition:GetPersonTracking
	rekognition:GetSegmentDetection
	rekognition:GetTextDetection

サービスプレフィックス	アクション
	rekognition:IndexFaces
	rekognition:ListCollections
	rekognition:ListDatasetEntries
	rekognition:ListDatasetLabels
	rekognition:ListFaces
	rekognition:ListMediaAnalysisJobs
	rekognition:ListProjectPolicies
	rekognition:ListStreamProcessors
	rekognition:ListUsers
	rekognition:PutProjectPolicy
	rekognition:RecognizeCelebrities
	rekognition:SearchFaces
	rekognition:SearchFacesByImage
	rekognition:SearchUsers
	rekognition:SearchUsersByImage
	rekognition:StartCelebrityRecognition
	rekognition:StartContentModeration
	rekognition:StartFaceDetection
	rekognition:StartFaceLivenessSession
	rekognition:StartFaceSearch
	rekognition:StartLabelDetection



サービスプレフィックス	アクション
	rekognition:StartMediaAnalysisJob
	rekognition:StartPersonTracking
	rekognition:StartProjectVersion
	rekognition:StartSegmentDetection
	rekognition:StartStreamProcessor
	rekognition:StartTextDetection
	rekognition:StopProjectVersion
	rekognition:StopStreamProcessor
	rekognition:UpdateDatasetEntries
	rekognition:UpdateStreamProcessor

サービスプレフィックス	アクション
resiliencehub	resiliencehub:AddDraftAppVersionResourceMappings
	resiliencehub:CreateApp
	resiliencehub:CreateAppVersionAppComponent
	resiliencehub:CreateAppVersionResource
	resiliencehub:CreateRecommendationTemplate
	resiliencehub:CreateResiliencyPolicy
	resiliencehub>DeleteApp
	resiliencehub>DeleteAppAssessment
	resiliencehub>DeleteAppInputSource
	resiliencehub>DeleteAppVersionAppComponent
	resiliencehub>DeleteAppVersionResource
	resiliencehub>DeleteRecommendationTemplate
	resiliencehub>DeleteResiliencyPolicy
	resiliencehub:DescribeApp
	resiliencehub:DescribeAppAssessment
	resiliencehub:DescribeAppVersion
	resiliencehub:DescribeAppVersionAppComponent
	resiliencehub:DescribeAppVersionResource
	resiliencehub:DescribeAppVersionResourcesResolutionStatus
	resiliencehub:DescribeAppVersionTemplate
	resiliencehub:DescribeDraftAppVersionResourcesImportStatus

サービスプレフィックス	アクション
	resiliencehub:DescribeResiliencyPolicy
	resiliencehub:ImportResourcesToDraftAppVersion
	resiliencehub:ListAlarmRecommendations
	resiliencehub:ListAppAssessments
	resiliencehub:ListAppComponentCompliances
	resiliencehub:ListAppComponentRecommendations
	resiliencehub:ListAppInputSources
	resiliencehub:ListApps
	resiliencehub:ListAppVersionAppComponent
	resiliencehub:ListAppVersionResourceMappings
	resiliencehub:ListAppVersionResources
	resiliencehub:ListAppVersions
	resiliencehub:ListRecommendationTemplates
	resiliencehub:ListResiliencyPolicies
	resiliencehub:ListSopRecommendations
	resiliencehub:ListSuggestedResiliencyPolicies
	resiliencehub:ListTestRecommendations
	resiliencehub:ListUnsupportedAppVersionResources
	resiliencehub:PublishAppVersion
	resiliencehub:PutDraftAppVersionTemplate
	resiliencehub:RemoveDraftAppVersionResourceMappings

サービスプレフィックス	アクション
	resiliencehub:ResolveAppVersionResources
	resiliencehub:StartAppAssessment
	resiliencehub:UpdateApp
	resiliencehub:UpdateAppVersion
	resiliencehub:UpdateAppVersionAppComponent
	resiliencehub:UpdateAppVersionResource
	resiliencehub:UpdateResiliencyPolicy

サービスプレフィックス	アクション
resource-explorer-2	resource-explorer-2:AssociateDefaultView
	resource-explorer-2:BatchGetView
	resource-explorer-2:CreateIndex
	resource-explorer-2:CreateView
	resource-explorer-2:DeleteIndex
	resource-explorer-2>DeleteView
	resource-explorer-2:DisassociateDefaultView
	resource-explorer-2:GetAccountLevelServiceConfiguration
	resource-explorer-2:GetDefaultView
	resource-explorer-2:GetIndex
	resource-explorer-2:ListIndexes
	resource-explorer-2:ListIndexesForMembers
	resource-explorer-2:ListSupportedResourceTypes
	resource-explorer-2:ListViews
	resource-explorer-2:Search
	resource-explorer-2:UpdateIndexType
	resource-explorer-2:UpdateView

サービスプレフィックス	アクション
resource-groups	resource-groups:CreateGroup
	resource-groups>DeleteGroup
	resource-groups:GetAccountSettings
	resource-groups:GetGroup
	resource-groups:GetGroupConfiguration
	resource-groups:GetGroupQuery
	resource-groups:GroupResources
	resource-groups:ListGroupResources
	resource-groups:ListGroups
	resource-groups:PutGroupConfiguration
	resource-groups:SearchResources
	resource-groups:UngroupResources
	resource-groups:UpdateAccountSettings
	resource-groups:UpdateGroup
	resource-groups:UpdateGroupQuery

サービスプレフィックス	アクション
robomaker	robomaker:BatchDeleteWorlds
	robomaker:BatchDescribeSimulationJob
	robomaker:CancelDeploymentJob
	robomaker:CancelSimulationJob
	robomaker:CancelSimulationJobBatch
	robomaker:CancelWorldExportJob
	robomaker:CancelWorldGenerationJob
	robomaker:CreateDeploymentJob
	robomaker:CreateFleet
	robomaker:CreateRobot
	robomaker:CreateRobotApplication
	robomaker:CreateRobotApplicationVersion
	robomaker:CreateSimulationApplication
	robomaker:CreateSimulationApplicationVersion
	robomaker:CreateSimulationJob
	robomaker:CreateWorldExportJob
	robomaker:CreateWorldGenerationJob
	robomaker:CreateWorldTemplate
	robomaker>DeleteFleet
	robomaker>DeleteRobot
	robomaker>DeleteRobotApplication

サービスプレフィックス	アクション
	robomaker:DeleteSimulationApplication
	robomaker:DeleteWorldTemplate
	robomaker:DeregisterRobot
	robomaker:DescribeDeploymentJob
	robomaker:DescribeFleet
	robomaker:DescribeRobot
	robomaker:DescribeRobotApplication
	robomaker:DescribeSimulationApplication
	robomaker:DescribeSimulationJob
	robomaker:DescribeSimulationJobBatch
	robomaker:DescribeWorld
	robomaker:DescribeWorldExportJob
	robomaker:DescribeWorldGenerationJob
	robomaker:DescribeWorldTemplate
	robomaker:GetWorldTemplateBody
	robomaker:ListDeploymentJobs
	robomaker:ListFleets
	robomaker:ListRobotApplications
	robomaker:ListRobots
	robomaker:ListSimulationApplications
	robomaker:ListSimulationJobBatches



サービスプレフィックス	アクション
	robomaker:ListSimulationJobs
	robomaker:ListWorldExportJobs
	robomaker:ListWorldGenerationJobs
	robomaker:ListWorlds
	robomaker:ListWorldTemplates
	robomaker:RegisterRobot
	robomaker:RestartSimulationJob
	robomaker:StartSimulationJobBatch
	robomaker:SyncDeploymentJob
	robomaker:UpdateRobotApplication
	robomaker:UpdateSimulationApplication
	robomaker:UpdateWorldTemplate

サービスプレフィックス	アクション
rolesanywhere	rolesanywhere:CreateProfile
	rolesanywhere:CreateTrustAnchor
	rolesanywhere>DeleteAttributeMapping
	rolesanywhere>DeleteCrl
	rolesanywhere>DeleteProfile
	rolesanywhere>DeleteTrustAnchor
	rolesanywhere:DisableCrl
	rolesanywhere:DisableProfile
	rolesanywhere:DisableTrustAnchor
	rolesanywhere:EnableCrl
	rolesanywhere:EnableProfile
	rolesanywhere:EnableTrustAnchor
	rolesanywhere:GetCrl
	rolesanywhere:GetProfile
	rolesanywhere:GetSubject
	rolesanywhere:GetTrustAnchor
	rolesanywhere:ImportCrl
	rolesanywhere:ListCrls
	rolesanywhere:ListProfiles
	rolesanywhere:ListSubjects
rolesanywhere:ListTrustAnchors	

サービスプレフィックス	アクション
	rolesanywhere:PutAttributeMapping rolesanywhere:PutNotificationSettings rolesanywhere:ResetNotificationSettings rolesanywhere:UpdateCrl rolesanywhere:UpdateProfile rolesanywhere:UpdateTrustAnchor

サービスプレフィックス	アクション
route53	route53:ActivateKeySigningKey
	route53:AssociateVPCWithHostedZone
	route53:ChangeCidrCollection
	route53:ChangeResourceRecordSets
	route53:CreateCidrCollection
	route53:CreateHealthCheck
	route53:CreateHostedZone
	route53:CreateKeySigningKey
	route53:CreateQueryLoggingConfig
	route53:CreateReusableDelegationSet
	route53:CreateTrafficPolicy
	route53:CreateTrafficPolicyInstance
	route53:CreateTrafficPolicyVersion
	route53:CreateVPCAssociationAuthorization
	route53:DeactivateKeySigningKey
	route53>DeleteCidrCollection
	route53>DeleteHealthCheck
	route53>DeleteHostedZone
	route53>DeleteKeySigningKey
	route53>DeleteQueryLoggingConfig
	route53>DeleteReusableDelegationSet

サービスプレフィックス	アクション
	route53:DeleteTrafficPolicy
	route53:DeleteTrafficPolicyInstance
	route53:DeleteVPCAssociationAuthorization
	route53:DisableHostedZoneDNSSEC
	route53:DisassociateVPCFromHostedZone
	route53:EnableHostedZoneDNSSEC
	route53:GetAccountLimit
	route53:GetChange
	route53:GetCheckerIpRanges
	route53:GetDNSSEC
	route53:GetGeoLocation
	route53:GetHealthCheck
	route53:GetHealthCheckCount
	route53:GetHealthCheckLastFailureReason
	route53:GetHealthCheckStatus
	route53:GetHostedZone
	route53:GetHostedZoneCount
	route53:GetHostedZoneLimit
	route53:GetQueryLoggingConfig
	route53:GetReusableDelegationSet
	route53:GetReusableDelegationSetLimit

サービスプレフィックス	アクション
	route53:GetTrafficPolicy
	route53:GetTrafficPolicyInstance
	route53:GetTrafficPolicyInstanceCount
	route53:ListCidrBlocks
	route53:ListCidrCollections
	route53:ListCidrLocations
	route53:ListGeoLocations
	route53:ListHealthChecks
	route53:ListHostedZones
	route53:ListHostedZonesByName
	route53:ListHostedZonesByVPC
	route53:ListQueryLoggingConfigs
	route53:ListResourceRecordSets
	route53:ListReusableDelegationSets
	route53:ListTrafficPolicies
	route53:ListTrafficPolicyInstances
	route53:ListTrafficPolicyInstancesByHostedZone
	route53:ListTrafficPolicyInstancesByPolicy
	route53:ListTrafficPolicyVersions
	route53:ListVPCAssociationAuthorizations
	route53:TestDNSAnswer

サービスプレフィックス	アクション
	route53:UpdateHealthCheck
	route53:UpdateHostedZoneComment
	route53:UpdateTrafficPolicyComment
	route53:UpdateTrafficPolicyInstance

サービスプレフィックス	アクション
route53-recovery-control-config	route53-recovery-control-config:CreateCluster
	route53-recovery-control-config:CreateControlPanel
	route53-recovery-control-config:CreateRoutingControl
	route53-recovery-control-config:CreateSafetyRule
	route53-recovery-control-config>DeleteCluster
	route53-recovery-control-config>DeleteControlPanel
	route53-recovery-control-config>DeleteRoutingControl
	route53-recovery-control-config>DeleteSafetyRule
	route53-recovery-control-config:DescribeCluster
	route53-recovery-control-config:DescribeControlPanel
	route53-recovery-control-config:DescribeRoutingControl
	route53-recovery-control-config:DescribeSafetyRule
	route53-recovery-control-config:GetResourcePolicy
	route53-recovery-control-config>ListAssociatedRoute53HealthChecks
	route53-recovery-control-config>ListClusters
	route53-recovery-control-config>ListControlPanels
	route53-recovery-control-config>ListRoutingControls
	route53-recovery-control-config>ListSafetyRules
	route53-recovery-control-config:UpdateControlPanel
	route53-recovery-control-config:UpdateRoutingControl



サービスプレフィックス	アクション
	route53-recovery-control-config:UpdateSafetyRule

サービスプレフィックス	アクション
route53-recovery-readiness	route53-recovery-readiness:CreateCell
	route53-recovery-readiness:CreateCrossAccountAuthorization
	route53-recovery-readiness:CreateReadinessCheck
	route53-recovery-readiness:CreateRecoveryGroup
	route53-recovery-readiness:CreateResourceSet
	route53-recovery-readiness>DeleteCell
	route53-recovery-readiness>DeleteCrossAccountAuthorization
	route53-recovery-readiness>DeleteReadinessCheck
	route53-recovery-readiness>DeleteRecoveryGroup
	route53-recovery-readiness>DeleteResourceSet
	route53-recovery-readiness:GetArchitectureRecommendations
	route53-recovery-readiness:GetCell
	route53-recovery-readiness:GetCellReadinessSummary
	route53-recovery-readiness:GetReadinessCheck
	route53-recovery-readiness:GetReadinessCheckResourceStatus
	route53-recovery-readiness:GetReadinessCheckStatus
	route53-recovery-readiness:GetRecoveryGroup
	route53-recovery-readiness:GetRecoveryGroupReadinessSummary
	route53-recovery-readiness:GetResourceSet
	route53-recovery-readiness:ListCells
route53-recovery-readiness:ListCrossAccountAuthorizations	

サービスプレフィックス	アクション
	route53-recovery-readiness:ListReadinessChecks
	route53-recovery-readiness:ListRecoveryGroups
	route53-recovery-readiness:ListResourceSets
	route53-recovery-readiness:ListRules
	route53-recovery-readiness:UpdateCell
	route53-recovery-readiness:UpdateReadinessCheck
	route53-recovery-readiness:UpdateRecoveryGroup
	route53-recovery-readiness:UpdateResourceSet

サービスプレフィックス	アクション
route53resolver	route53resolver:AssociateFirewallRuleGroup
	route53resolver:AssociateResolverEndpointIpAddress
	route53resolver:AssociateResolverQueryLogConfig
	route53resolver:AssociateResolverRule
	route53resolver:CreateFirewallDomainList
	route53resolver:CreateFirewallRule
	route53resolver:CreateFirewallRuleGroup
	route53resolver:CreateResolverEndpoint
	route53resolver:CreateResolverQueryLogConfig
	route53resolver:CreateResolverRule
	route53resolver>DeleteFirewallDomainList
	route53resolver>DeleteFirewallRule
	route53resolver>DeleteFirewallRuleGroup
	route53resolver>DeleteOutpostResolver
	route53resolver>DeleteResolverEndpoint
	route53resolver>DeleteResolverQueryLogConfig
	route53resolver>DeleteResolverRule
	route53resolver:DisassociateFirewallRuleGroup
	route53resolver:DisassociateResolverEndpointIpAddress
	route53resolver:DisassociateResolverQueryLogConfig
	route53resolver:DisassociateResolverRule

サービスプレフィックス	アクション
	route53resolver:GetFirewallConfig
	route53resolver:GetFirewallDomainList
	route53resolver:GetFirewallRuleGroup
	route53resolver:GetFirewallRuleGroupAssociation
	route53resolver:GetFirewallRuleGroupPolicy
	route53resolver:GetOutpostResolver
	route53resolver:GetResolverConfig
	route53resolver:GetResolverDnssecConfig
	route53resolver:GetResolverEndpoint
	route53resolver:GetResolverQueryLogConfig
	route53resolver:GetResolverQueryLogConfigAssociation
	route53resolver:GetResolverQueryLogConfigPolicy
	route53resolver:GetResolverRule
	route53resolver:GetResolverRuleAssociation
	route53resolver:GetResolverRulePolicy
	route53resolver:ImportFirewallDomains
	route53resolver:ListFirewallConfigs
	route53resolver:ListFirewallDomainLists
	route53resolver:ListFirewallDomains
	route53resolver:ListFirewallRuleGroupAssociations
	route53resolver:ListFirewallRuleGroups

サービスプレフィックス	アクション
	route53resolver:ListFirewallRules
	route53resolver:ListOutpostResolvers
	route53resolver:ListResolverConfigs
	route53resolver:ListResolverDnssecConfigs
	route53resolver:ListResolverEndpointIpAddresses
	route53resolver:ListResolverEndpoints
	route53resolver:ListResolverQueryLogConfigAssociations
	route53resolver:ListResolverQueryLogConfigs
	route53resolver:ListResolverRuleAssociations
	route53resolver:ListResolverRules
	route53resolver:PutFirewallRuleGroupPolicy
	route53resolver:PutResolverQueryLogConfigPolicy
	route53resolver:UpdateFirewallConfig
	route53resolver:UpdateFirewallDomains
	route53resolver:UpdateFirewallRule
	route53resolver:UpdateFirewallRuleGroupAssociation
	route53resolver:UpdateOutpostResolver
	route53resolver:UpdateResolverConfig
	route53resolver:UpdateResolverDnssecConfig
	route53resolver:UpdateResolverEndpoint
	route53resolver:UpdateResolverRule

サービスプレフィックス	アクション
rum	rum:BatchCreateRumMetricDefinitions
	rum:BatchDeleteRumMetricDefinitions
	rum:BatchGetRumMetricDefinitions
	rum:CreateAppMonitor
	rum>DeleteAppMonitor
	rum>DeleteRumMetricsDestination
	rum:GetAppMonitor
	rum:GetAppMonitorData
	rum:ListAppMonitors
	rum:ListRumMetricsDestinations
	rum:PutRumMetricsDestination
	rum:UpdateAppMonitor
	rum:UpdateRumMetricDefinition

サービスプレフィックス	アクション
s3	s3:AssociateAccessGrantsIdentityCenter
	s3:CreateAccessGrant
	s3:CreateAccessGrantsInstance
	s3:CreateAccessGrantsLocation
	s3:CreateAccessPoint
	s3:CreateAccessPointForObjectLambda
	s3:CreateBucket
	s3:CreateJob
	s3:CreateMultiRegionAccessPoint
	s3>DeleteAccessGrant
	s3>DeleteAccessGrantsInstance
	s3>DeleteAccessGrantsInstanceResourcePolicy
	s3>DeleteAccessGrantsLocation
	s3>DeleteAccessPoint
	s3>DeleteAccessPointForObjectLambda
	s3>DeleteAccessPointPolicy
	s3>DeleteAccessPointPolicyForObjectLambda
	s3:PutAccountPublicAccessBlock
	s3>DeleteBucket
	s3:PutAnalyticsConfiguration
	s3:PutBucketCORS



サービスプレフィックス	アクション
	s3:PutEncryptionConfiguration
	s3:PutIntelligentTieringConfiguration
	s3:PutInventoryConfiguration
	s3:PutLifecycleConfiguration
	s3:PutMetricsConfiguration
	s3:PutBucketOwnershipControls
	s3>DeleteBucketPolicy
	s3:PutBucketPublicAccessBlock
	s3:PutReplicationConfiguration
	s3>DeleteBucketWebsite
	s3>DeleteMultiRegionAccessPoint
	s3>DeleteStorageLensConfiguration
	s3:DescribeJob
	s3:DescribeMultiRegionAccessPointOperation
	s3:DissociateAccessGrantsIdentityCenter
	s3:GetAccelerateConfiguration
	s3:GetAccessGrant
	s3:GetAccessGrantsInstance
	s3:GetAccessGrantsInstanceForPrefix
	s3:GetAccessGrantsInstanceResourcePolicy
	s3:GetAccessGrantsLocation

サービスプレフィックス	アクション
	s3:GetAccessPoint
	s3:GetAccessPointConfigurationForObjectLambda
	s3:GetAccessPointForObjectLambda
	s3:GetAccessPointPolicy
	s3:GetAccessPointPolicyForObjectLambda
	s3:GetAccessPointPolicyStatus
	s3:GetAccessPointPolicyStatusForObjectLambda
	s3:GetAccountPublicAccessBlock
	s3:GetBucketAcl
	s3:GetAnalyticsConfiguration
	s3:GetBucketCORS
	s3:GetEncryptionConfiguration
	s3:GetIntelligentTieringConfiguration
	s3:GetInventoryConfiguration
	s3:GetLifecycleConfiguration
	s3:GetBucketLocation
	s3:GetBucketLogging
	s3:GetMetricsConfiguration
	s3:GetBucketNotification
	s3:GetBucketObjectLockConfiguration
	s3:GetBucketOwnershipControls

サービスプレフィックス	アクション
	s3:GetBucketPolicy
	s3:GetBucketPolicyStatus
	s3:GetBucketPublicAccessBlock
	s3:GetReplicationConfiguration
	s3:GetBucketRequestPayment
	s3:GetBucketVersioning
	s3:GetBucketWebsite
	s3:GetDataAccess
	s3:GetMultiRegionAccessPoint
	s3:GetMultiRegionAccessPointPolicy
	s3:GetMultiRegionAccessPointPolicyStatus
	s3:GetMultiRegionAccessPointRoutes
	s3:GetObjectAttributes
	s3:GetStorageLensConfiguration
	s3:GetStorageLensDashboard
	s3:ListAccessGrants
	s3:ListAccessGrantsInstances
	s3:ListAccessGrantsLocations
	s3:ListAccessPoints
	s3:ListAccessPointsForObjectLambda
	s3:ListAllMyBuckets

サービスプレフィックス	アクション
	s3:ListJobs
	s3:ListBucketMultipartUploads
	s3:ListMultiRegionAccessPoints
	s3:ListStorageLensConfigurations
	s3:PutAccelerateConfiguration
	s3:PutAccessGrantsInstanceResourcePolicy
	s3:PutAccessPointConfigurationForObjectLambda
	s3:PutAccessPointPolicy
	s3:PutAccessPointPolicyForObjectLambda
	s3:PutAccountPublicAccessBlock
	s3:PutBucketAcl
	s3:PutAnalyticsConfiguration
	s3:PutBucketCORS
	s3:PutEncryptionConfiguration
	s3:PutIntelligentTieringConfiguration
	s3:PutInventoryConfiguration
	s3:PutLifecycleConfiguration
	s3:PutBucketLogging
	s3:PutMetricsConfiguration
	s3:PutBucketNotification
	s3:PutBucketObjectLockConfiguration

サービスプレフィックス	アクション
	s3:PutBucketOwnershipControls
	s3:PutBucketPolicy
	s3:PutBucketPublicAccessBlock
	s3:PutReplicationConfiguration
	s3:PutBucketRequestPayment
	s3:PutBucketVersioning
	s3:PutBucketWebsite
	s3:PutMultiRegionAccessPointPolicy
	s3:PutStorageLensConfiguration
	s3:SubmitMultiRegionAccessPointRoutes
	s3:UpdateAccessGrantsLocation
	s3:UpdateJobPriority
	s3:UpdateJobStatus
s3-outposts	s3-outposts:CreateEndpoint
	s3-outposts>DeleteEndpoint
	s3-outposts:ListEndpoints
	s3-outposts:ListOutpostsWithS3
	s3-outposts:ListSharedEndpoints

サービスプレフィックス	アクション
sagemaker-geospatial	sagemaker-geospatial:DeleteEarthObservationJob
	sagemaker-geospatial:DeleteVectorEnrichmentJob
	sagemaker-geospatial:ExportEarthObservationJob
	sagemaker-geospatial:ExportVectorEnrichmentJob
	sagemaker-geospatial:GetEarthObservationJob
	sagemaker-geospatial:GetRasterDataCollection
	sagemaker-geospatial:GetTile
	sagemaker-geospatial:GetVectorEnrichmentJob
	sagemaker-geospatial:ListEarthObservationJobs
	sagemaker-geospatial:ListRasterDataCollections
	sagemaker-geospatial:ListVectorEnrichmentJobs
	sagemaker-geospatial:SearchRasterDataCollection
	sagemaker-geospatial:StartEarthObservationJob
	sagemaker-geospatial:StartVectorEnrichmentJob
	sagemaker-geospatial:StopEarthObservationJob
	sagemaker-geospatial:StopVectorEnrichmentJob

サービスプレフィックス	アクション
savingsplans	savingsplans:CreateSavingsPlan savingsplans>DeleteQueuedSavingsPlan savingsplans:DescribeSavingsPlanRates savingsplans:DescribeSavingsPlans savingsplans:DescribeSavingsPlansOfferingRates savingsplans:DescribeSavingsPlansOfferings savingsplans:ReturnSavingsPlan

サービスプレフィックス	アクション
schemas	schemas:CreateDiscoverer
	schemas:CreateRegistry
	schemas:CreateSchema
	schemas>DeleteDiscoverer
	schemas>DeleteRegistry
	schemas>DeleteResourcePolicy
	schemas>DeleteSchema
	schemas>DeleteSchemaVersion
	schemas:DescribeCodeBinding
	schemas:DescribeDiscoverer
	schemas:DescribeRegistry
	schemas:DescribeSchema
	schemas:ExportSchema
	schemas:GetCodeBindingSource
	schemas:GetDiscoveredSchema
	schemas:GetResourcePolicy
	schemas:ListDiscoverers
	schemas:ListRegistries
	schemas:ListSchemas
	schemas:ListSchemaVersions
	schemas:PutCodeBinding



サービスプレフィックス	アクション
	schemas:PutResourcePolicy
	schemas:SearchSchemas
	schemas:StartDiscoverer
	schemas:StopDiscoverer
	schemas:UpdateDiscoverer
	schemas:UpdateRegistry
	schemas:UpdateSchema
sdb	sdb:CreateDomain
	sdb>DeleteDomain
	sdb:DomainMetadata
	sdb:ListDomains

サービスプレフィックス	アクション
secretsmanager	secretsmanager:CancelRotateSecret
	secretsmanager:CreateSecret
	secretsmanager>DeleteResourcePolicy
	secretsmanager>DeleteSecret
	secretsmanager:DescribeSecret
	secretsmanager:GetRandomPassword
	secretsmanager:GetResourcePolicy
	secretsmanager:GetSecretValue
	secretsmanager:ListSecrets
	secretsmanager:ListSecretVersionIds
	secretsmanager:PutResourcePolicy
	secretsmanager:PutSecretValue
	secretsmanager:RemoveRegionsFromReplication
	secretsmanager:ReplicateSecretToRegions
	secretsmanager:RestoreSecret
	secretsmanager:RotateSecret
	secretsmanager:StopReplicationToReplica
	secretsmanager:UpdateSecret
	secretsmanager:ValidateResourcePolicy

サービスプレフィックス	アクション
securityhub	securityhub:AcceptAdministratorInvitation
	securityhub:AcceptInvitation
	securityhub:BatchDeleteAutomationRules
	securityhub:BatchDisableStandards
	securityhub:BatchEnableStandards
	securityhub:BatchGetAutomationRules
	securityhub:BatchGetConfigurationPolicyAssociations
	securityhub:BatchGetSecurityControls
	securityhub:BatchGetStandardsControlAssociations
	securityhub:BatchImportFindings
	securityhub:BatchUpdateAutomationRules
	securityhub:BatchUpdateFindings
	securityhub:BatchUpdateStandardsControlAssociations
	securityhub:CreateActionTarget
	securityhub:CreateAutomationRule
	securityhub:CreateConfigurationPolicy
	securityhub:CreateFindingAggregator
	securityhub:CreateInsight
	securityhub:CreateMembers
	securityhub:DeclineInvitations
	securityhub>DeleteActionTarget

サービスプレフィックス	アクション
	securityhub:DeleteConfigurationPolicy
	securityhub:DeleteFindingAggregator
	securityhub:DeleteInsight
	securityhub:DeleteInvitations
	securityhub:DeleteMembers
	securityhub:DescribeActionTargets
	securityhub:DescribeHub
	securityhub:DescribeOrganizationConfiguration
	securityhub:DescribeProducts
	securityhub:DescribeStandards
	securityhub:DisableImportFindingsForProduct
	securityhub:DisableOrganizationAdminAccount
	securityhub:DisableSecurityHub
	securityhub:DisassociateFromAdministratorAccount
	securityhub:DisassociateFromMasterAccount
	securityhub:DisassociateMembers
	securityhub:EnableImportFindingsForProduct
	securityhub:EnableOrganizationAdminAccount
	securityhub:EnableSecurityHub
	securityhub:GetAdministratorAccount
	securityhub:GetConfigurationPolicy

サービスプレフィックス	アクション
	securityhub:GetConfigurationPolicyAssociation
	securityhub:GetEnabledStandards
	securityhub:GetFindingAggregator
	securityhub:GetFindingHistory
	securityhub:GetFindings
	securityhub:GetInsightResults
	securityhub:GetInsights
	securityhub:GetInvitationsCount
	securityhub:GetMasterAccount
	securityhub:GetMembers
	securityhub:GetSecurityControlDefinition
	securityhub:InviteMembers
	securityhub:ListAutomationRules
	securityhub:ListConfigurationPolicies
	securityhub:ListConfigurationPolicyAssociations
	securityhub:ListEnabledProductsForImport
	securityhub:ListFindingAggregators
	securityhub:ListInvitations
	securityhub:ListMembers
	securityhub:ListOrganizationAdminAccounts
	securityhub:ListSecurityControlDefinitions

サービスプレフィックス	アクション
	securityhub:ListStandardsControlAssociations
	securityhub:StartConfigurationPolicyAssociation
	securityhub:StartConfigurationPolicyDisassociation
	securityhub:UpdateActionTarget
	securityhub:UpdateConfigurationPolicy
	securityhub:UpdateFindingAggregator
	securityhub:UpdateFindings
	securityhub:UpdateInsight
	securityhub:UpdateOrganizationConfiguration
	securityhub:UpdateSecurityControl
	securityhub:UpdateSecurityHubConfiguration

サービスプレフィックス	アクション
securitylake	securitylake:CreateAwsLogSource
	securitylake:CreateCustomLogSource
	securitylake:CreateDataLakeExceptionSubscription
	securitylake:CreateDataLakeOrganizationConfiguration
	securitylake:CreateSubscriber
	securitylake:CreateSubscriberNotification
	securitylake>DeleteAwsLogSource
	securitylake>DeleteCustomLogSource
	securitylake>DeleteDataLakeExceptionSubscription
	securitylake>DeleteDataLakeOrganizationConfiguration
	securitylake>DeleteSubscriber
	securitylake>DeleteSubscriberNotification
	securitylake:DeregisterDataLakeDelegatedAdministrator
	securitylake:GetDataLakeExceptionSubscription
	securitylake:GetDataLakeOrganizationConfiguration
	securitylake:GetDataLakeSources
	securitylake:GetSubscriber
	securitylake:ListDataLakes
	securitylake:ListLogSources
	securitylake:ListSubscribers
securitylake:RegisterDataLakeDelegatedAdministrator	

サービスプレフィックス	アクション
	securitylake:UpdateDataLakeExceptionSubscription
	securitylake:UpdateSubscriber
	securitylake:UpdateSubscriberNotification
serverlessrepo	serverlessrepo:CreateApplication
	serverlessrepo:CreateApplicationVersion
	serverlessrepo:CreateCloudFormationChangeSet
	serverlessrepo:CreateCloudFormationTemplate
	serverlessrepo>DeleteApplication
	serverlessrepo:GetApplication
	serverlessrepo:GetApplicationPolicy
	serverlessrepo:GetCloudFormationTemplate
	serverlessrepo:ListApplicationDependencies
	serverlessrepo:ListApplications
	serverlessrepo:ListApplicationVersions
	serverlessrepo:PutApplicationPolicy
	serverlessrepo:UnshareApplication
	serverlessrepo:UpdateApplication



サービスプレフィックス	アクション
servicecatalog	servicecatalog:AcceptPortfolioShare
	servicecatalog:AssociateBudgetWithResource
	servicecatalog:AssociatePrincipalWithPortfolio
	servicecatalog:AssociateProductWithPortfolio
	servicecatalog:AssociateServiceActionWithProvisioningArtifact
	servicecatalog:BatchAssociateServiceActionWithProvisioningArtifact
	servicecatalog:BatchDisassociateServiceActionFromProvisioningArtifact
	servicecatalog:CopyProduct
	servicecatalog>CreateConstraint
	servicecatalog>CreatePortfolio
	servicecatalog>CreatePortfolioShare
	servicecatalog>CreateProduct
	servicecatalog>CreateProvisionedProductPlan
	servicecatalog>CreateProvisioningArtifact
	servicecatalog>CreateServiceAction
	servicecatalog>DeleteConstraint
	servicecatalog>DeletePortfolio
	servicecatalog>DeletePortfolioShare
	servicecatalog>DeleteProduct
	servicecatalog>DeleteProvisionedProductPlan

サービスプレフィックス	アクション
	servicecatalog:DeleteProvisioningArtifact
	servicecatalog:DeleteServiceAction
	servicecatalog:DescribeConstraint
	servicecatalog:DescribeCopyProductStatus
	servicecatalog:DescribePortfolio
	servicecatalog:DescribePortfolioShares
	servicecatalog:DescribePortfolioShareStatus
	servicecatalog:DescribeProduct
	servicecatalog:DescribeProductAsAdmin
	servicecatalog:DescribeProductView
	servicecatalog:DescribeProvisionedProductPlan
	servicecatalog:DescribeProvisioningArtifact
	servicecatalog:DescribeProvisioningParameters
	servicecatalog:DescribeRecord
	servicecatalog:DescribeServiceAction
	servicecatalog:DescribeServiceActionExecutionParameters
	servicecatalog:DisableAWSOrganizationsAccess
	servicecatalog:DisassociateBudgetFromResource
	servicecatalog:DisassociatePrincipalFromPortfolio
	servicecatalog:DisassociateProductFromPortfolio
	servicecatalog:DisassociateServiceActionFromProvisioningArtifact

サービスプレフィックス	アクション
	<p>servicecatalog:EnableAWSOrganizationsAccess</p> <p>servicecatalog:ExecuteProvisionedProductPlan</p> <p>servicecatalog:ExecuteProvisionedProductServiceAction</p> <p>servicecatalog:GetAWSOrganizationsAccessStatus</p> <p>servicecatalog:GetProvisionedProductOutputs</p> <p>servicecatalog:ImportAsProvisionedProduct</p> <p>servicecatalog&gt;ListAcceptedPortfolioShares</p> <p>servicecatalog&gt;ListBudgetsForResource</p> <p>servicecatalog&gt;ListConstraintsForPortfolio</p> <p>servicecatalog&gt;ListLaunchPaths</p> <p>servicecatalog&gt;ListOrganizationPortfolioAccess</p> <p>servicecatalog&gt;ListPortfolioAccess</p> <p>servicecatalog&gt;ListPortfolios</p> <p>servicecatalog&gt;ListPortfoliosForProduct</p> <p>servicecatalog&gt;ListPrincipalsForPortfolio</p> <p>servicecatalog&gt;ListProvisionedProductPlans</p> <p>servicecatalog&gt;ListProvisioningArtifacts</p> <p>servicecatalog&gt;ListProvisioningArtifactsForServiceAction</p> <p>servicecatalog&gt;ListRecordHistory</p> <p>servicecatalog&gt;ListServiceActions</p> <p>servicecatalog&gt;ListServiceActionsForProvisioningArtifact</p>

サービスプレフィックス	アクション
	<p>servicecatalog:ListStackInstancesForProvisionedProduct</p> <p>servicecatalog:NotifyProvisionProductEngineWorkflowResult</p> <p>servicecatalog:NotifyTerminateProvisionedProductEngineWorkflowResult</p> <p>servicecatalog:NotifyUpdateProvisionedProductEngineWorkflowResult</p> <p>servicecatalog:ProvisionProduct</p> <p>servicecatalog:RejectPortfolioShare</p> <p>servicecatalog:ScanProvisionedProducts</p> <p>servicecatalog:SearchProducts</p> <p>servicecatalog:SearchProductsAsAdmin</p> <p>servicecatalog:SearchProvisionedProducts</p> <p>servicecatalog:TerminateProvisionedProduct</p> <p>servicecatalog:UpdateConstraint</p> <p>servicecatalog:UpdatePortfolio</p> <p>servicecatalog:UpdatePortfolioShare</p> <p>servicecatalog:UpdateProduct</p> <p>servicecatalog:UpdateProvisionedProduct</p> <p>servicecatalog:UpdateProvisionedProductProperties</p> <p>servicecatalog:UpdateProvisioningArtifact</p> <p>servicecatalog:UpdateServiceAction</p>

サービスプレフィックス	アクション
servicediscovery	servicediscovery:CreateHttpNamespace
	servicediscovery:CreatePrivateDnsNamespace
	servicediscovery:CreatePublicDnsNamespace
	servicediscovery:CreateService
	servicediscovery>DeleteNamespace
	servicediscovery>DeleteService
	servicediscovery:DeregisterInstance
	servicediscovery:GetInstance
	servicediscovery:GetInstancesHealthStatus
	servicediscovery:GetNamespace
	servicediscovery:GetOperation
	servicediscovery:GetService
	servicediscovery:ListInstances
	servicediscovery:ListNamespaces
	servicediscovery:ListOperations
	servicediscovery:ListServices
	servicediscovery:RegisterInstance
	servicediscovery:UpdateHttpNamespace
	servicediscovery:UpdateInstanceCustomHealthStatus
	servicediscovery:UpdatePrivateDnsNamespace
servicediscovery:UpdatePublicDnsNamespace	

サービスプレフィックス	アクション
	servicediscovery:UpdateService
servicequotas	servicequotas:AssociateServiceQuotaTemplate
	servicequotas>DeleteServiceQuotaIncreaseRequestFromTemplate
	servicequotas:DisassociateServiceQuotaTemplate
	servicequotas:GetAssociationForServiceQuotaTemplate
	servicequotas:GetAWSDefaultServiceQuota
	servicequotas:GetRequestedServiceQuotaChange
	servicequotas:GetServiceQuota
	servicequotas:GetServiceQuotaIncreaseRequestFromTemplate
	servicequotas:ListAWSDefaultServiceQuotas
	servicequotas:ListRequestedServiceQuotaChangeHistory
	servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota
	servicequotas:ListServiceQuotaIncreaseRequestsInTemplate
	servicequotas:ListServiceQuotas
	servicequotas:ListServices
	servicequotas:PutServiceQuotaIncreaseRequestIntoTemplate
	servicequotas:RequestServiceQuotaIncrease

サービスプレフィックス	アクション
ses	ses:BatchGetMetricData
	ses:CloneReceiptRuleSet
	ses:CreateConfigurationSet
	ses:CreateConfigurationSetEventDestination
	ses:CreateConfigurationSetTrackingOptions
	ses:CreateContact
	ses:CreateContactList
	ses:CreateCustomVerificationEmailTemplate
	ses:CreateDedicatedIpPool
	ses:CreateDeliverabilityTestReport
	ses:CreateEmailIdentity
	ses:CreateEmailIdentityPolicy
	ses:CreateEmailTemplate
	ses:CreateImportJob
	ses:CreateReceiptFilter
	ses:CreateReceiptRule
	ses:CreateReceiptRuleSet
	ses:CreateTemplate
	ses>DeleteConfigurationSet
	ses>DeleteConfigurationSetEventDestination
	ses>DeleteConfigurationSetTrackingOptions

サービスプレフィックス	アクション
	ses>DeleteContact
	ses>DeleteContactList
	ses>DeleteCustomVerificationEmailTemplate
	ses>DeleteDedicatedIpPool
	ses>DeleteEmailIdentity
	ses>DeleteEmailIdentityPolicy
	ses>DeleteEmailTemplate
	ses>DeleteIdentity
	ses>DeleteIdentityPolicy
	ses>DeleteReceiptFilter
	ses>DeleteReceiptRule
	ses>DeleteReceiptRuleSet
	ses>DeleteSuppressedDestination
	ses>DeleteTemplate
	ses>DeleteVerifiedEmailAddress
	ses:DescribeActiveReceiptRuleSet
	ses:DescribeConfigurationSet
	ses:DescribeReceiptRule
	ses:DescribeReceiptRuleSet
	ses:GetAccount
	ses:GetAccountSendingEnabled



サービスプレフィックス	アクション
	ses:GetBlacklistReports
	ses:GetConfigurationSet
	ses:GetConfigurationSetEventDestinations
	ses:GetContact
	ses:GetContactList
	ses:GetCustomVerificationEmailTemplate
	ses:GetDedicatedIp
	ses:GetDedicatedIpPool
	ses:GetDedicatedIps
	ses:GetDeliverabilityDashboardOptions
	ses:GetDeliverabilityTestReport
	ses:GetDomainDeliverabilityCampaign
	ses:GetDomainStatisticsReport
	ses:GetEmailIdentity
	ses:GetEmailIdentityPolicies
	ses:GetEmailTemplate
	ses:GetIdentityDkimAttributes
	ses:GetIdentityMailFromDomainAttributes
	ses:GetIdentityNotificationAttributes
	ses:GetIdentityPolicies
	ses:GetIdentityVerificationAttributes

サービスプレフィックス	アクション
	ses:GetImportJob
	ses:GetMessageInsights
	ses:GetSendQuota
	ses:GetSendStatistics
	ses:GetSuppressedDestination
	ses:GetTemplate
	ses:ListConfigurationSets
	ses:ListContactLists
	ses:ListContacts
	ses:ListCustomVerificationEmailTemplates
	ses:ListDedicatedIpPools
	ses:ListDeliverabilityTestReports
	ses:ListDomainDeliverabilityCampaigns
	ses:ListEmailIdentities
	ses:ListEmailTemplates
	ses:ListExportJobs
	ses:ListIdentities
	ses:ListIdentityPolicies
	ses:ListImportJobs
	ses:ListReceiptFilters
	ses:ListReceiptRuleSets

サービスプレフィックス	アクション
	ses:ListRecommendations
	ses:ListSuppressedDestinations
	ses:ListTemplates
	ses:ListVerifiedEmailAddresses
	ses:PutAccountDedicatedIpWarmupAttributes
	ses:PutAccountDetails
	ses:PutAccountSendingAttributes
	ses:PutAccountSuppressionAttributes
	ses:PutAccountVdmAttributes
	ses:PutConfigurationSetDeliveryOptions
	ses:PutConfigurationSetReputationOptions
	ses:PutConfigurationSetSendingOptions
	ses:PutConfigurationSetSuppressionOptions
	ses:PutConfigurationSetTrackingOptions
	ses:PutConfigurationSetVdmOptions
	ses:PutDedicatedIpInPool
	ses:PutDedicatedIpPoolScalingAttributes
	ses:PutDedicatedIpWarmupAttributes
	ses:PutDeliverabilityDashboardOption
	ses:PutEmailIdentityConfigurationSetAttributes
	ses:PutEmailIdentityDkimAttributes

サービスプレフィックス	アクション
	ses:PutEmailIdentityDkimSigningAttributes
	ses:PutEmailIdentityFeedbackAttributes
	ses:PutEmailIdentityMailFromAttributes
	ses:PutIdentityPolicy
	ses:PutSuppressedDestination
	ses:ReorderReceiptRuleSet
	ses:SendBounce
	ses:SendCustomVerificationEmail
	ses:SetActiveReceiptRuleSet
	ses:SetIdentityDkimEnabled
	ses:SetIdentityFeedbackForwardingEnabled
	ses:SetIdentityHeadersInNotificationsEnabled
	ses:SetIdentityMailFromDomain
	ses:SetIdentityNotificationTopic
	ses:SetReceiptRulePosition
	ses:TestRenderEmailTemplate
	ses:TestRenderTemplate
	ses:UpdateAccountSendingEnabled
	ses:UpdateConfigurationSetEventDestination
	ses:UpdateConfigurationSetReputationMetricsEnabled
	ses:UpdateConfigurationSetSendingEnabled

サービスプレフィックス	アクション
	<code>ses:UpdateConfigurationSetTrackingOptions</code>
	<code>ses:UpdateContact</code>
	<code>ses:UpdateContactList</code>
	<code>ses:UpdateCustomVerificationEmailTemplate</code>
	<code>ses:UpdateEmailIdentityPolicy</code>
	<code>ses:UpdateEmailTemplate</code>
	<code>ses:UpdateReceiptRule</code>
	<code>ses:UpdateTemplate</code>
	<code>ses:VerifyDomainDkim</code>
	<code>ses:VerifyDomainIdentity</code>
	<code>ses:VerifyEmailAddress</code>
	<code>ses:VerifyEmailIdentity</code>

サービスプレフィックス	アクション
shield	shield:AssociateDRTLogBucket
	shield:AssociateHealthCheck
	shield:AssociateProactiveEngagementDetails
	shield:CreateProtection
	shield:CreateProtectionGroup
	shield:CreateSubscription
	shield>DeleteProtection
	shield>DeleteProtectionGroup
	shield>DeleteSubscription
	shield:DescribeAttack
	shield:DescribeAttackStatistics
	shield:DescribeDRTAccess
	shield:DescribeEmergencyContactSettings
	shield:DescribeProtection
	shield:DescribeProtectionGroup
	shield:DescribeSubscription
	shield:DisableApplicationLayerAutomaticResponse
	shield:DisableProactiveEngagement
	shield:DisassociateDRTLogBucket
	shield:DisassociateDRTRole
	shield:DisassociateHealthCheck

サービスプレフィックス	アクション
	shield:EnableApplicationLayerAutomaticResponse
	shield:EnableProactiveEngagement
	shield:GetSubscriptionState
	shield:ListAttacks
	shield:ListProtectionGroups
	shield:ListProtections
	shield:ListResourcesInProtectionGroup
	shield:UpdateApplicationLayerAutomaticResponse
	shield:UpdateEmergencyContactSettings
	shield:UpdateProtectionGroup
	shield:UpdateSubscription

サービスプレフィックス	アクション
signer	signer:AddProfilePermission
	signer:CancelSigningProfile
	signer:DescribeSigningJob
	signer:GetRevocationStatus
	signer:GetSigningPlatform
	signer:GetSigningProfile
	signer:ListProfilePermissions
	signer:ListSigningJobs
	signer:ListSigningPlatforms
	signer:ListSigningProfiles
	signer:PutSigningProfile
	signer:RemoveProfilePermission
	signer:RevokeSignature
	signer:RevokeSigningProfile
	signer:SignPayload
	signer:StartSigningJob



サービスプレフィックス	アクション
simspaceweaver	simspaceweaver:CreateSnapshot
	simspaceweaver>DeleteApp
	simspaceweaver>DeleteSimulation
	simspaceweaver:DescribeApp
	simspaceweaver:DescribeSimulation
	simspaceweaver:ListApps
	simspaceweaver:ListSimulations
	simspaceweaver:StartApp
	simspaceweaver:StartClock
	simspaceweaver:StartSimulation
	simspaceweaver:StopApp
	simspaceweaver:StopClock
	simspaceweaver:StopSimulation

サービスプレフィックス	アクション
sms	sms:CreateApp
	sms:CreateReplicationJob
	sms>DeleteApp
	sms>DeleteAppLaunchConfiguration
	sms>DeleteAppReplicationConfiguration
	sms>DeleteAppValidationConfiguration
	sms>DeleteReplicationJob
	sms>DeleteServerCatalog
	sms:DisassociateConnector
	sms:GenerateChangeSet
	sms:GenerateTemplate
	sms:GetApp
	sms:GetAppLaunchConfiguration
	sms:GetAppReplicationConfiguration
	sms:GetAppValidationConfiguration
	sms:GetAppValidationOutput
	sms:GetConnectors
	sms:GetReplicationJobs
	sms:GetReplicationRuns
	sms:GetServers
sms:ImportAppCatalog	

サービスプレフィックス	アクション
	sms:ImportServerCatalog
	sms:LaunchApp
	sms:ListApps
	sms:NotifyAppValidationOutput
	sms:PutAppLaunchConfiguration
	sms:PutAppReplicationConfiguration
	sms:PutAppValidationConfiguration
	sms:StartAppReplication
	sms:StartOnDemandAppReplication
	sms:StartOnDemandReplicationRun
	sms:StopAppReplication
	sms:TerminateApp
	sms:UpdateApp
	sms:UpdateReplicationJob

サービスプレフィックス	アクション
sms-voice	sms-voice:AssociateProtectConfiguration
	sms-voice:CreateConfigurationSet
	sms-voice:CreateConfigurationSetEventDestination
	sms-voice:CreateEventDestination
	sms-voice:CreateOptOutList
	sms-voice:CreatePool
	sms-voice:CreateProtectConfiguration
	sms-voice:CreateRegistration
	sms-voice:CreateRegistrationAssociation
	sms-voice:CreateRegistrationAttachment
	sms-voice:CreateRegistrationVersion
	sms-voice:CreateVerifiedDestinationNumber
	sms-voice>DeleteAccountDefaultProtectConfiguration
	sms-voice>DeleteConfigurationSet
	sms-voice>DeleteConfigurationSetEventDestination
	sms-voice>DeleteDefaultMessageType
	sms-voice>DeleteDefaultSenderId
	sms-voice>DeleteEventDestination
	sms-voice>DeleteKeyword
	sms-voice>DeleteMediaMessageSpendLimitOverride
sms-voice>DeleteOptedOutNumber	

サービスプレフィックス	アクション
	sms-voice:DeleteOptOutList
	sms-voice:DeletePool
	sms-voice:DeleteProtectConfiguration
	sms-voice:DeleteRegistration
	sms-voice:DeleteRegistrationAttachment
	sms-voice:DeleteTextMessageSpendLimitOverride
	sms-voice:DeleteVerifiedDestinationNumber
	sms-voice:DeleteVoiceMessageSpendLimitOverride
	sms-voice:DescribeAccountAttributes
	sms-voice:DescribeAccountLimits
	sms-voice:DescribeConfigurationSets
	sms-voice:DescribeKeywords
	sms-voice:DescribeOptedOutNumbers
	sms-voice:DescribeOptOutLists
	sms-voice:DescribePhoneNumbers
	sms-voice:DescribePools
	sms-voice:DescribeProtectConfigurations
	sms-voice:DescribeRegistrationAttachments
	sms-voice:DescribeRegistrationFieldDefinitions
	sms-voice:DescribeRegistrationFieldValues
	sms-voice:DescribeRegistrations

サービスプレフィックス	アクション
	sms-voice:DescribeRegistrationSectionDefinitions
	sms-voice:DescribeRegistrationTypeDefinitions
	sms-voice:DescribeRegistrationVersions
	sms-voice:DescribeSenderIds
	sms-voice:DescribeSpendLimits
	sms-voice:DescribeVerifiedDestinationNumbers
	sms-voice:DisassociateOriginationIdentity
	sms-voice:DisassociateProtectConfiguration
	sms-voice:DiscardRegistrationVersion
	sms-voice:GetConfigurationSetEventDestinations
	sms-voice:GetProtectConfigurationCountryRuleSet
	sms-voice:ListConfigurationSets
	sms-voice:ListPoolOriginationIdentities
	sms-voice:ListRegistrationAssociations
	sms-voice:PutKeyword
	sms-voice:PutOptedOutNumber
	sms-voice:ReleasePhoneNumber
	sms-voice:ReleaseSenderId
	sms-voice:RequestPhoneNumber
	sms-voice:RequestSenderId
	sms-voice:SendDestinationNumberVerificationCode

サービスプレフィックス	アクション
	<p>sms-voice:SetAccountDefaultProtectConfiguration</p> <p>sms-voice:SetDefaultMessageType</p> <p>sms-voice:SetDefaultSenderId</p> <p>sms-voice:SetMediaMessageSpendLimitOverride</p> <p>sms-voice:SetTextMessageSpendLimitOverride</p> <p>sms-voice:SetVoiceMessageSpendLimitOverride</p> <p>sms-voice:SubmitRegistrationVersion</p> <p>sms-voice:UpdateConfigurationSetEventDestination</p> <p>sms-voice:UpdateEventDestination</p> <p>sms-voice:UpdatePhoneNumber</p> <p>sms-voice:UpdatePool</p> <p>sms-voice:UpdateProtectConfiguration</p> <p>sms-voice:UpdateProtectConfigurationCountryRuleSet</p> <p>sms-voice:UpdateSenderId</p>

サービスプレフィックス	アクション
snowball	snowball:CancelCluster
	snowball:CancelJob
	snowball:CreateAddress
	snowball:CreateCluster
	snowball:CreateJob
	snowball:CreateLongTermPricing
	snowball:CreateReturnShippingLabel
	snowball:DescribeAddress
	snowball:DescribeAddresses
	snowball:DescribeCluster
	snowball:DescribeJob
	snowball:DescribeReturnShippingLabel
	snowball:GetJobManifest
	snowball:GetJobUnlockCode
	snowball:GetSnowballUsage
	snowball:GetSoftwareUpdates
	snowball>ListClusterJobs
	snowball>ListClusters
	snowball>ListCompatibleImages
	snowball>ListJobs
snowball>ListLongTermPricing	



サービスプレフィックス	アクション
	snowball:ListPickupLocations snowball:ListServiceVersions snowball:UpdateCluster snowball:UpdateJob snowball:UpdateJobShipmentState snowball:UpdateLongTermPricing
sqs	sqs:AddPermission sqs:CancelMessageMoveTask sqs:CreateQueue sqs>DeleteQueue sqs:PurgeQueue sqs:RemovePermission sqs:SetQueueAttributes

サービスプレフィックス	アクション
ssm	ssm:AssociateOpsItemRelatedItem
	SSM: CancelCommand
	ssm:CancelMaintenanceWindowExecution
	ssm:CreateActivation
	ssm:CreateAssociation
	ssm:CreateAssociationBatch
	ssm:CreateDocument
	ssm:CreateMaintenanceWindow
	ssm:CreateOpsItem
	ssm:CreateOpsMetadata
	ssm:CreatePatchBaseline
	ssm:CreateResourceDataSync
	ssm>DeleteActivation
	ssm>DeleteAssociation
	ssm>DeleteDocument
	ssm>DeleteInventory
	ssm>DeleteMaintenanceWindow
	ssm>DeleteOpsItem
	ssm>DeleteOpsMetadata
	ssm>DeleteParameter
	ssm>DeleteParameters

サービスプレフィックス	アクション
	ssm:DeletePatchBaseline
	ssm:DeleteResourceDataSync
	ssm:DeleteResourcePolicy
	SSM: DeregisterManagedInstance
	ssm:DeregisterPatchBaselineForPatchGroup
	ssm:DeregisterTargetFromMaintenanceWindow
	ssm:DeregisterTaskFromMaintenanceWindow
	ssm:DescribeActivations
	ssm:DescribeAssociation
	ssm:DescribeAssociationExecutions
	ssm:DescribeAssociationExecutionTargets
	ssm:DescribeAutomationExecutions
	ssm:DescribeAutomationStepExecutions
	ssm:DescribeAvailablePatches
	ssm:DescribeDocument
	ssm:DescribeDocumentParameters
	ssm:DescribeDocumentPermission
	ssm:DescribeEffectiveInstanceAssociations
	ssm:DescribeEffectivePatchesForPatchBaseline
	ssm:DescribeInstanceAssociationsStatus
	ssm:DescribeInstanceInformation

サービスプレフィックス	アクション
	ssm:DescribeInstancePatches
	ssm:DescribeInstancePatchStates
	ssm:DescribeInstancePatchStatesForPatchGroup
	ssm:DescribeInstanceProperties
	ssm:DescribeInventoryDeletions
	ssm:DescribeMaintenanceWindowExecutions
	ssm:DescribeMaintenanceWindowExecutionTaskInvocations
	ssm:DescribeMaintenanceWindowExecutionTasks
	ssm:DescribeMaintenanceWindows
	ssm:DescribeMaintenanceWindowSchedule
	ssm:DescribeMaintenanceWindowsForTarget
	ssm:DescribeMaintenanceWindowTargets
	ssm:DescribeMaintenanceWindowTasks
	ssm:DescribeOpsItems
	ssm:DescribeParameters
	ssm:DescribePatchBaselines
	ssm:DescribePatchGroups
	ssm:DescribePatchGroupState
	ssm:DescribePatchProperties
	ssm:DescribeSessions
	ssm:DisassociateOpsItemRelatedItem

サービスプレフィックス	アクション
	ssm:GetAutomationExecution
	ssm:GetCalendarState
	ssm:GetCommandInvocation
	ssm:GetConnectionStatus
	ssm:GetDefaultPatchBaseline
	ssm:GetDeployablePatchSnapshotForInstance
	ssm:GetDocument
	ssm:GetInventory
	ssm:GetInventorySchema
	ssm:GetMaintenanceWindow
	ssm:GetMaintenanceWindowExecution
	ssm:GetMaintenanceWindowExecutionTask
	ssm:GetMaintenanceWindowExecutionTaskInvocation
	ssm:GetMaintenanceWindowTask
	ssm:GetOpsItem
	ssm:GetOpsMetadata
	ssm:GetOpsSummary
	ssm:GetParameter
	ssm:GetParameterHistory
	ssm:GetParameters
	ssm:GetParametersByPath

サービスプレフィックス	アクション
	ssm:GetPatchBaseline
	ssm:GetPatchBaselineForPatchGroup
	ssm:GetResourcePolicies
	ssm:GetServiceSetting
	ssm:LabelParameterVersion
	ssm:ListAssociations
	ssm:ListAssociationVersions
	ssm:ListCommandInvocations
	ssm:ListCommands
	ssm:ListComplianceItems
	ssm:ListComplianceSummaries
	ssm:ListDocumentMetadataHistory
	ssm:ListDocuments
	ssm:ListDocumentVersions
	ssm:ListInstanceAssociations
	ssm:ListInventoryEntries
	ssm:ListOpsItemEvents
	ssm:ListOpsItemRelatedItems
	ssm:ListOpsMetadata
	ssm:ListResourceComplianceSummaries
	ssm:ListResourceDataSync

サービスプレフィックス	アクション
	ssm:ModifyDocumentPermission
	ssm:PutComplianceItems
	ssm:PutInventory
	ssm:PutParameter
	ssm:PutResourcePolicy
	ssm:RegisterDefaultPatchBaseline
	ssm:RegisterManagedInstance
	ssm:RegisterPatchBaselineForPatchGroup
	ssm:RegisterTargetWithMaintenanceWindow
	ssm:RegisterTaskWithMaintenanceWindow
	ssm:ResetServiceSetting
	ssm:ResumeSession
	ssm:SendAutomationSignal
	SSM: SendCommand
	ssm:StartAssociationsOnce
	ssm:StartAutomationExecution
	ssm:StartChangeRequestExecution
	ssm:StartSession
	ssm:StopAutomationExecution
	ssm:TerminateSession
	ssm:UnlabelParameterVersion

サービスプレフィックス	アクション
	ssm:UpdateAssociation
	ssm:UpdateAssociationStatus
	ssm:UpdateDocument
	ssm:UpdateDocumentDefaultVersion
	ssm:UpdateDocumentMetadata
	ssm:UpdateInstanceInformation
	ssm:UpdateMaintenanceWindow
	ssm:UpdateMaintenanceWindowTarget
	ssm:UpdateMaintenanceWindowTask
	ssm:UpdateManagedInstanceRole
	ssm:UpdateOpsItem
	ssm:UpdateOpsMetadata
	ssm:UpdatePatchBaseline
	ssm:UpdateResourceDataSync
	ssm:UpdateServiceSetting



サービスプレフィックス	アクション
ssm-incidents	ssm-incidents:BatchGetIncidentFindings
	ssm-incidents:CreateReplicationSet
	ssm-incidents:CreateResponsePlan
	ssm-incidents:CreateTimelineEvent
	ssm-incidents>DeleteIncidentRecord
	ssm-incidents>DeleteReplicationSet
	ssm-incidents>DeleteResourcePolicy
	ssm-incidents>DeleteResponsePlan
	ssm-incidents>DeleteTimelineEvent
	ssm-incidents:GetIncidentRecord
	ssm-incidents:GetReplicationSet
	ssm-incidents:GetResourcePolicies
	ssm-incidents:GetResponsePlan
	ssm-incidents:GetTimelineEvent
	ssm-incidents:ListIncidentFindings
	ssm-incidents:ListIncidentRecords
	ssm-incidents:ListRelatedItems
	ssm-incidents:ListReplicationSets
	ssm-incidents:ListResponsePlans
	ssm-incidents:ListTimelineEvents
	ssm-incidents:PutResourcePolicy

サービスプレフィックス	アクション
	ssm-incidents:StartIncident
	ssm-incidents:UpdateDeletionProtection
	ssm-incidents:UpdateIncidentRecord
	ssm-incidents:UpdateRelatedItems
	ssm-incidents:UpdateReplicationSet
	ssm-incidents:UpdateResponsePlan
	ssm-incidents:UpdateTimelineEvent

サービスプレフィックス	アクション
ssm-sap	ssm-sap:BackupDatabase
	ssm-sap>DeleteResourcePermission
	ssm-sap:DeregisterApplication
	ssm-sap:GetApplication
	ssm-sap:GetComponent
	ssm-sap:GetDatabase
	ssm-sap:GetOperation
	ssm-sap:GetResourcePermission
	ssm-sap:ListApplications
	ssm-sap:ListComponents
	ssm-sap:ListDatabases
	ssm-sap:ListOperationEvents
	ssm-sap:ListOperations
	ssm-sap:PutResourcePermission
	ssm-sap:RegisterApplication
	ssm-sap:RestoreDatabase
	ssm-sap:StartApplication
	ssm-sap:StartApplicationRefresh
	ssm-sap:StopApplication
	ssm-sap:UpdateApplicationSettings
	ssm-sap:UpdateHANABackupSettings

サービスプレフィックス	アクション
states	states:CreateActivity
	states:CreateStateMachine
	states:CreateStateMachineAlias
	states>DeleteActivity
	states>DeleteStateMachine
	states>DeleteStateMachineAlias
	states>DeleteStateMachineVersion
	states:DescribeActivity
	states:DescribeExecution
	states:DescribeMapRun
	states:DescribeStateMachine
	states:DescribeStateMachineAlias
	states:DescribeStateMachineForExecution
	states:GetExecutionHistory
	states:ListActivities
	states:ListExecutions
	states:ListMapRuns
	states:ListStateMachineAliases
	states:ListStateMachines
	states:ListStateMachineVersions
states:SendTaskFailure	

サービスプレフィックス	アクション
	states:SendTaskHeartbeat
	states:SendTaskSuccess
	states:StartExecution
	states:StopExecution
	states:UpdateMapRun
	states:UpdateStateMachine
	states:UpdateStateMachineAlias
	states:ValidateStateMachineDefinition
sts	sts:AssumeRole
	sts:AssumeRoleWithSAML
	sts:AssumeRoleWithWebIdentity
	sts:DecodeAuthorizationMessage
	sts:GetAccessKeyInfo
	sts:GetCallerIdentity
	sts:GetFederationToken
	sts:GetSessionToken

サービスプレフィックス	アクション
swf	swf:DeprecateActivityType
	swf:DeprecateDomain
	swf:DeprecateWorkflowType
	swf:DescribeActivityType
	swf:DescribeDomain
	swf:DescribeWorkflowType
	swf:ListActivityTypes
	swf:ListDomains
	swf:ListWorkflowTypes
	swf:RegisterActivityType
	swf:RegisterDomain
	swf:RegisterWorkflowType
	swf:UndeprecateActivityType
	swf:UndeprecateDomain
	swf:UndeprecateWorkflowType

サービスプレフィックス	アクション
synthetics	synthetics:AssociateResource
	synthetics:CreateCanary
	synthetics:CreateGroup
	synthetics>DeleteCanary
	synthetics>DeleteGroup
	synthetics:DescribeCanaries
	synthetics:DescribeCanariesLastRun
	synthetics:DescribeRuntimeVersions
	synthetics:DisassociateResource
	synthetics:GetCanary
	synthetics:GetCanaryRuns
	synthetics:GetGroup
	synthetics>ListAssociatedGroups
	synthetics>ListGroupResources
	synthetics:ListGroupsWithResources
	synthetics:StartCanary
	synthetics:StopCanary
	synthetics:UpdateCanary

サービスプレフィックス	アクション
タグ	tag:DescribeReportCreation tag:GetComplianceSummary tag:GetResources tag:StartReportCreation



サービスプレフィックス	アクション
textextract	textextract:AnalyzeDocument
	textextract:AnalyzeExpense
	textextract:AnalyzeID
	textextract:CreateAdapter
	textextract:CreateAdapterVersion
	textextract>DeleteAdapter
	textextract>DeleteAdapterVersion
	textextract:DetectDocumentText
	textextract:GetAdapter
	textextract:GetAdapterVersion
	textextract:GetDocumentAnalysis
	textextract:GetDocumentTextDetection
	textextract:GetExpenseAnalysis
	textextract:GetLendingAnalysis
	textextract:GetLendingAnalysisSummary
	textextract:ListAdapters
	textextract:ListAdapterVersions
	textextract:StartDocumentAnalysis
	textextract:StartDocumentTextDetection
	textextract:StartExpenseAnalysis
	textextract:StartLendingAnalysis

サービスプレフィックス	アクション
	textract:UpdateAdapter

サービスプレフィックス	アクション
timestream	timestream:CancelQuery
	timestream:CreateDatabase
	timestream:CreateScheduledQuery
	timestream:CreateTable
	timestream>DeleteDatabase
	timestream>DeleteScheduledQuery
	timestream>DeleteTable
	timestream:DescribeAccountSettings
	timestream:DescribeDatabase
	timestream:DescribeScheduledQuery
	timestream:DescribeTable
	timestream:ExecuteScheduledQuery
	timestream>ListBatchLoadTasks
	timestream>ListDatabases
	timestream>ListScheduledQueries
	timestream>ListTables
	timestream:PrepareQuery
	timestream:UpdateAccountSettings
	timestream:UpdateDatabase
	timestream:UpdateScheduledQuery
	timestream:UpdateTable

サービスプレフィックス	アクション
tnb	tnb:CancelSolNetworkOperation
	tnb:CreateSolFunctionPackage
	tnb:CreateSolNetworkInstance
	tnb:CreateSolNetworkPackage
	tnb>DeleteSolFunctionPackage
	tnb>DeleteSolNetworkInstance
	tnb>DeleteSolNetworkPackage
	tnb:GetSolFunctionInstance
	tnb:GetSolFunctionPackage
	tnb:GetSolFunctionPackageContent
	tnb:GetSolFunctionPackageDescriptor
	tnb:GetSolNetworkInstance
	tnb:GetSolNetworkOperation
	tnb:GetSolNetworkPackage
	tnb:GetSolNetworkPackageContent
	tnb:GetSolNetworkPackageDescriptor
	tnb:InstantiateSolNetworkInstance
	tnb:ListSolFunctionInstances
	tnb:ListSolFunctionPackages
	tnb:ListSolNetworkInstances
	tnb:ListSolNetworkOperations

サービスプレフィックス	アクション
	tnb:ListSolNetworkPackages
	tnb:PutSolFunctionPackageContent
	tnb:PutSolNetworkPackageContent
	tnb:TerminateSolNetworkInstance
	tnb:UpdateSolFunctionPackage
	tnb:UpdateSolNetworkInstance
	tnb:UpdateSolNetworkPackage
	tnb:ValidateSolFunctionPackageContent
	tnb:ValidateSolNetworkPackageContent

サービスプレフィックス	アクション
transcribe	transcribe:CreateCallAnalyticsCategory
	transcribe:CreateLanguageModel
	transcribe:CreateMedicalVocabulary
	transcribe:CreateVocabulary
	transcribe:CreateVocabularyFilter
	transcribe>DeleteCallAnalyticsCategory
	transcribe>DeleteCallAnalyticsJob
	transcribe>DeleteLanguageModel
	transcribe>DeleteMedicalScribeJob
	transcribe>DeleteMedicalTranscriptionJob
	transcribe>DeleteMedicalVocabulary
	transcribe>DeleteTranscriptionJob
	transcribe>DeleteVocabulary
	transcribe>DeleteVocabularyFilter
	transcribe:DescribeLanguageModel
	transcribe:GetCallAnalyticsCategory
	transcribe:GetCallAnalyticsJob
	transcribe:GetMedicalScribeJob
	transcribe:GetMedicalTranscriptionJob
	transcribe:GetMedicalVocabulary
	transcribe:GetTranscriptionJob

サービスプレフィックス	アクション
	transcribe:GetVocabulary
	transcribe:GetVocabularyFilter
	transcribe:ListCallAnalyticsCategories
	transcribe:ListCallAnalyticsJobs
	transcribe:ListLanguageModels
	transcribe:ListMedicalScribeJobs
	transcribe:ListMedicalTranscriptionJobs
	transcribe:ListMedicalVocabularies
	transcribe:ListTranscriptionJobs
	transcribe:ListVocabularies
	transcribe:ListVocabularyFilters
	transcribe:StartCallAnalyticsJob
	transcribe:StartCallAnalyticsStreamTranscription
	transcribe:StartCallAnalyticsStreamTranscriptionWebSocket
	transcribe:StartMedicalScribeJob
	transcribe:StartMedicalStreamTranscription
	transcribe:StartMedicalStreamTranscriptionWebSocket
	transcribe:StartMedicalTranscriptionJob
	transcribe:StartStreamTranscription
	transcribe:StartStreamTranscriptionWebSocket
	transcribe:StartTranscriptionJob

サービスプレフィックス	アクション
	transcribe:UpdateCallAnalyticsCategory transcribe:UpdateMedicalVocabulary transcribe:UpdateVocabulary transcribe:UpdateVocabularyFilter



サービスプレフィックス	アクション
移管	transfer:CreateAccess
	transfer:CreateAgreement
	transfer:CreateConnector
	transfer:CreateProfile
	transfer:CreateServer
	transfer:CreateUser
	transfer:CreateWorkflow
	transfer>DeleteAccess
	transfer>DeleteAgreement
	transfer>DeleteCertificate
	transfer>DeleteConnector
	transfer>DeleteHostKey
	transfer>DeleteProfile
	transfer>DeleteServer
	transfer>DeleteSshPublicKey
	transfer>DeleteUser
	transfer>DeleteWorkflow
	transfer:DescribeAccess
	transfer:DescribeAgreement
	transfer:DescribeCertificate
transfer:DescribeConnector	

サービスプレフィックス	アクション
	transfer:DescribeExecution
	transfer:DescribeHostKey
	transfer:DescribeProfile
	transfer:DescribeSecurityPolicy
	transfer:DescribeServer
	transfer:DescribeUser
	transfer:DescribeWorkflow
	transfer:ImportCertificate
	transfer:ImportHostKey
	transfer:ImportSshPublicKey
	transfer:ListAccesses
	transfer:ListCertificates
	transfer:ListConnectors
	transfer:ListExecutions
	transfer:ListHostKeys
	transfer:ListProfiles
	transfer:ListSecurityPolicies
	transfer:ListServers
	transfer:ListUsers
	transfer:ListWorkflows
	transfer:SendWorkflowStepState

サービスプレフィックス	アクション
	transfer:StartDirectoryListing
	transfer:StartFileTransfer
	transfer:StartServer
	transfer:StopServer
	transfer:TestConnection
	transfer:TestIdentityProvider
	transfer:UpdateAccess
	transfer:UpdateAgreement
	transfer:UpdateCertificate
	transfer:UpdateConnector
	transfer:UpdateHostKey
	transfer:UpdateProfile
	transfer:UpdateServer
	transfer:UpdateUser

サービスプレフィックス	アクション
translate	translate:CreateParallelData
	translate>DeleteParallelData
	translate>DeleteTerminology
	translate:DescribeTextTranslationJob
	translate:GetParallelData
	translate:GetTerminology
	translate:ImportTerminology
	translate:ListLanguages
	translate:ListParallelData
	translate:ListTerminologies
	translate:ListTextTranslationJobs
	translate:StartTextTranslationJob
	translate:StopTextTranslationJob
	translate:TranslateDocument
	translate:TranslateText
	translate:UpdateParallelData

サービスプレフィックス	アクション
voiceid	voiceid:AssociateFraudster
	voiceid:CreateDomain
	voiceid:CreateWatchlist
	voiceid>DeleteDomain
	voiceid>DeleteFraudster
	voiceid>DeleteSpeaker
	voiceid>DeleteWatchlist
	voiceid:DescribeDomain
	voiceid:DescribeFraudster
	voiceid:DescribeFraudsterRegistrationJob
	voiceid:DescribeSpeaker
	voiceid:DescribeSpeakerEnrollmentJob
	voiceid:DescribeWatchlist
	voiceid:DisassociateFraudster
	voiceid:EvaluateSession
	voiceid:ListDomains
	voiceid:ListFraudsterRegistrationJobs
	voiceid:ListFraudsters
	voiceid:ListSpeakerEnrollmentJobs
	voiceid:ListSpeakers
	voiceid:ListWatchlists

サービスプレフィックス	アクション
	voiceid:OptOutSpeaker
	voiceid:StartFraudsterRegistrationJob
	voiceid:StartSpeakerEnrollmentJob
	voiceid:UpdateDomain
	voiceid:UpdateWatchlist

サービスプレフィックス	アクション
vpc-lattice	vpc-lattice:CreateAccessLogSubscription
	vpc-lattice:CreateListener
	vpc-lattice:CreateRule
	vpc-lattice:CreateService
	vpc-lattice:CreateServiceNetwork
	vpc-lattice:CreateServiceNetworkServiceAssociation
	vpc-lattice:CreateServiceNetworkVpcAssociation
	vpc-lattice:CreateTargetGroup
	vpc-lattice>DeleteAccessLogSubscription
	vpc-lattice>DeleteAuthPolicy
	vpc-lattice>DeleteListener
	vpc-lattice>DeleteResourcePolicy
	vpc-lattice>DeleteRule
	vpc-lattice>DeleteService
	vpc-lattice>DeleteServiceNetwork
	vpc-lattice>DeleteServiceNetworkServiceAssociation
	vpc-lattice>DeleteServiceNetworkVpcAssociation
	vpc-lattice>DeleteTargetGroup
	vpc-lattice:DeregisterTargets
	vpc-lattice:GetAccessLogSubscription
	vpc-lattice:GetAuthPolicy

サービスプレフィックス	アクション
	vpc-lattice:GetListener
	vpc-lattice:GetResourcePolicy
	vpc-lattice:GetRule
	vpc-lattice:GetService
	vpc-lattice:GetServiceNetwork
	vpc-lattice:GetServiceNetworkServiceAssociation
	vpc-lattice:GetServiceNetworkVpcAssociation
	vpc-lattice:GetTargetGroup
	vpc-lattice:ListAccessLogSubscriptions
	vpc-lattice:ListListeners
	vpc-lattice:ListRules
	vpc-lattice:ListServiceNetworks
	vpc-lattice:ListServiceNetworkServiceAssociations
	vpc-lattice:ListServiceNetworkVpcAssociations
	vpc-lattice:ListServices
	vpc-lattice:ListTargetGroups
	vpc-lattice:ListTargets
	vpc-lattice:PutAuthPolicy
	vpc-lattice:PutResourcePolicy
	vpc-lattice:RegisterTargets
	vpc-lattice:UpdateAccessLogSubscription



サービスプレフィックス	アクション
	vpc-lattice:UpdateListener
	vpc-lattice:UpdateRule
	vpc-lattice:UpdateService
	vpc-lattice:UpdateServiceNetwork
	vpc-lattice:UpdateServiceNetworkVpcAssociation
	vpc-lattice:UpdateTargetGroup

サービスプレフィックス	アクション
wafv2	wafv2:AssociateWebACL
	wafv2:CheckCapacity
	wafv2:CreateAPIKey
	wafv2:CreateIPSet
	wafv2:CreateRegexPatternSet
	wafv2:CreateRuleGroup
	wafv2:CreateWebACL
	wafv2>DeleteAPIKey
	wafv2>DeleteFirewallManagerRuleGroups
	wafv2>DeleteIPSet
	wafv2>DeleteLoggingConfiguration
	wafv2>DeletePermissionPolicy
	wafv2>DeleteRegexPatternSet
	wafv2>DeleteRuleGroup
	wafv2>DeleteWebACL
	wafv2:DescribeAllManagedProducts
	wafv2:DescribeManagedProductsByVendor
	wafv2:DescribeManagedRuleGroup
	wafv2:DisassociateWebACL
	wafv2:GenerateMobileSdkReleaseUrl
	wafv2:GetDecryptedAPIKey

サービスプレフィックス	アクション
	wafv2:GetIPSet
	wafv2:GetLoggingConfiguration
	wafv2:GetManagedRuleSet
	wafv2:GetMobileSdkRelease
	wafv2:GetPermissionPolicy
	wafv2:GetRateBasedStatementManagedKeys
	wafv2:GetRegexPatternSet
	wafv2:GetRuleGroup
	wafv2:GetSampledRequests
	wafv2:GetWebACLForResource
	wafv2:ListAPIKeys
	wafv2:ListAvailableManagedRuleGroups
	wafv2:ListAvailableManagedRuleGroupVersions
	wafv2:ListIPSets
	wafv2:ListLoggingConfigurations
	wafv2:ListManagedRuleSets
	wafv2:ListMobileSdkReleases
	wafv2:ListRegexPatternSets
	wafv2:ListResourcesForWebACL
	wafv2:ListRuleGroups
	wafv2:ListWebACLs

サービスプレフィックス	アクション
	wafv2:PutLoggingConfiguration
	wafv2:PutManagedRuleSetVersions
	wafv2:PutPermissionPolicy
	wafv2:UpdateIPSet
	wafv2:UpdateManagedRuleSetVersionExpiryDate
	wafv2:UpdateRegexPatternSet
	wafv2:UpdateRuleGroup
	wafv2:UpdateWebACL

サービスプレフィックス	アクション
wellarchitected	wellarchitected:AssociateLenses
	wellarchitected:AssociateProfiles
	wellarchitected:CreateLensShare
	wellarchitected:CreateLensVersion
	wellarchitected:CreateMilestone
	wellarchitected:CreateProfile
	wellarchitected:CreateProfileShare
	wellarchitected:CreateReviewTemplate
	wellarchitected:CreateWorkload
	wellarchitected:CreateWorkloadShare
	wellarchitected>DeleteLens
	wellarchitected>DeleteLensShare
	wellarchitected>DeleteProfile
	wellarchitected>DeleteProfileShare
	wellarchitected>DeleteReviewTemplate
	wellarchitected>DeleteTemplateShare
	wellarchitected>DeleteWorkload
	wellarchitected>DeleteWorkloadShare
	wellarchitected:DisassociateLenses
	wellarchitected:DisassociateProfiles
wellarchitected:ExportLens	

サービスプレフィックス	アクション
	wellarchitected:GetAnswer
	wellarchitected:GetConsolidatedReport
	wellarchitected:GetGlobalSettings
	wellarchitected:GetLens
	wellarchitected:GetLensReview
	wellarchitected:GetLensReviewReport
	wellarchitected:GetLensVersionDifference
	wellarchitected:GetMilestone
	wellarchitected:GetProfile
	wellarchitected:GetProfileTemplate
	wellarchitected:GetReviewTemplate
	wellarchitected:GetReviewTemplateAnswer
	wellarchitected:GetReviewTemplateLensReview
	wellarchitected:GetWorkload
	wellarchitected:ImportLens
	wellarchitected:ListAnswers
	wellarchitected:ListCheckDetails
	wellarchitected:ListCheckSummaries
	wellarchitected:ListLenses
	wellarchitected:ListLensReviewImprovements
	wellarchitected:ListLensReviews

サービスプレフィックス	アクション
	wellarchitected:ListLensShares
	wellarchitected:ListMilestones
	wellarchitected:ListNotifications
	wellarchitected:ListProfileNotifications
	wellarchitected:ListProfiles
	wellarchitected:ListProfileShares
	wellarchitected:ListReviewTemplateAnswers
	wellarchitected:ListReviewTemplates
	wellarchitected:ListShareInvitations
	wellarchitected:ListTemplateShares
	wellarchitected:ListWorkloads
	wellarchitected:ListWorkloadShares
	wellarchitected:UpdateAnswer
	wellarchitected:UpdateGlobalSettings
	wellarchitected:UpdateIntegration
	wellarchitected:UpdateLensReview
	wellarchitected:UpdateProfile
	wellarchitected:UpdateReviewTemplate
	wellarchitected:UpdateReviewTemplateLensReview
	wellarchitected:UpdateShareInvitation
	wellarchitected:UpdateWorkload

サービスプレフィックス	アクション
	wellarchitected:UpdateWorkloadShare wellarchitected:UpgradeLensReview wellarchitected:UpgradeProfileVersion wellarchitected:UpgradeReviewTemplateLensReview



サービスプレフィックス	アクション
wisdom	wisdom:CreateAssistant
	wisdom:CreateAssistantAssociation
	wisdom:CreateContent
	wisdom:CreateKnowledgeBase
	wisdom:CreateQuickResponse
	wisdom:CreateSession
	wisdom>DeleteAssistant
	wisdom>DeleteAssistantAssociation
	wisdom>DeleteContent
	wisdom>DeleteImportJob
	wisdom>DeleteKnowledgeBase
	wisdom>DeleteQuickResponse
	wisdom:GetAssistant
	wisdom:GetAssistantAssociation
	wisdom:GetContent
	wisdom:GetContentSummary
	wisdom:GetImportJob
	wisdom:GetKnowledgeBase
	wisdom:GetRecommendations
	wisdom:GetSession
	wisdom:ListAssistantAssociations

サービスプレフィックス	アクション
	wisdom:ListAssistants
	wisdom:ListContents
	wisdom:ListImportJobs
	wisdom:ListKnowledgeBases
	wisdom:ListQuickResponses
	wisdom:NotifyRecommendationsReceived
	wisdom:QueryAssistant
	wisdom:RemoveKnowledgeBaseTemplateUri
	wisdom:SearchContent
	wisdom:SearchQuickResponses
	wisdom:SearchSessions
	wisdom:StartContentUpload
	wisdom:StartImportJob
	wisdom:UpdateContent
	wisdom:UpdateKnowledgeBaseTemplateUri
	wisdom:UpdateQuickResponse

サービスプレフィックス	アクション
worklink	worklink:AssociateDomain
	worklink:AssociateWebsiteAuthorizationProvider
	worklink:AssociateWebsiteCertificateAuthority
	worklink:CreateFleet
	worklink>DeleteFleet
	worklink:DescribeAuditStreamConfiguration
	worklink:DescribeCompanyNetworkConfiguration
	worklink:DescribeDevice
	worklink:DescribeDevicePolicyConfiguration
	worklink:DescribeDomain
	worklink:DescribeFleetMetadata
	worklink:DescribeIdentityProviderConfiguration
	worklink:DescribeWebsiteCertificateAuthority
	worklink:DisassociateDomain
	worklink:DisassociateWebsiteAuthorizationProvider
	worklink:DisassociateWebsiteCertificateAuthority
	worklink:ListDevices
	worklink:ListDomains
	worklink:ListFleets
	worklink:ListWebsiteAuthorizationProviders
	worklink:ListWebsiteCertificateAuthorities

サービスプレフィックス	アクション
	<p>worklink:RestoreDomainAccess</p> <p>worklink:RevokeDomainAccess</p> <p>worklink:SignOutUser</p> <p>worklink:UpdateAuditStreamConfiguration</p> <p>worklink:UpdateCompanyNetworkConfiguration</p> <p>worklink:UpdateDevicePolicyConfiguration</p> <p>worklink:UpdateDomainMetadata</p> <p>worklink:UpdateFleetMetadata</p> <p>worklink:UpdateIdentityProviderConfiguration</p>

サービスプレフィックス	アクション
ワークスペース	<code>workspaces:AcceptAccountLinkInvitation</code>
	<code>workspaces:AssociateConnectionAlias</code>
	<code>workspaces:AssociateIpGroups</code>
	<code>workspaces:AssociateWorkspaceApplication</code>
	<code>workspaces:CopyWorkspaceImage</code>
	<code>workspaces:CreateConnectClientAddIn</code>
	<code>workspaces:CreateConnectionAlias</code>
	<code>workspaces:CreateIpGroup</code>
	<code>workspaces:CreateStandbyWorkspaces</code>
	<code>workspaces:CreateUpdatedWorkspaceImage</code>
	<code>workspaces:CreateWorkspaceBundle</code>
	<code>workspaces:CreateWorkspaceImage</code>
	<code>workspaces:CreateWorkspaces</code>
	<code>workspaces&gt;DeleteClientBranding</code>
	<code>workspaces&gt;DeleteConnectClientAddIn</code>
	<code>workspaces&gt;DeleteConnectionAlias</code>
	<code>workspaces&gt;DeleteIpGroup</code>
	<code>workspaces&gt;DeleteWorkspaceBundle</code>
	<code>workspaces&gt;DeleteWorkspaceImage</code>
	<code>workspaces:DeployWorkspaceApplications</code>
	<code>workspaces:DeregisterWorkspaceDirectory</code>

サービスプレフィックス	アクション
	<code>workspaces:DescribeAccount</code>
	<code>workspaces:DescribeAccountModifications</code>
	<code>workspaces:DescribeApplicationAssociations</code>
	<code>workspaces:DescribeApplications</code>
	<code>workspaces:DescribeBundleAssociations</code>
	<code>workspaces:DescribeClientBranding</code>
	<code>workspaces:DescribeClientProperties</code>
	<code>workspaces:DescribeConnectClientAddIns</code>
	<code>workspaces:DescribeConnectionAliases</code>
	<code>workspaces:DescribeConnectionAliasPermissions</code>
	<code>workspaces:DescribeImageAssociations</code>
	<code>workspaces:DescribeIpGroups</code>
	<code>workspaces:DescribeWorkspaceAssociations</code>
	<code>workspaces:DescribeWorkspaceBundles</code>
	<code>workspaces:DescribeWorkspaceDirectories</code>
	<code>workspaces:DescribeWorkspaceImagePermissions</code>
	<code>workspaces:DescribeWorkspaces</code>
	<code>workspaces:DescribeWorkspacesConnectionStatus</code>
	<code>workspaces:DescribeWorkspaceSnapshots</code>
	<code>workspaces:DisassociateConnectionAlias</code>
	<code>workspaces:DisassociateIpGroups</code>

サービスプレフィックス	アクション
	<code>workspaces:DisassociateWorkspaceApplication</code>
	<code>workspaces:GetAccountLink</code>
	<code>workspaces:ImportClientBranding</code>
	<code>workspaces:ImportWorkspaceImage</code>
	<code>workspaces:ListAccountLinks</code>
	<code>workspaces:ListAvailableManagementCidrRanges</code>
	<code>workspaces:MigrateWorkspace</code>
	<code>workspaces:ModifyAccount</code>
	<code>workspaces:ModifyCertificateBasedAuthProperties</code>
	<code>workspaces:ModifyClientProperties</code>
	<code>workspaces:ModifySamlProperties</code>
	<code>workspaces:ModifySelfservicePermissions</code>
	<code>workspaces:ModifyWorkspaceAccessProperties</code>
	<code>workspaces:ModifyWorkspaceCreationProperties</code>
	<code>workspaces:ModifyWorkspaceProperties</code>
	<code>workspaces:ModifyWorkspaceState</code>
	<code>workspaces:RebootWorkspaces</code>
	<code>workspaces:RebuildWorkspaces</code>
	<code>workspaces:RegisterWorkspaceDirectory</code>
	<code>workspaces:RejectAccountLinkInvitation</code>
	<code>workspaces:RestoreWorkspace</code>

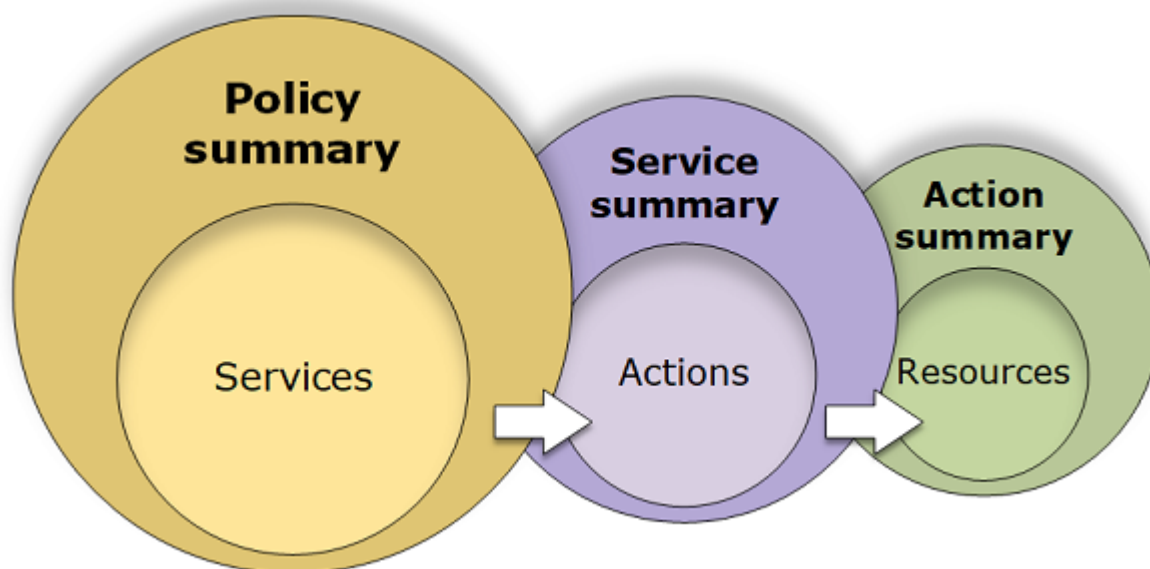
サービスプレフィックス	アクション
	workspaces:StartWorkspaces
	workspaces:StopWorkspaces
	workspaces:TerminateWorkspaces
	workspaces:UpdateConnectClientAddIn
	workspaces:UpdateConnectionAliasPermission
	workspaces:UpdateWorkspaceBundle
	workspaces:UpdateWorkspaceImagePermission



サービスプレフィックス	アクション
xray	xray:CreateGroup
	xray:CreateSamplingRule
	xray>DeleteGroup
	xray>DeleteResourcePolicy
	xray>DeleteSamplingRule
	xray:GetEncryptionConfig
	xray:GetGroup
	xray:GetGroups
	xray:GetInsight
	xray:GetInsightEvents
	xray:GetInsightImpactGraph
	xray:GetInsightSummaries
	xray:GetSamplingRules
	xray:ListResourcePolicies
	xray:PutEncryptionConfig
	xray:PutResourcePolicy
	xray:UpdateGroup
	xray:UpdateSamplingRule

## ポリシーによって付与されるアクセス許可について

IAM コンソールには、ポリシー内の各サービスに対して許可または拒否されるアクセスレベル、リソース、条件を定義するポリシー概要テーブルが含まれます。ポリシーは、[ポリシー概要](#)、[サービス概要](#)、[アクション概要](#)の3つのテーブルにまとめられています。ポリシー概要テーブルには、サービスのリストが含まれます。そのリストからサービスを選択して、サービス概要を表示します。この概要テーブルには、選択したサービスに対して定義されているアクションとその関連するアクセス権限のリストが含まれます。そのテーブルからアクションを選択して、アクション概要を表示できます。このテーブルには、選択したアクションのリソースと条件のリストが含まれます。



そのユーザーにアタッチされているすべてのポリシー (管理およびインライン) について、[ユーザー] ページまたは [ロール] ページでポリシー概要を表示できます。すべての管理ポリシーについて、[ポリシー] ページで概要を表示します。管理ポリシーには、AWS 管理ポリシー、AWS 管理ジョブ機能ポリシー、カスタマー管理ポリシーが含まれます。これらのポリシーの概要は、ユーザーまたは他の IAM ID にアタッチされているかどうかにかかわらず、[Policies] ページで表示できます。

ポリシー概要の情報を使用して、ポリシーによって許可または拒否されるアクセス権限を理解できます。ポリシー概要は、想定するアクセス許可を付与しないポリシーの[トラブルシューティング](#)に役立ちます。

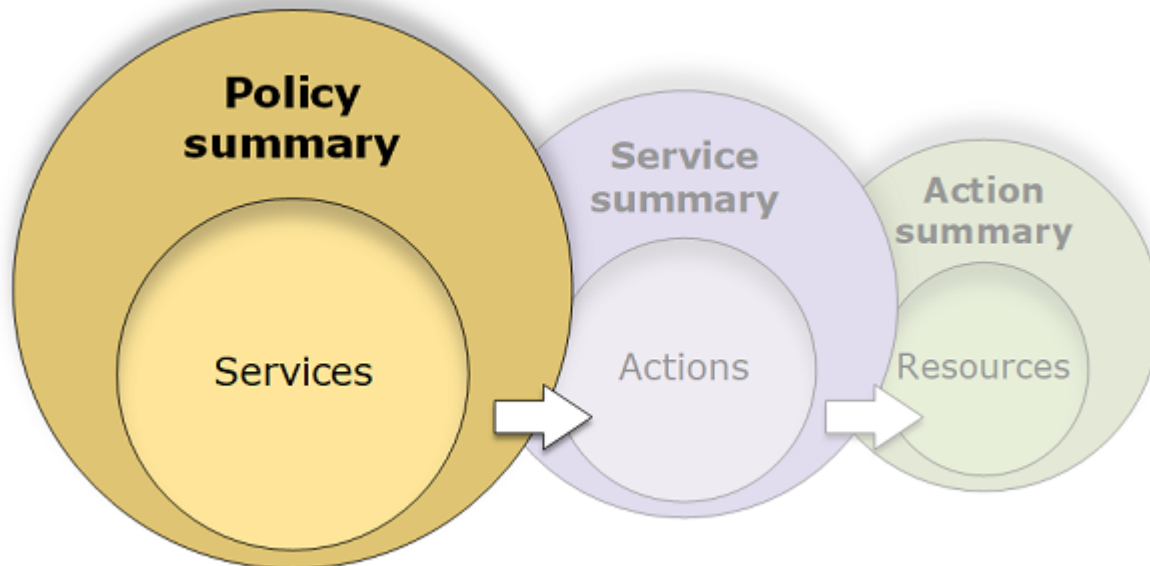
### トピック

- [ポリシー概要 \(サービスの一覧\)](#)
- [サービス概要 \(アクションのリスト\)](#)
- [アクション概要 \(リソースのリスト\)](#)

- [ポリシー概要の例](#)

## ポリシー概要 (サービスの一覧)

ポリシーは、ポリシー概要、[サービス概要](#)、[アクション概要](#)の3つのテーブルにまとめられています。ポリシー概要テーブルには、選択したポリシーによって定義されているサービスとアクセス許可の概要のリストが含まれます。



ポリシー概要テーブルは、[Uncategorized services (未分類サービス)]、[Explicit deny (明示的な拒否)]、[許可] という1つ以上のセクションにグループ化されます。IAMによって認識されないサービスがポリシーに含まれる場合、そのサービスはテーブルの [Uncategorized services (未分類サービス)] セクションに含まれます。IAMによって認識されるサービスがポリシーに含まれる場合、ポリシーの効果 (Deny または Allow) に応じて、そのサービスはテーブルの [Explicit deny (明示的な拒否)] セクションまたは [許可] セクションに含まれます。

### ポリシーの概要の表示

[ユーザーの詳細] 詳細ページの [アクセス許可] タブでポリシー名を選択することで、ユーザーにアタッチされているポリシーの概要を表示できます。[ロールの詳細] 詳細ページの [アクセス許可] タブでポリシー名を選択することで、ロールにアタッチされているポリシーの概要を表示できます。[ポリシー] ページで、管理ポリシーのポリシー概要を表示できます。ポリシー概要がポリシーに含まれていない場合、その理由については、「[欠落しているポリシーの概要](#)」を参照してください。

[ポリシー] ページのポリシー概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、表示するポリシーの名前を選択します。
4. ポリシーの [ポリシー詳細] ページで [アクセス許可] タブを表示して、ポリシー概要を確認します。

ユーザーにアタッチされているポリシーの概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択します。
3. ユーザーのリストから、ポリシーを表示するユーザーを選択します。
4. ユーザーの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ユーザーに直接、またはグループのユーザーにアタッチされているポリシーのリストを表示します。
5. ユーザーのポリシーのテーブルで、表示するポリシーの行を展開します。

ロールにアタッチされているポリシーの概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. ロールのリストから、表示するポリシーを含むロールの名前を選択します。
4. ロールの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ロールにアタッチされているポリシーのリストを表示します。
5. ロールのポリシーのテーブルで、表示するポリシーの行を展開します。

## ポリシーの編集による警告の編集

ポリシー概要の表示中に、タイプミスを見つけたり、必要なアクセス許可がポリシーから付与されないことに気付いたりすることがあります。ポリシー概要を直接編集することはできません。ただし、ビジュアルポリシーエディタを使用してお客様のマネージドポリシーを編集できます。これに

より、ポリシー概要レポートで報告されるのと同じエラーや警告の多くをキャッチできます。次に、ポリシー概要の変更を表示して、これらのすべての問題を修正したことを確認できます。インラインポリシーを編集する方法については、「[the section called “IAM ポリシーの編集”](#)」を参照してください。AWS の管理ポリシーを編集することはできません。

[Visual] オプションを使用してポリシー概要のポリシーを編集するには

1. 前の手順で説明されているように、ポリシー概要を開きます。
2. [編集] を選択します。

[ユーザー] ページが表示された状態で、ユーザーにアタッチされているカスタマー管理ポリシーを編集しようとする、[ポリシー] ページにリダイレクトされます。カスタマー管理ポリシーは、[ポリシー] ページでのみ編集できます。

3. [Visual] オプションを選択して、ポリシーの編集可能な視覚表現を表示します。IAM はポリシーを再構成してビジュアルエディタに合わせて最適化し、問題を見つけて修正しやすくする場合があります。ページの警告とエラーメッセージは、ポリシーの問題を修正するために役立ちます。IAM によるポリシーの再構築の詳細については、「[ポリシーの再構成](#)」を参照してください。
4. ポリシー概要に反映された変更を表示するには、ポリシーを編集し、[次へ] を選択します。それでも問題が表示される場合は、[Previous (前に戻る)] を選択して編集画面に戻ります。
5. [変更を保存] を選択して、変更を保存します。

[JSON] オプションを使用してポリシー概要のポリシーを編集するには

1. 前の手順で説明されているように、ポリシー概要を開きます。
2. [概要] と [JSON] ボタンを使用して、ポリシー概要と JSON ポリシードキュメントを比較します。この情報を使用して、ポリシードキュメントで変更する行を確認できます。
3. [編集] を選択し、[JSON] オプションを選択して、JSON ポリシードキュメントを編集します。

#### Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで変更を行うか [次へ] を押した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

[ユーザー] ページが表示された状態で、ユーザーにアタッチされているカスタマー管理ポリシーを編集しようとする、[ポリシー] ページにリダイレクトされます。カスタマー管理ポリシーは、[ポリシー] ページでのみ編集できます。

4. ポリシーを編集します。[ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。それでも問題が表示される場合は、[Previous (前に戻る)] を選択して編集画面に戻ります。
5. [変更を保存] を選択して、変更を保存します。

## ポリシー概要の要素について

次のポリシー詳細ページの例で、[SummaryAllElements] ポリシーは、ユーザーに直接アタッチされている管理ポリシー (カスタマー管理ポリシー) です。ポリシー概要を表示するには、このポリシーを展開します。

Policy details

Type Customer managed	Creation time September 13, 2022, 16:37 (UTC-05:00)	Edited time September 13, 2022, 16:40 (UTC-05:00)	ARN arn:aws:iam::[redacted]:policy/SummaryAllElements
--------------------------	--	--	--

1 Permissions Entitles attached Tags Policy versions Access Advisor

2 This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose [Show remaining](#). [Learn more](#)

3 Edit Summary JSON

4 Search

5 Explicit deny (1 of 338 services)

Service	Access level	Resource	Request condition
S3	Limited: List, Permissions management, Read, Write, Tagging	Multiple	None

Show remaining 334 services

Allow (3 of 338 services)

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp   IP Address   203.0.113.0/24
CodeDeploy	Limited: List, Read, Write, Tagging	DeploymentGroupName   string like   All, region   string like   us-west-2	None
EC2	Limited: Read	All resources	None

前のイメージで、ポリシー概要は、[ポリシー] ページ内に表示されます。

1. [アクセス許可] タブには、ポリシーで定義されたアクセス許可が含まれています。
2. ポリシーに定義されているすべてのアクション、リソース、条件に対し、ポリシーで権限が付与されない場合、警告またはエラーバナーがページの上部に表示されます。すると、問題に関する詳細がポリシー概要に含まれます。ポリシーで付与されるアクセス許可について理解し、問題の解決にポリシー概要をどのように役立てるかについては、「[the section called “使用するポリシーが予期するアクセス許可を付与しない”](#)」を参照してください。

3. [概要] と [JSON] ボタンを使用して、ポリシー概要と JSON ポリシードキュメントを切り替えま  
す。
4. [検索] ボックスを使用してサービスのリストを縮小すると、特定のサービスを簡単に検索できま  
す。
5. 展開されたビューには、[SummaryAllElements] ポリシーの追加の詳細が表示されます。

以下のポリシー概要テーブルの図は、[ポリシー詳細] ページで展開した [SummaryAllElements] ポリ  
シーを示しています。

Explicit deny (1 of 338 services) <span style="color: red;">A</span>			
Service <span style="color: red;">B</span>	Access level <span style="color: red;">C</span>	Resource <span style="color: red;">D</span>	Request condition <span style="color: red;">E</span>
S3	Limited: List, Permissions management, Read, Write, Tagging	Multiple	None

Allow (3 of 338 services) <span style="color: red;">F</span> <input type="checkbox"/> Show remaining 334 services			
Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp   IP Address   203.0.113.0/24
CodeDeploy	Limited: List, Read, Write, Tagging	DeploymentGroupName   string like   All, region   string like   us-west-2	None
EC2	Limited: Read	All resources	None

前のイメージで、ポリシー概要は、[ポリシー] ページ内に表示されます。

- A. IAM によって認識されるサービスは、ポリシーによってそのサービスの使用が許可されるか明示的に拒否されるかに応じて、グループ化されます。この例のポリシーには、Amazon S3 サービスに関する Deny ステートメントと、請求、CodeDeploy、Amazon EC2 サービスに関する Allow ステートメントが含まれています。
- B. [Service (サービス)] – この列には、ポリシー内で定義されているサービス一覧と各サービスの詳細が表示されています。ポリシー概要テーブルの各サービス名は、[service summary (サービス概要)] テーブルへのリンクです。詳細については、「[サービス概要 \(アクションのリスト\)](#)」を参照してください。この例のアクセス許可は、Amazon S3、請求、CodeDeploy、Amazon EC2 サービス向けに定義されています。
- C. [アクセスレベル] – この列には、各アクセスレベル (List、Read、Write、Permission Management、および Tagging) のアクションに、ポリシーで定義されている Full または Limited アクセス許可があるかどうかが表示されます。アクセスレベルの概要の詳細と例については、「[ポリシー概要内のアクセスレベルの概要](#)」を参照してください。



- フルアクセス – このエントリは、そのサービスに使用可能な 4 つの全アクセスレベルのすべてのアクションへのアクセスが許可されていることを示します。
- エントリに [フルアクセス] が含まれていない場合、一部のサービスは利用できますが、すべてのアクションを利用できるわけではありません。このアクセスは、各アクセスレベル (List、Read、Write、Permission Management、および Tagging) の説明のとおり定義されます。

[フル]: このポリシーでは、表示されている各アクセスレベル分類のすべてのアクションへのアクセスを許可します。この例では、ポリシーによって、すべての請求アクションに Read 権限が付与されます。

Limited (制限あり): このポリシーでは、表示されている各アクセスレベル分類の 1 つ以上のアクションへのアクセスを許可しますが、すべてのアクションへのアクセスを許可するわけではありません。この例では、ポリシーによって、一部の請求アクションに Write 権限が付与されます。

D. Resource (リソース) – この列には、各サービスに対してポリシーで指定したリソースが表示されます。

- 複数 – このポリシーには、サービス内に複数のリソースが含まれていますが、すべてのリソースが含まれているわけではありません。この例では、複数の Amazon S3 リソースへのアクセスは明示的に拒否されます。
- すべてのリソース – このポリシーは、サービス内のすべてのリソースに対して定義されています。この例では、ポリシーによって、一覧表示されたアクションをすべての請求リソースで行うことができますようになります。
- リソーステキスト – このポリシーには、1 つのリソースがサービス内に含まれます。この例では、DeploymentGroupName CodeDeploy リソースに対してのみ、表示されているアクションを行うことができます。サービスから IAM に渡される情報に応じて、ARN が表示されるか、定義されたリソースタイプが表示される場合があります。

#### Note

この列には、別のサービスのリソースが含まれることがあります。リソースを含むポリシーステートメントで、アクションとリソースの両方がサービスに一致しない場合、ポリシーには不一致のリソースが含まれます。IAM は、ポリシーを作成するときや、ポリシー概要でポリシーを表示するとき、リソースの不一致のリソースについて警告されません。この列に不一致のリソースが含まれる場合は、ポリシーのエラーを確認する必要



があります。ポリシーをより深く理解するために、必ず [Policy Simulator](#) でテストしてください。

E. Request condition (リクエストの条件) – この列では、リソースに関連付けられたサービスまたはアクションが条件の対象かどうかを示します。

- None (なし) – このポリシーには、サービスの条件は含まれません。この例では、Amazon S3 サービスで拒否されたアクションに条件は適用されません。
- 条件のテキスト – このポリシーには、サービスに対する 1 つの条件が含まれます。この例では、表示されている [請求] アクションは、ソースの IP アドレスが 203.0.113.0/24 と一致する場合にのみ許可されます。
- 複数 – このポリシーでは、サービスに対する複数の条件が含まれます。ポリシーの複数の条件のそれぞれを表示するには、[JSON] を選択してポリシードキュメントを表示します。

F. 残りのサービスを表示 – このボタンを選択すると、テーブルが展開されて、ポリシーで定義されていないサービスが含まれます。これらのサービスは、このポリシー内で暗黙的に拒否 (またはデフォルトで拒否) されています。ただし、別のポリシーのステートメントでは、そのサービスの使用が許可されているか明示的に拒否されている可能性があります。ポリシー概要には、1 つのポリシーのアクセス許可がまとめられています。特定のリクエストを許可するか拒否するかを AWS サービスがどのようにして決定しているかについては、「[ポリシーの評価論理](#)」を参照してください。

ポリシー内のポリシーまたは要素にアクセス許可を与えない場合、IAM にはポリシー概要に関する追加の警告と情報が表示されます。以下のポリシー概要テーブルは、SummaryAllElements ポリシー詳細ページの [残りのサービスを表示] サービスを拡張し、想定される警告と共に示しています。

Explicit deny (1 of 338 services)			
Service	Access level	Resource <b>a</b>	Request condition <b>b</b>
S3	Limited: List, Permissions management, Read, Write, Tagging	<b>c</b> Multiple <b>a</b> One or more actions do not have an applicable resource.	None

Allow (3 of 338 services) <span style="float: right;">Show remaining 334 services</span>			
Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp   IP Address   203.0.113.0/24
CodeCommit	None	<b>d</b> <b>a</b> No resources are defined.	None
CodeDeploy	Limited: List, Read, Write, Tagging	<b>e</b> DeploymentGroupName   string like   All, region   string like   us-west-2 <b>a</b> One or more actions do not have an applicable resource.	None
EC2	Limited: Read	All resources	None
S3	None	None <b>a</b> One or more actions do not have an applicable resource.	<b>f</b> None <b>a</b> One or more conditions do not have an applicable action.

前のイメージには、アクセス許可のない定義済みアクション、リソース、または条件を含むすべてのサービスを示しています。

a. Resource warnings (リソースの警告) – すべてのアクションまたはリソースに対するアクセス許可がないサービスについては、以下のいずれかの警告がテーブルの [Resource (リソース)] 列に表示されます。



リソースが定義されていません。 – これは、サービスではアクションが定義されていますが、ポリシーにはサポートされるリソースが含まれていないことを意味します。



1つ以上のアクションには該当するリソースがありません。 – これは、サービスではアクションが定義されていますが、アクションの一部にサポートされているリソースがないことを意味します。



1つ以上のリソースには該当するアクションがありません。 – これは、サービスではリソースが定義されていますが、一部のリソースにサポートされているアクションがないことを意味します。

サービスに、該当するリソースがないアクションと、該当するリソースがあるリソースの両方が含まれている場合、「1つ以上のリソースに該当するアクションがありません」という警告が表示されます。これは、サービスの概要を表示する際に、アクションに適用されないリソースが表示されないために発生します。ListAllMyBuckets アクションの場合、このポリシーには最後の警告が含まれます。アクションがリソースレベルのアクセス許可をサポートせず、s3:x-amz-acl 条件キーをサポートしていないためです。リソースまたは条件に関する問題が解決すると、残りの問題が警告の詳細に表示されます。

b. リクエスト条件の警告 – すべての条件に対するアクセス許可がないサービスについては、以下のいずれかの警告がテーブルの [Request condition (リクエストの条件)] 列に表示されます。



1つ以上のアクションには該当する条件がありません。 – これは、サービスではアクションが定義されていますが、一部のアクションにはサポートされている条件がないことを意味します。



1つ以上の条件には該当するアクションがありません。 – これは、サービスでは条件が定義されていますが、一部の条件にはサポートされているアクションがないことを意味します。

c. Multiple |



1つ以上のアクションには該当するリソースがありません。 – Amazon S3 に対する Deny ステートメントで、複数のリソースが含まれます。また、複数のアクションも含まれ、一部のアクションはリソースをサポートしていますが、他のアクションはサポートしていません。このポリシーを表示するには、[the section called “SummaryAllElements JSON ポリシードキュメント”](#) を参照してください。この場合、ポリシーにはすべての Amazon S3 アクションが含まれ、バケットまたはバケット内のオブジェクトで実行可能なアクションのみが拒否されます。

d.

リソースは定義されていません – Tサービスはアクションを定義済みですが、サポートされるリソースはポリシーに含まれていないため、サービスはアクセス許可を付与しません。この場合、ポリシーには CodeCommit アクションが含まれますが、CodeCommit リソースは含まれません。

e. DeploymentGroupName | のような文字列 | すべて、リージョン | のような文字列 | us-west-2



1つまたは複数のアクションに該当するリソースがありません。 – サービスには定義されたアクションと、サポートするリソースを持たないアクションが少なくとも1つあります。

## f. なし |



1 つ以上の条件に該当するアクションがありません。 – サービスに、サポートするアクションのない条件キーが 1 つ以上あります。

## SummaryAllElements JSON ポリシードキュメント

SummaryAllElements ポリシーは、アカウントのアクセス許可を定義するために使用するものではありません。むしろ、ポリシー概要を表示している間に発生する可能性のあるエラーや警告について説明することが目的です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "billing:Get*",
        "payments:List*",
        "payments:Update*",
        "account:Get*",
        "account:List*",
        "cur:GetUsage*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "203.0.113.0/24"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::customer",
        "arn:aws:s3:::customer/*"
      ]
    }
  ]
}
```

```
    ],
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:GetConsoleScreenshots"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "codedploy:*",
      "codecommit:*"
    ],
    "Resource": [
      "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
      "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListAllMyBuckets",
      "s3:GetObject",
      "s3:DeleteObject",
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": [
      "arn:aws:s3:::developer_bucket",
      "arn:aws:s3:::developer_bucket/*",
      "arn:aws:autoscaling:us-east-2:123456789012:autoscalgrp"
    ],
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": [
          "public-read"
        ],
      },
      "s3:prefix": [
        "custom",
        "other"
      ]
    }
  }
}
```



サービス	アクセスレベル	このポリシーでは、以下を提供します。
		Permissions management アクションにはアクセスできません。
EC2	フル: List、Read 制限: Write	すべての Amazon EC2、List および Read アクションにアクセスでき、さらに Amazon EC2 Write アクションの少なくとも 1 つのアクションにアクセスできますが、すべてにはアクセスできません。また Permissions management アクセスレベル分類のアクションにはアクセスできません。
S3	Limited: Read、Write、Permissions management	少なくとも 1 つのアクセス許可がありますが、Amazon S3 Read の Write、および Permissions management アクションのすべてではありません。
CodeDeploy	(空)	IAM によってこのサービスが認識されないため、不明なアクセスです。
API Gateway	なし	ポリシーにアクセス許可が定義されていません。
CodeBuild	 アクションは定義されていません。	サービスにアクションが定義されていないためアクセス許可がありません。この問題を理解して解決する方法については、「 <a href="#">the section called “使用するポリシーが予期するアクセス許可を付与しない”</a> 」を参照してください。

[前述](#)のように、フルアクセスは、ポリシーによりサービス内のすべてのアクションに対するアクセス許可が付与されることを示します。サービス内の一部のアクションへアクセスを提供するポリシーは、アクセスレベルの分類に従ってさらにグループ化されます。これは、以下のアクセスレベルのグループ化の 1 つによって示されます。

- Full: このポリシーは、指定されたアクセスレベル分類のすべてのアクションへのアクセスを許可します。

- Limited: このポリシーは、指定されたアクセスレベル分類内の 1 つ以上のアクションへのアクセスを許可しますが、すべてのアクションへのアクセスを同時には許可しません。
- None: このポリシーはいずれのアクセス許可も付与しません。
- (空): IAM はこのサービスを認識しません。サービス名にタイプミスが含まれる場合、ポリシーによってサービスへのアクセスは許可されません。サービス名が正しい場合、サービスがポリシー概要をサポートしていないか、プレビュー中である可能性があります。この場合は、ポリシーによってアクセスが許可される可能性があります。この場合は、ポリシーによってアクセスが許可される可能性がありますが、そのアクセスがポリシー概要に表示されることはありません。一般公開された (GA) サービスに対するポリシー概要のサポートをリクエストするには、「[サービスが IAM ポリシー概要をサポートしていない](#)」を参照してください。

アクションへの制限付き (部分) アクセスを含むアクセスレベル概要は、AWS のアクセスレベル分類 List、Read、Tagging、Write、または Permissions management を使用してグループ化されます。

## AWS アクセスレベル

AWS は、サービスのアクションについて以下のアクセスレベル分類を定義します。

- List: オブジェクトが存在するかどうかを判断するためにサービス内のリソースを一覧表示するアクセス許可。このレベルのアクセス権を持つアクションはオブジェクトをリストできますが、リソースのコンテンツは表示されません。たとえば、Amazon S3 アクション ListBucket には List アクセスレベルがあります。
- Read: サービス内のリソースのコンテンツと属性を読み取るアクセス許可。ただし、編集するアクセス許可はありません。たとえば、Amazon S3 アクション GetObject および GetBucketLocation には、読み取りアクセスレベルがあります。
- Tagging: リソースタグの状態のみを変更するアクションを実行する権限。たとえば、IAM のアクション TagRole および UntagRole は、ロールのタグ付けまたはタグ付け解除のみを許可するため、タグ付けアクセスレベルがあります。ただし、CreateRole アクションは、ロール作成時にそのロールリソースのタグ付けを許可します。このアクションはタグの追加にとどまらないため、このアクションには Write アクセスレベルがあります。
- Write: サービス内のリソースを作成、削除、または変更するアクセス許可。たとえば、Amazon S3 アクション DeleteBucket、CreateBucket、および PutObject には Write アクセスレベルがあります。また、Write アクションにより、リソースタグの変更も許可される場合があります。ただし、タグへの変更のみを許可するアクションには Tagging アクセスレベルがあります。
- Permissions management: サービスのリソースに対するアクセス許可を付与または変更するアクセス許可。たとえば、IAM や AWS Organizations のほとんどのアクション、Amazon S3



の PutBucketPolicy や DeleteBucketPolicy のようなアクションには、Permissions management (アクセス許可管理) アクセスレベルがあります。

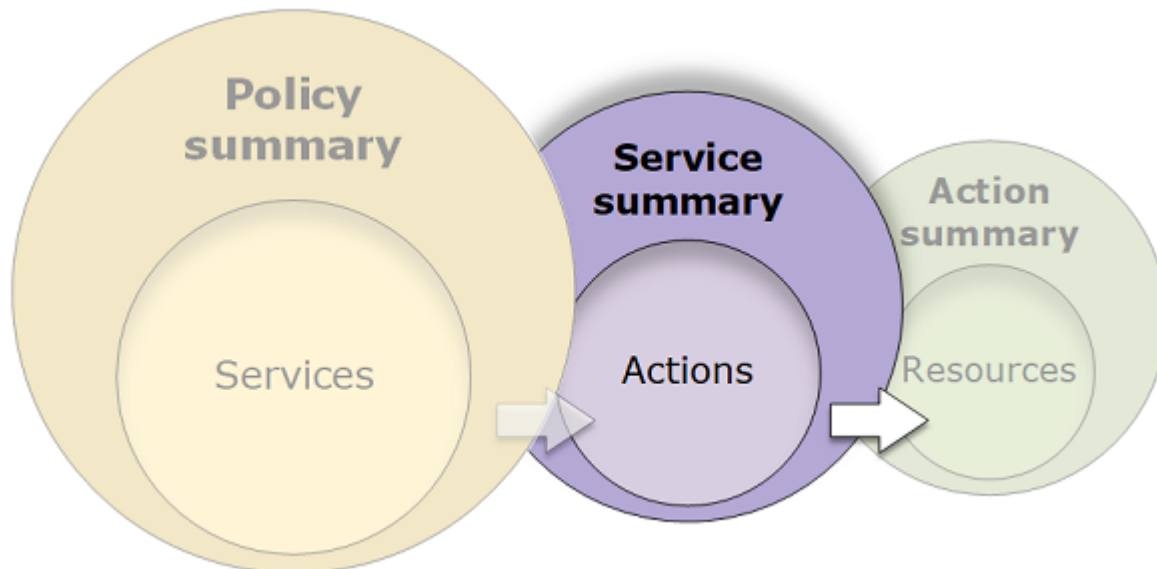
### 📘 ヒント

AWS アカウント のセキュリティを強化するには、[Permissions management] (アクセス許可の管理) アクセスレベル分類を含むポリシーを制限したり定期的にモニタリングしたりします。

サービス内の各アクションに割り当てられているアクセスレベルの分類を表示するには、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## サービス概要 (アクションのリスト)

ポリシーは、ポリシー概要、[サービス概要](#)、[アクション概要](#)の 3 つのテーブルにまとめられています。サービス概要テーブルには、選択したサービスのポリシーによって定義されているアクションとアクセス許可の概要のリストが含まれます。



アクセス許可を付与するポリシー概要に示されている各サービスの概要を表示できます。テーブルは、[Uncategorized actions (未分類アクション)]、[Uncategorized resource types (未分類リソースタイプ)]、およびアクセスレベルのセクションにグループ化されます。IAM によって認識されないアクションがポリシーに含まれる場合、そのアクションはテーブルの [Uncategorized actions (未分類アクション)] セクションに含まれます。IAM によって認識されるアクションがポリシーに含まれる場合、そのアクションはテーブルのいずれかのアクセスレベル (List、Read、Write、Permissions

management) のセクションに含まれます。サービス内の各アクションに割り当てられているアクセスレベルの分類を表示するには、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## サービス概要の表示

[ポリシー] ページで、管理ポリシーのサービス概要を表示できます。

管理ポリシーのサービス概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、表示するポリシーの名前を選択します。
4. ポリシーの [ポリシー詳細] ページで [アクセス許可] タブを表示して、ポリシー概要を確認します。
5. サービスのポリシー概要のリストで、表示するサービス名を選択します。

ユーザーにアタッチされたポリシー概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. ユーザーのリストから、ポリシーを表示するユーザーを選択します。
4. ユーザーの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ユーザーに直接、またはグループのユーザーにアタッチされているポリシーのリストを表示します。
5. ユーザーのポリシーのテーブルで、表示するポリシーの名前を選択します。

[ユーザー] ページが表示された状態で、ユーザーにアタッチされているポリシーのサービス概要を選択すると、[ポリシー] ページにリダイレクトされます。サービスの概要は、[ポリシー] ページでのみ表示できます。

6. [概要] を選択します。サービスのポリシー概要のリストで、表示するサービス名を選択します。

### Note

選択したポリシーがユーザーに直接アタッチされているインラインポリシーの場合、サービス概要テーブルが表示されます。ポリシーがグループからアタッチされたインラ

インポリシーの場合、そのグループの JSON ポリシードキュメントに移動します。ポリシーが管理ポリシーの場合、そのポリシーのサービス概要が [ポリシー] ページに表示されます。

ロールにアタッチされたポリシーのサービス概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ロール] を選択します。
3. ロールのリストから、表示するポリシーを含むロールの名前を選択します。
4. ロールの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ロールにアタッチされているポリシーのリストを表示します。
5. ユーザーのポリシーのテーブルで、表示するポリシーの名前を選択します。

[ロール] ページが表示された状態で、ユーザーにアタッチされているポリシーのサービス概要を選択すると、[ポリシー] ページにリダイレクトされます。サービスの概要は、[ポリシー] ページでのみ表示できます。

6. サービスのポリシー概要のリストで、表示するサービス名を選択します。

## サービス概要の要素を理解する

以下の例は、ポリシー概要から許可されている Amazon S3 アクションのサービス概要です。このサービスのアクションは、アクセスレベルごとにグループ化されています。例えば、サービスで使用可能な合計 52 の読み取りアクションから 35 の読み取りアクションが定義されます。

Permissions

Entities attached

Tags

Policy versions

Access Advisor

**i** This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining**. [Learn more](#)

### Permissions defined in this policy [Info](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Edit

Summary

JSON

< Services Actions in S3 (82 of 128)

Read (35 of 52)

 Show remaining 46 actions

Action

Resource

Request condition

DescribeJob (No access)

**!** This action does not have an applicable resource.

None

DescribeMultiRegionAccessPointOperation (No access)

**!** This action does not have an applicable resource.

None

GetAccelerateConfiguration

BucketName | string like | customer

None

GetAccessPoint (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointConfigurationForObjectLambda (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointForObjectLambda (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointPolicy (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointPolicyForObjectLambda (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointPolicyStatus (No access)

**!** This action does not have an applicable resource.

None

GetAccessPointPolicyStatusForObjectLambda (No access)

**!** This action does not have an applicable resource.

None

GetAccountPublicAccessBlock (No access)

**!** This action does not have an applicable resource.

None

GetAnalyticsConfiguration

BucketName | string like | customer

None

GetBucketAcl

BucketName | string like | customer

None

管理ポリシーのサービス概要ページには、以下の情報が含まれます。

1. ポリシーのサービスに定義されているすべてのアクション、リソース、条件に対し、ポリシーでアクセス許可が付与されない場合、警告バナーがページの上部に表示されます。すると、問題に

- 関する詳細がサービス概要に含まれます。ポリシーで付与されるアクセス許可について理解し、問題の解決にポリシー概要をどのように役立てるかについては、「[the section called “使用するポリシーが予期するアクセス許可を付与しない”](#)」を参照してください。
- [JSON] を選択すると、ポリシーに関する追加の詳細が表示されます。この操作により、アクションに適用されるすべての条件を表示できます。(ユーザーに直接アタッチされているインラインポリシーのサービス概要を表示している場合は、サービス概要ダイアログボックスを閉じてポリシー概要に戻り、JSON ポリシードキュメントにアクセスする必要があります。)
  - 特定のアクションの概要を表示するには、キーワードを [検索] ボックスに入力して、使用可能なアクションのリストを減らします。
  - [サービス] 戻る矢印の隣に、サービスの名前が表示されます (この場合は S3)。このサービスのサービス概要には、ポリシーで定義されている許可または拒否されたアクションのリストが含まれています。サービスが [アクセス許可] タブの [(Explicit deny)] の下に表示されている場合、サービスの概要テーブルに記載されているアクションは明示的に拒否されます。サービスが [アクセス許可] タブの [許可] の下に表示されている場合、サービスの概要テーブルに記載されているアクションは許可されます。
  - アクション – この列には、ポリシー内で定義されたアクションが一覧表示され、各アクションのリソースと条件が示されます。ポリシーでアクションにアクセス許可が付与または拒否されると、アクション名が [\[アクション概要\]](#) テーブルにリンクされます。このテーブルでは、それらのアクションが、ポリシーによるアクセスレベル (許可または拒否) に応じて、1 ~ 5 のセクションに分類されます。それらのセクションは [List]、[Read]、[Write]、[Permission Management]、[Tagging] です。カウントは、それぞれのアクセスレベルでアクセス許可を付与する認識されたアクションの数を示します。合計は、サービスの既知のアクションの数です。この例では、合計 52 の既知の Amazon S3 読み取りアクションのうちの 35 のアクションがアクセス許可を付与します。サービス内の各アクションに割り当てられているアクセスレベルの分類を表示するには、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。
  - [残りのアクションを表示] – このボタンをクリックするとテーブルが展開されるか非表示になり、このサービスで既知ではあるがアクセス許可を指定しないアクションが表示されます。ボタンを展開すると、アクセス許可を指定しないすべての要素の警告も表示されます。
  - [Resource (リソース)] – この列には、ポリシーによってサービスに対して定義されているリソースが表示されます。IAM では、リソースが各アクションに適用されるかどうかは確認されません。この例では、Amazon S3 サービスのアクションは developer\_bucket Amazon S3 バケットリソースに対してのみ許可されます。サービスから IAM に渡される情報に応じて、arn:aws:s3:::developer\_bucket/\* などの ARN が表示されるか、BucketName = developer\_bucket などの定義されたリソースタイプが表示される場合があります。

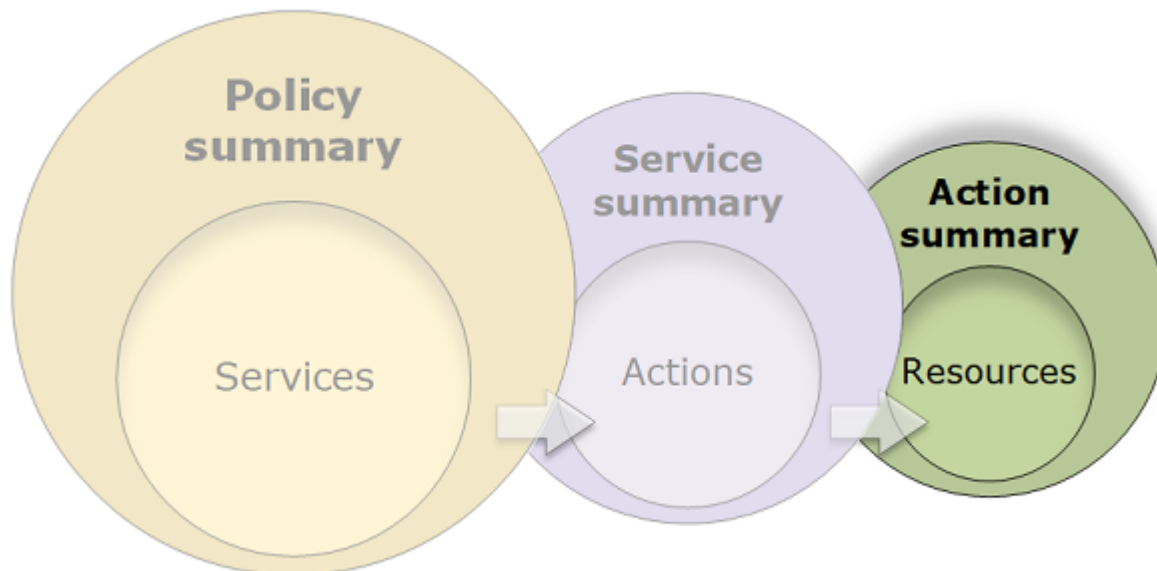
**Note**

この列には、別のサービスのリソースが含まれることがあります。リソースを含むポリシーステートメントで、アクションとリソースの両方がサービスに一致しない場合、ポリシーには不一致のリソースが含まれます。IAM は、ポリシーを作成するときや、サービス概要でポリシーを表示するとき、リソースの不一致のリソースについて警告されません。IAM は、アクションがリソースに適用されるかどうかを示せず、サービスが一致するかどうかだけを示します。この列に不一致のリソースが含まれる場合は、ポリシーのエラーを確認する必要があります。ポリシーをより深く理解するために、必ず [Policy Simulator](#) でテストしてください。

8. Request condition (リクエストの状態) – この列は、リソースに関連付けられたアクションが条件の対象かどうかを示します。これらの条件の詳細については、[JSON] を選択して JSON ポリシードキュメントを確認してください。
9. No access (アクセスなし) – このポリシーにはアクセス許可が指定されないアクションが含まれています。
10. Resource warning (リソースの警告) – フルアクセスが指定されないリソースに対するアクションについては、以下のいずれかの警告が表示されます。
  - [This action does not support resource-level permissions.] (このアクションでは、リソースレベルのアクセス許可はサポートされません。) [This requires a wildcard (\*) for the resource.] (これにはリソース用のワイルドカード (\*) が必要です。) – これは、ポリシーにはリソースレベルのアクセス許可が含まれているが、このアクションに対するアクセス許可を指定するには、"Resource": ["\*"] を含める必要があることを意味しています。
  - このアクションに該当するリソースがありません。 – これは、ポリシーに含まれているアクションにサポートされたリソースがないことを意味しています。
  - このアクションには、該当するリソースと条件がありません。 – これは、ポリシーに含まれているアクションにサポートされたリソースおよび条件がないことを意味しています。この場合、このサービスのポリシーに条件は含まれていますが、このアクションに適用される条件はありません。
11. アクセス許可が指定されるアクションには、アクション概要へのリンクが含まれます。

## アクション概要 (リソースのリスト)

ポリシーは、ポリシー概要、[サービス概要](#)、[アクション概要](#)の3つのテーブルにまとめられています。アクション概要テーブルには、選択したアクションに対して適用されているリソースとその関連する条件のリストが含まれます。



アクセス許可が付与されるアクションごとにアクション概要を表示するには、サービス概要のリンクをクリックしてください。アクション概要テーブルには [Region (リージョン)] や [Account (アカウント)] など、リソースに関する詳細が表示されます。各リソースに適用される条件を表示することもできます。これにより、一部のリソースに適用されて他のリソースには適用されない条件が表示されます。

### アクション概要の表示

[ポリシー] ページでは、管理ポリシー、ユーザーにアタッチされているポリシー、およびロールにアタッチされているポリシーのアクション概要を表示できます。

管理ポリシーのアクション概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、表示するポリシーの名前を選択します。
4. ポリシーの [ポリシー詳細] ページで [アクセス許可] タブを表示して、ポリシー概要を確認します。



5. サービスのポリシー概要のリストで、表示するサービス名を選択します。
6. アクションのサービス概要リストで、表示するアクション名を選択します。

ユーザーにアタッチされたポリシーのアクション概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで [ユーザー] を選択します。
3. ユーザーのリストから、ポリシーを表示するユーザーを選択します。
4. ユーザーの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ユーザーに直接、またはグループのユーザーにアタッチされているポリシーのリストを表示します。
5. ユーザーのポリシーのテーブルで、表示するポリシーの名前を選択します。

[ユーザー] ページが表示された状態で、ユーザーにアタッチされているポリシーのサービス概要を選択すると、[ポリシー] ページにリダイレクトされます。サービスの概要は、[ポリシー] ページでのみ表示できます。

6. サービスのポリシー概要のリストで、表示するサービス名を選択します。

#### Note

選択したポリシーがユーザーに直接アタッチされているインラインポリシーの場合、サービス概要テーブルが表示されます。ポリシーがグループからアタッチされたインラインポリシーの場合、そのグループの JSON ポリシードキュメントに移動します。ポリシーが管理ポリシーの場合、そのポリシーのサービス概要が [ポリシー] ページに表示されます。

7. アクションのサービス概要リストで、表示するアクション名を選択します。

ロールにアタッチされたポリシーのアクション概要を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. ロールのリストから、表示するポリシーを含むロールの名前を選択します。



4. ロールの [Summary (概要)] ページの [Permissions (アクセス許可)] タブから、ロールにアタッチされているポリシーのリストを表示します。
5. ユーザーのポリシーのテーブルで、表示するポリシーの名前を選択します。

[ロール] ページが表示された状態で、ユーザーにアタッチされているポリシーのサービス概要を選択すると、[ポリシー] ページにリダイレクトされます。サービスの概要は、[ポリシー] ページでのみ表示できます。

6. サービスのポリシー概要のリストで、表示するサービス名を選択します。
7. アクションのサービス概要リストで、表示するアクション名を選択します。

## アクション概要の要素を理解する

以下の例は、Amazon S3 サービス概要 (「PutObject」を参照) からの [サービス概要 \(アクションのリスト\)](#) (Write) アクションの概要です。このアクションについて、ポリシーでは 1 つのリソースに対して複数の条件を定義しています。

**Permissions defined in this policy** [Info](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an [IAM](#) identity (user, user group, or role), attach a policy to it.

[Edit](#) [Summary](#) [JSON](#)

< [Actions](#) PutObject action in S3

Resource	Region	Account	Request condition
BucketName   string like   customer, ObjectPath   string like   All	All regions	All accounts	s3:x-amz-acl = public-read

アクション概要ページには、以下の情報が含まれます。

1. アクションに適用される複数の条件の表示など、ポリシーに関する追加の詳細を表示するには、[JSON] を選択します (ユーザーに直接アタッチされているインラインポリシーのアクション概要を表示している場合、手順は異なります。その場合、JSON ポリシードキュメントにアクセスするには、アクション概要ダイアログボックスを閉じて、ポリシー概要に戻る必要があります。)
2. 特定のリソースの概要を表示するには、キーワードを [検索] ボックスに入力して、使用可能なリソースのリストを減らします。

3. [アクション] リンクの横に、サービス名とアクションが `action name action in service` の形式で表示されます (この場合は `PutObject action in S3`)。このサービスのアクション概要には、ポリシーで定義されているリソースのリストが含まれます。
4. Resource (リソース) – この列には、選択したサービスに対してポリシーで定義したリソースが表示されます。この例では、すべてのオブジェクトパスで `PutObject` アクションが許可されていますが、Amazon S3 バケットリソース `developer_bucket` に対してのみ許可されます。サービスから `arn:aws:s3:::developer_bucket/*` に渡される情報に応じて、`BucketName = developer_bucket`, `ObjectPath = All` などの ARN が表示されるか、IAM などの定義されたリソースタイプが表示される場合があります。
5. リージョン – この列には、リソースが定義されているリージョンが表示されます。リソースは、すべてのリージョンまたは 1 つのリージョンに対して定義できます。特定の複数のリージョンに存在することはできません。
  - すべてのリージョン – リソースに関連付けられているアクションは、すべてのリージョンに適用されます。この例では、アクションはグローバルサービス Amazon S3 に属します。グローバルサービスに属するアクションは、すべてのリージョンに適用されます。
  - Region text – リソースに関連付けられているアクションは、1 つのリージョンに適用されます。たとえば、ポリシーではリソースに対して `us-east-2` リージョンを指定できます。
6. アカウント – この列には、リソースに関連付けられているサービスまたはアクションが特定のアカウントに適用されるかが示されます。リソースはすべてのアカウントか 1 つのアカウントに存在できます。特定の複数のアカウントに存在することはできません。
  - All accounts (すべてのアカウント) – リソースに関連付けられているアクションは、すべてのアカウントに適用されます。この例では、アクションはグローバルサービス Amazon S3 に属します。グローバルサービスに属するアクションは、すべてのアカウントに適用されます。
  - このアカウント – リソースに関連付けられているアクションは、現在のアカウントにのみ適用されます。
  - Account number – リソースに関連付けられているアクションは、1 つのアカウント (現在ログインしていないアカウント) に適用されます。たとえば、ポリシーでリソースに対して `123456789012` アカウントが指定されている場合、そのアカウント番号がポリシー概要に表示されます。
7. Request condition (リクエスト条件) – この列には、リソースに関連付けられているアクションが条件の対象かが示されます。この例には `s3:x-amz-acl = public-read` 条件が含まれています。これらの条件の詳細については、[JSON] を選択して JSON ポリシードキュメントを確認してください。

## ポリシー概要の例

以下の例には、JSON ポリシーと共に、関連する[ポリシー概要](#)、[サービス概要](#)、[アクション概要](#)が含まれており、ポリシーを通じて付与されるアクセス許可を理解するのに役立ちます。

### ポリシー 1: DenyCustomerBucket

このポリシーは、同じサービスに対する許可および拒否を示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccess",
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": ["*"]
    },
    {
      "Sid": "DenyCustomerBucket",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*" ]
    }
  ]
}
```

DenyCustomerBucket ポリシー概要:

**i** This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining**. [Learn more](#)

### Permissions defined in this policy [Info](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

[Edit](#)
[Summary](#)
[JSON](#)


#### Explicit deny (1 of 371 services)

Service	Access level	Resource	Request condition
S3	Limited: List, Permissions management, Read, Write, Tagging	Multiple	None

#### Allow (1 of 371 services)

 Show remaining 369 services

Service	Access level	Resource	Request condition
S3	Full access	All resources	None

DenyCustomerBucket S3 (Explicit deny) サービス概要:

< Services Actions in S3 (82 of 130) Show remaining 48 actions

**Read (35 of 53)**

Action	Resource	Request condition
<a href="#">GetAccelerateConfiguration</a>	BucketName  string like  customer	None
<a href="#">GetAnalyticsConfiguration</a>	BucketName  string like  customer	None
<a href="#">GetBucketAcl</a>	BucketName  string like  customer	None
<a href="#">GetBucketCORS</a>	BucketName  string like  customer	None
<a href="#">GetBucketLocation</a>	BucketName  string like  customer	None
<a href="#">GetBucketLogging</a>	BucketName  string like  customer	None
<a href="#">GetBucketNotification</a>	BucketName  string like  customer	None
<a href="#">GetBucketObjectLockConfiguration</a>	BucketName  string like  customer	None
<a href="#">GetBucketOwnershipControls</a>	BucketName  string like  customer	None
<a href="#">GetBucketPolicy</a>	BucketName  string like  customer	None
<a href="#">GetBucketPolicyStatus</a>	BucketName  string like  customer	None
<a href="#">GetBucketPublicAccessBlock</a>	BucketName  string like  customer	None
<a href="#">GetBucketRequestPayment</a>	BucketName  string like  customer	None
<a href="#">GetBucketTagging</a>	BucketName  string like  customer	None
<a href="#">GetBucketVersioning</a>	BucketName  string like  customer	None
<a href="#">GetBucketWebsite</a>	BucketName  string like  customer	None

### GetObject (Read) アクション概要:

< Actions GetObject action in S3

Resource	Region	Account	Request condition
BucketName  string like  customer, ObjectPath  string like  All	-	All accounts	None

### ポリシー 2: DynamoDbRowCognitoID

このポリシーでは、ユーザーの Amazon Cognito ID に基づいて、行レベルで Amazon DynamoDB にアクセスできます。

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DeleteItem",
      "dynamodb:GetItem",
      "dynamodb:PutItem",
      "dynamodb:UpdateItem"
    ],
    "Resource": [
      "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
    ],
    "Condition": {
      "ForAllValues:StringEquals": {
        "dynamodb:LeadingKeys": [
          "${cognito-identity.amazonaws.com:sub}"
        ]
      }
    }
  }
]
}

```

## DynamoDbRowCognitoID ポリシー概要:

Allow (1 of 370 services)		<input type="checkbox"/> Show remaining 369 services	
Service	Access level	Resource	Request condition
DynamoDB	Limited: Read, Write	region  string like  us-west-1, TableName  string like  myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

## DynamoDbRowCognitoID DynamoDB (Allow) サービス概要:

< Services Actions in DynamoDB (4 of 65)			Show remaining 61 actions
<b>Read (1 of 26)</b>			
Action	Resource	Request condition	
GetItem	region  string like [us-west-1, TableName] string like  myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}	
<b>Write (3 of 33)</b>			
Action	Resource	Request condition	
DeleteItem	region  string like [us-west-1, TableName] string like  myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}	
PutItem	region  string like [us-west-1, TableName] string like  myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}	
UpdateItem	region  string like [us-west-1, TableName] string like  myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}	

### GetItem (List) アクション概要:

< Actions GetItem action in DynamoDB			
Resource	Region	Account	Request condition
region  string like [us-west-1, TableName] string like  myDynamoTable	us-west-1	123456789012	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

## ポリシー 3: MultipleResourceCondition

このポリシーには、複数のリソースと条件が含まれます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": ["arn:aws:s3:::Apple_bucket/*"],
      "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
    },
    {
```

```

    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": ["arn:aws:s3:::Orange_bucket/*"],
    "Condition": {"StringEquals": {
      "s3:x-amz-acl": ["custom"],
      "s3:x-amz-grant-full-control": ["1234"]
    }}
  }
]
}

```

### MultipleResourceCondition ポリシー概要:

Allow (1 of 370 services) <span style="float: right;">Show remaining 369 services</span>			
Service	Access level	Resource	Request condition
S3	Limited: Permissions management, Write	Multiple	Multiple

### MultipleResourceCondition S3 (Allow) サービス概要:

< Services Actions in S3 (2 of 130) <span style="float: right;">Show remaining 128 actions</span>			
Write (1 of 47)			
Action	Resource	Request condition	
PutObject	Multiple	Multiple	
Permission Management (1 of 15)			
Action	Resource	Request condition	
PutObjectAcl	Multiple	Multiple	

### PutObject (Write) アクション概要:

< Actions PutObject action in S3			
Resource	Region	Account	Request condition
Multiple	-	All accounts	Multiple



## ポリシー 4: EC2\_Troubleshoot

以下のポリシーでは、ユーザーは実行中の Amazon EC2 インスタンスのスクリーンショットを取得できます。このスクリーンショットは EC2 のトラブルシューティングに役立つ場合があります。このポリシーでは、Amazon S3 開発者バケット内のアイテムに関する情報の表示も許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:GetConsoleScreenshot"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::developer"
      ]
    }
  ]
}
```

### EC2\_Troubleshoot ポリシー概要:

Allow (2 of 370 services) <span style="float: right;">Show remaining 368 services</span>			
Service ▲	Access level ▼	Resource	Request condition
EC2	Limited: Read	All resources	None
S3	Limited: List	BucketName  string like  developer	None

### EC2\_Troubleshoot S3 (Allow) サービス概要:

Action	Resource	Request condition
ListBucket	BucketName  string like  developer	None

### ListBucket (List) アクション概要:

Resource	Region	Account	Request condition
BucketName  string like  developer	-	All accounts	None

## ポリシー 5: CodeBuild\_CodeCommit\_CodeDeploy

このポリシーでは、特定の CodeBuild、CodeCommit、および CodeDeploy リソースへのアクセスを許可します。これらのリソースは各サービスに固有であるため、一致するサービスでのみ表示されます。いずれのサービスとも一致しないリソースを Action 要素の含めた場合、そのリソースはすべてのアクション概要に表示されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487980617000",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "codecommit:*",
        "codedeploy:*"
      ],
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project",
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
        "arn:aws:codedeploy:us-east-2:123456789012:application:WordPress_App",
        "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
      ]
    }
  ]
}
```

### CodeBuild\_CodeCommit\_CodeDeploy ポリシー概要:

Allow (3 of 370 services) <span style="float: right;">Show remaining 367 services</span>			
Service ▲	Access level ▼	Resource	Request condition
<a href="#">CodeBuild</a>	Full: Permissions management Limited: List, Read, Write	region  string like  us-east-2	None
<a href="#">CodeCommit</a>	Full: Tagging Limited: List, Read, Write	ResourceSpecifier  string like  MyDemoRepo, region  string like  us-east-2	None
<a href="#">CodeDeploy</a>	Full: Tagging Limited: List, Read, Write	Multiple	None

CodeBuild\_CodeCommit\_CodeDeploy CodeBuild (Allow) サービス概要:

< Services Actions in CodeBuild (24 of 53) Show remaining 29 actions

**Read (4 of 9)**

Action	Resource	Request condition
BatchGetBuildBatches	region  string like  us-east-2	None
BatchGetBuilds	region  string like  us-east-2	None
BatchGetProjects	region  string like  us-east-2	None
GetResourcePolicy	region  string like  us-east-2	None

**Write (16 of 28)**

Action	Resource	Request condition
BatchDeleteBuilds	region  string like  us-east-2	None
CreateProject	region  string like  us-east-2	None
CreateWebhook	region  string like  us-east-2	None
DeleteBuildBatch	region  string like  us-east-2	None
DeleteProject	region  string like  us-east-2	None
DeleteWebhook	region  string like  us-east-2	None
InvalidateProjectCache	region  string like  us-east-2	None
RetryBuild	region  string like  us-east-2	None
RetryBuildBatch	region  string like  us-east-2	None
StartBuild	region  string like  us-east-2	None
StartBuildBatch	region  string like  us-east-2	None
StopBuild	region  string like  us-east-2	None
StopBuildBatch	region  string like  us-east-2	None
UpdateProject	region  string like  us-east-2	None
UpdateProjectVisibility	region  string like  us-east-2	None
UpdateWebhook	region  string like  us-east-2	None

**List (2 of 14)**

## CodeBuild\_CodeCommit\_CodeDeploy StartBuild (Write) アクション概要:

< Actions StartBuild action in CodeBuild

Resource	Region	Account	Request condition
region  string like  us-east-2	us-east-2	123456789012	None

## 他の IAM リソースにアクセスするのに必要なアクセス許可

リソースはサービス内のオブジェクトです。IAM リソースには、グループ、ユーザー、ロール、およびポリシーが含まれます。AWS アカウントのルートユーザー 認証情報を使用してサインインしている場合は、IAM 認証情報または IAM リソースの管理に制限はありません。ただし、IAM ユーザーには、認証情報または IAM リソースを管理する権限が明示的に与えられている必要があります。これは、ユーザーに ID ベースのポリシーをアタッチすることで実行できます。

### Note

AWS ドキュメント全体で、特定のカテゴリに言及せずに IAM ポリシーを参照する場合は、それはアイデンティティベースのカスタマー管理ポリシーを意味します。ポリシーカテゴリの詳細については、「[the section called “ポリシーとアクセス許可”](#)」を参照してください。

## IAM ID を管理するためのアクセス許可

通常、IAM グループ、ユーザー、ロール、および認証情報を管理するために必要な権限は、タスクの API アクションに対応します。たとえば、IAM ユーザーを作成するには、対応する API コマンド: [iam:CreateUser](#) を持つ CreateUser アクセス許可を持っている必要があります。IAM ユーザーに他の IAM ユーザーを作成する権限を与えるには、そのユーザーに次のような IAM ポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

ポリシーの Resource 要素の値は、アクション、およびそのアクションの影響を受ける可能性のあるリソースによって決まります。前述の例のポリシーでは、ユーザーが任意のユーザーを作成することができます (\* はすべての文字列に一致するワイルドカードです)。対照的に、ユーザーに自分のアクセスキー (API アクション [CreateAccessKey](#) および [UpdateAccessKey](#)) のみの変更を許可するポリシーには、通常 Resource 要素が含まれます。この場合、ARN には次の例のように現在のユーザーの名前を解決する変数が含まれます (`${aws:username}`)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListUsersForConsole",
      "Effect": "Allow",
      "Action": "iam:ListUsers",
      "Resource": "arn:aws:iam::*:*"
    },
    {
      "Sid": "ViewAndUpdateAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:UpdateAccessKey",
        "iam:CreateAccessKey",
        "iam:ListAccessKeys"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

前の例では、`${aws:username}` は現在のユーザーのユーザー名に解決される変数です。ポリシー変数の詳細については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

多くの場合、アクション名にワイルドカード文字 (\*) を使用すると、特定のタスクに関連するすべてのアクションの権限を容易に付与できます。たとえば、ユーザーが任意の IAM アクションを実行することを許可するには、アクションに対して `iam:*` を使用できます。ユーザーがアクセスキーのみに関連する任意のアクションを実行することを許可するには、ポリシーステートメントの `iam:*AccessKey*` 要素で Action を使用できます。これにより、[CreateAccessKey](#)、[DeleteAccessKey](#)、[GetAccessKeyLastUsed](#)、[ListAccessKeys](#)、[UpdateAccessKey](#) の各アクションを実行するアクセス許可がユーザーに付与されます。(将来、名前に「AccessKey」を含むアクションが IAM に追加された場合、`iam:*AccessKey*` 要素の Action を使用していれば、その新しいアクションに対する権限もユーザーに与えられます)。以下の例では、自分のアクセスキーに関連するすべてのアクションの実行をユーザーに許可するポリシーを示しています (`account-id` は AWS アカウント ID に置き換えてください)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
```

```
"Action": "iam:*AccessKey*",
"Resource": "arn:aws:iam::account-id:user/${aws:username}"
}
}
```

一部のタスクには複数のアクションが必要です。たとえば、グループの削除では、まずグループからユーザーを削除し、グループのポリシーをデタッチまたは削除してから、グループを実際に削除する必要があります。ユーザーがグループを削除できるようにする場合は、関連するすべてのアクションの実行権限をそのユーザーに与える必要があります。

## AWS Management Consoleで作業するための許可

前の例では、[AWS CLI](#) または [AWS SDK](#) の使用によるアクションの実行をユーザーに許可するポリシーの例を示しています。

ユーザーがコンソールを操作すると、コンソールは、グループ、ユーザー、ロール、ポリシーのリストの取得、およびグループ、ユーザー、またはロールに関連付けられたポリシーの取得を行うためのリクエストを IAM に対して発行します。また、コンソールは AWS アカウント 情報とプリンシパルに関する情報を取得するリクエストを発行します。プリンシパルは、コンソールでリクエストを行うユーザーです。

一般に、アクションを実行するには、ポリシーに一致するアクションのみを含める必要があります。ユーザーを作成するには、CreateUser アクションを呼び出すアクセス許可が必要です。多くの場合、コンソールを使用してアクションを実行するときは、コンソール内のリソースを表示、一覧表示、取得、またはその他の方法で表示するアクセス許可が必要です。これは、指定されたアクションを実行するためにコンソールをナビゲートできるようにするために必要です。たとえば、ジョージというユーザーがコンソールを使用して自分のアクセスキーを変更する場合、ジョージはまず IAM コンソールに移動し、[Users] を選択します。このアクションにより、コンソールで [ListUsers](#) リクエストが発行されます。ジョージに iam:ListUsers アクションの権限がない場合は、コンソールがユーザーのリストの取得を試みたときに、コンソールはアクセスを拒否されます。結果として、Jorge は自分の名前やアクセスキーにアクセスできません。Jorge に [CreateAccessKey](#) アクションと [UpdateAccessKey](#) アクションに対するアクセス許可があっても関係ありません。

グループ、ユーザー、ロール、ポリシー、認証情報を管理するために AWS Management Console で作業する権限をユーザーに与える場合は、ユーザーがコンソールで実行するアクションに対する権限を含める必要があります。これらのアクセス権限をユーザーに付与するために使用できるポリシーの例については、「[IAM リソースの管理に関するポリシーの例](#)」を参照してください。

## AWS アカウント全体にわたるアクセス権限の付与

お客様は自らのアカウント内の IAM ユーザーに対し、お客様のリソースへのアクセス権限を直接付与できます。他のアカウントのユーザーがお客様のリソースへのアクセスを必要としている場合は、IAM ロールを作成します。このロールは、特定の権限を含むエンティティであり、特定のユーザーに関連付けられることはありません。これにより他のアカウントのユーザーはロールを使用して、ロールに割り当てられた権限に応じてリソースにアクセスできます。詳細については、「[所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供](#)」を参照してください。

### Note

一部のサービスは、[アイデンティティベースおよびリソースベースのポリシー](#) で説明されているリソースベースのポリシーをサポートしています (Amazon S3、Amazon SNS、Amazon SQS など)。これらのサービスでは、ロールを使用する代わりに、共有するリソース (バケット、トピック、またはキュー) にポリシーをアタッチします。リソースベースのポリシーでは、AWS アカウントで、リソースへのアクセス許可を持ちます。

## あるサービスから他のサービスへのアクセス権限

多くの AWS サービスは、他の AWS サービスにアクセスします。たとえば、Amazon EMR、Elastic Load Balancing、Amazon EC2 Auto Scaling などのいくつかの AWS サービスは、Amazon EC2 インスタンスを管理します。その他の AWS サービスは、Amazon S3 バケット、Amazon SNS トピック、Amazon SQS キューなどを利用します。

こうした場合におけるアクセス権限の管理方法は、サービスによって異なります。ここでは、異なるサービスでアクセス権限がどのように扱われるかについての例をいくつか紹介します。

- Amazon EC2 Auto Scaling では、ユーザーに Auto Scaling を使用する許可がある必要がありますが、Amazon EC2 インスタンスを管理する権限を明示的に付与されている必要はありません。
- AWS Data Pipeline では、IAM ロールによってパイプラインで何ができるかが決定され、ユーザーはロールを引き受けるためのアクセス許可が必要です (詳細については、[&guide-edp-dev;](#) の「AWS Data PipelineIAM を使用してパイプラインにアクセス権限を付与する」を参照してください)。

アクセス権限を適切に設定して AWS サービスが意図するタスクを実行できるようにする方法の詳細については、呼び出すサービスのドキュメントを参照してください。サービスのロールを作成する方法については、「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。



## ユーザーの代理操作を実行する IAM ロールでサービスを設定する

ユーザーの代理操作を行うように AWS サービスを設定する場合は、通常、サービスに許可する操作を定義する IAM ロールの ARN を指定します。AWS によって、サービスにロールを渡すアクセス許可があるかがチェックされます。詳細については、「[AWS のサービスにロールを渡すアクセス権限をユーザーに付与する](#)」を参照してください。

## 必須アクション

アクションは、リソースの表示、作成、編集、削除など、リソースに対して実行できる処理です。アクションは各 AWS サービスによって定義されます。

ユーザーがアクションを実行できるようにするには、呼び出し ID または影響を受けるリソースに適用されるポリシーに必要なアクションを含める必要があります。一般的に、アクションの実行に必要なアクセス許可を提供するには、そのアクションをポリシーに含める必要があります。たとえば、ユーザーを作成するには、CreateUser アクションをポリシーに追加する必要があります。

場合によっては、ポリシーに追加の関連アクションを含めなければならない場合があります。たとえば、AWS Directory Service にディレクトリを作成するアクセス許可をユーザーに付与するには、ds:CreateDirectory オペレーションを使用して、以下のアクションをポリシーに含める必要があります。

- ds:CreateDirectory
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- ec2:CreateSecurityGroup
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:AuthorizeSecurityGroupIngress
- ec2:AuthorizeSecurityGroupEgress

ビジュアルエディタを使用してポリシーを作成または編集すると、ポリシーに必要なすべてのアクションを選択するのに役立つ警告とプロンプトが表示されます。

AWS Directory Service にディレクトリを作成するために必要なアクセス許可の詳細については、「[例 2: ディレクトリを作成することをユーザーに許可する](#)」を参照してください。

## IAM リソースの管理に関するポリシーの例

以下に示すのは、IAM ユーザー、グループ、および認証情報の管理に関連するタスクをユーザーが実行可能にする IAM ポリシーの例です。これには、自分のパスワード、アクセスキー、多要素認証 (MFA) デバイスの管理をユーザーに許可するポリシーも含まれます。

Amazon S3、Amazon EC2、DynamoDB など、他の AWS サービスの使用によるタスクの実行をユーザーに許可するポリシーの例については、「[IAM アイデンティティベースのポリシーの例](#)」を参照してください。

### トピック

- [ユーザーがレポート作成の目的でアカウントのグループ、ユーザー、ポリシーなどを一覧表示することを許可する](#)
- [ユーザーがグループのメンバーシップを管理することを許可する](#)
- [ユーザーが IAM ユーザーを管理することを許可する](#)
- [ユーザーがアカウントパスワードポリシーを設定することを許可する](#)
- [ユーザーが IAM 認証情報レポートを生成、取得することを許可する](#)
- [すべての IAM アクション \(管理アクセス\) を許可する](#)

ユーザーがレポート作成の目的でアカウントのグループ、ユーザー、ポリシーなどを一覧表示することを許可する

次のポリシーでは、ユーザーは文字列 Get または List で始まる任意の IAM アクションを呼び出し、レポートを生成することができます。ポリシーの例を表示するには、「[IAM: IAM コンソールへの読み取り専用アクセスを許可する](#)」を参照してください。

ユーザーがグループのメンバーシップを管理することを許可する

以下のポリシーでは、ユーザーは MarketingGroup というグループのメンバーシップを更新することができます。ポリシーの例を表示するには、「[IAM:: グループのメンバーシップをプログラムによりコンソールで管理することを許可する](#)」を参照してください。

ユーザーが IAM ユーザーを管理することを許可する

次のポリシーでは、IAM ユーザーの管理に関連するすべてのタスクの実行がユーザーに許可されますが、グループやポリシーの作成など、他のエンティティに対するアクションの実行は許可されません。許可されるアクションは以下のとおりです。

- ユーザーの作成 ([CreateUser](#) アクション)。
- ユーザーの削除。このタスクでは、アクション [DeleteSigningCertificate](#)、[DeleteLoginProfile](#)、[RemoveUserFromGroup](#)、[DeleteUser](#) をすべて実行するアクセス許可が必要です。
- アカウントおよびグループのユーザーのリストの取得 ([GetUser](#)、[ListUsers](#)、[ListGroupsWithUser](#) アクション)。
- ユーザーのポリシーのリスト取得と削除 ([ListUserPolicies](#)、[ListAttachedUserPolicies](#)、[DetachUserPolicy](#)、[DeleteUserPolicy](#) アクション)
- ユーザーのパスの名前変更または変更 ([UpdateUser](#) アクション)。Resource 要素には、ソースパスとターゲットパスの両方に対応する ARN を含める必要があります。パスの詳細については、「[フレンドリ名とパス](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUsersToPerformUserActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListPolicies",
        "iam:GetPolicy",
        "iam:UpdateUser",
        "iam:AttachUserPolicy",
        "iam:ListEntitiesForPolicy",
        "iam:DeleteUserPolicy",
        "iam:DeleteUser",
        "iam:ListUserPolicies",
        "iam:CreateUser",
        "iam:RemoveUserFromGroup",
        "iam:AddUserToGroup",
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:PutUserPolicy",
        "iam:ListAttachedUserPolicies",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:DetachUserPolicy"
      ]
    }
  ],
}
```

```
    "Resource": "*"
  },
  {
    "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
    "Effect": "Allow",
    "Action": [
      "iam:GetAccount*",
      "iam:ListAccount*"
    ],
    "Resource": "*"
  }
]
```

上のポリシーに含まれているアクセス許可のうちいくつかは、AWS Management Consoleでタスクを実行することを許可します。[AWS CLI](#)、[AWS SDK](#)、または IAM HTTP クエリ API のみからユーザー関連のタスクを実行するユーザーには、一部のアクセス許可が不要になる可能性があります。たとえば、あるユーザーからデタッチするポリシーの ARN が既にわかっているユーザーに、iam:ListAttachedUserPolicies 権限は不要です。ユーザーが必要とする正確な権限リストは、ユーザーが他のユーザーを管理するときに実行する必要があるタスクによって決まります。

ポリシーの以下の権限では、ユーザーは AWS Management Console を介してタスクにアクセスすることができます。

- iam:GetAccount\*
- iam:ListAccount\*

## ユーザーがアカウントパスワードポリシーを設定することを許可する

AWS アカウントの[パスワードポリシー](#)を取得および更新するアクセス許可を一部のユーザーに付与することもできます。ポリシーの例を表示するには、「[IAM: アカウントのパスワード要件の設定をプログラムによりコンソールで許可する](#)」を参照してください。

## ユーザーが IAM 認証情報レポートを生成、取得することを許可する

AWS アカウントのすべてのユーザーを一覧表示するレポートを生成してダウンロードするアクセス許可をユーザーに付与できます。このレポートには、パスワード、アクセスキー、MFA デバイス、署名証明書など、さまざまなユーザー認証情報のステータスも一覧表示されます。認証情報レポートの詳細については、「[AWS アカウントの認証情報レポートの取得](#)」を参照してください。ポリシーの例を表示するには、「[IAM: 認証情報レポートを生成して取得する](#)」を参照してください。

## すべての IAM アクション (管理アクセス) を許可する

パスワード、アクセスキー、ユーザー証明書、MFA デバイス、ユーザー証明書の管理など、IAM 内のすべてのアクションを実行するための管理権限を一部のユーザーに与える場合があります。以下のポリシーの例は、次の権限を付与します。

### Warning

ユーザーに IAM に対するフルアクセスを許可する場合、ユーザーが自分または他者に付与できるアクセス許可に制限がなくなります。ユーザーは新しい IAM エンティティ (ユーザーまたはロール) を作成することや、それらのエンティティに AWS アカウントのすべてのリソースに対するフルアクセスを許可することができます。ユーザーに IAM に対するフルアクセスを許可した場合、事実上、AWS アカウントのすべてのリソースに対するフルアクセスを許可したことになります。これには、すべてのリソースを削除するためのアクセス権限も含まれます。これらのアクセス権限は信頼されている管理者にのみ付与してください。また、これらの管理者には多要素認証 (MFA) を適用してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*",
    "Resource": "*"
  }
}
```

# AWS SDK を使用した IAM のコード例

以下は、AWS Software Development Kit (SDK) で IAM を使用方法を説明するコード例です。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## コードの例

- [AWS SDK を使用した IAM のコード例](#)
  - [AWS SDK を使用した IAM 向けアクション](#)
    - [AWS SDK または CLI で AddClientIdToOpenIdConnectProvider を使用する](#)
    - [AWS SDK または CLI で AddRoleToInstanceProfile を使用する](#)
    - [AWS SDK または CLI で AddUserToGroup を使用する](#)
    - [AWS SDK または CLI で AttachGroupPolicy を使用する](#)
    - [AWS SDK または CLI で AttachRolePolicy を使用する](#)
    - [AWS SDK または CLI で AttachUserPolicy を使用する](#)
    - [AWS SDK または CLI で ChangePassword を使用する](#)
    - [AWS SDK または CLI で CreateAccessKey を使用する](#)
    - [AWS SDK または CLI で CreateAccountAlias を使用する](#)
    - [AWS SDK または CLI で CreateGroup を使用する](#)
    - [AWS SDK または CLI で CreateInstanceProfile を使用する](#)
    - [AWS SDK または CLI で CreateLoginProfile を使用する](#)
    - [AWS SDK または CLI で CreateOpenIdConnectProvider を使用する](#)
    - [AWS SDK または CLI で CreatePolicy を使用する](#)
    - [AWS SDK または CLI で CreatePolicyVersion を使用する](#)
    - [AWS SDK または CLI で CreateRole を使用する](#)
    - [AWS SDK または CLI で CreateSAMLProvider を使用する](#)
    - [AWS SDK または CLI で CreateServiceLinkedRole を使用する](#)
    - [AWS SDK または CLI で CreateUser を使用する](#)
    - [AWS SDK または CLI で CreateVirtualMfaDevice を使用する](#)
    - [AWS SDK または CLI で DeactivateMfaDevice を使用する](#)

- [AWS SDK または CLI で DeleteAccessKey を使用する](#)
- [AWS SDK または CLI で DeleteAccountAlias を使用する](#)
- [AWS SDK または CLI で DeleteAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で DeleteGroup を使用する](#)
- [AWS SDK または CLI で DeleteGroupPolicy を使用する](#)
- [AWS SDK または CLI で DeleteInstanceProfile を使用する](#)
- [AWS SDK または CLI で DeleteLoginProfile を使用する](#)
- [AWS SDK または CLI で DeleteOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で DeletePolicy を使用する](#)
- [AWS SDK または CLI で DeletePolicyVersion を使用する](#)
- [AWS SDK または CLI で DeleteRole を使用する](#)
- [AWS SDK または CLI で DeleteRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteRolePolicy を使用する](#)
- [AWS SDK または CLI で DeleteSAMLProvider を使用する](#)
- [AWS SDK または CLI で DeleteServerCertificate を使用する](#)
- [AWS SDK または CLI で DeleteServiceLinkedRole を使用する](#)
- [AWS SDK または CLI で DeleteSigningCertificate を使用する](#)
- [AWS SDK または CLI で DeleteUser を使用する](#)
- [AWS SDK または CLI で DeleteUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteUserPolicy を使用する](#)
- [AWS SDK または CLI で DeleteVirtualMfaDevice を使用する](#)
- [AWS SDK または CLI で DetachGroupPolicy を使用する](#)
- [AWS SDK または CLI で DetachRolePolicy を使用する](#)
- [AWS SDK または CLI で DetachUserPolicy を使用する](#)
- [AWS SDK または CLI で EnableMfaDevice を使用する](#)
- [AWS SDK または CLI で GenerateCredentialReport を使用する](#)
- [AWS SDK または CLI で GenerateServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetAccessKeyLastUsed を使用する](#)
- [AWS SDK または CLI で GetAccountAuthorizationDetails を使用する](#)
- [AWS SDK または CLI で GetAccountPasswordPolicy を使用する](#)



- [AWS SDK または CLI で GetAccountSummary を使用する](#)
- [AWS SDK または CLI で GetContextKeysForCustomPolicy を使用する](#)
- [AWS SDK または CLI で GetContextKeysForPrincipalPolicy を使用する](#)
- [AWS SDK または CLI で GetCredentialReport を使用する](#)
- [AWS SDK または CLI で GetGroup を使用する](#)
- [AWS SDK または CLI で GetGroupPolicy を使用する](#)
- [AWS SDK または CLI で GetInstanceProfile を使用する](#)
- [AWS SDK または CLI で GetLoginProfile を使用する](#)
- [AWS SDK または CLI で GetOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で GetPolicy を使用する](#)
- [AWS SDK または CLI で GetPolicyVersion を使用する](#)
- [AWS SDK または CLI で GetRole を使用する](#)
- [AWS SDK または CLI で GetRolePolicy を使用する](#)
- [AWS SDK または CLI で GetSamlProvider を使用する](#)
- [AWS SDK または CLI で GetServerCertificate を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetailsWithEntities を使用する](#)
- [AWS SDK または CLI で GetServiceLinkedRoleDeletionStatus を使用する](#)
- [AWS SDK または CLI で GetUser を使用する](#)
- [AWS SDK または CLI で GetUserPolicy を使用する](#)
- [AWS SDK または CLI で ListAccessKeys を使用する](#)
- [AWS SDK または CLI で ListAccountAliases を使用する](#)
- [AWS SDK または CLI で ListAttachedGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedRolePolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedUserPolicies を使用する](#)
- [AWS SDK または CLI で ListEntitiesForPolicy を使用する](#)
- [AWS SDK または CLI で ListGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListGroups を使用する](#)
- [AWS SDK または CLI で ListGroupsForUser を使用する](#)
- [AWS SDK または CLI で ListInstanceProfiles を使用する](#)



- [AWS SDK または CLI で ListInstanceProfilesForRole を使用する](#)
- [AWS SDK または CLI で ListMfaDevices を使用する](#)
- [AWS SDK または CLI で ListOpenIdConnectProviders を使用する](#)
- [AWS SDK または CLI で ListPolicies を使用する](#)
- [AWS SDK または CLI で ListPolicyVersions を使用する](#)
- [AWS SDK または CLI で ListRolePolicies を使用する](#)
- [AWS SDK または CLI で ListRoleTags を使用する](#)
- [AWS SDK または CLI で ListRoles を使用する](#)
- [AWS SDK または CLI で ListSAMLProviders を使用する](#)
- [AWS SDK または CLI で ListServerCertificates を使用する](#)
- [AWS SDK または CLI で ListSigningCertificates を使用する](#)
- [AWS SDK または CLI で ListUserPolicies を使用する](#)
- [AWS SDK または CLI で ListUserTags を使用する](#)
- [AWS SDK または CLI で ListUsers を使用する](#)
- [AWS SDK または CLI で ListVirtualMfaDevices を使用する](#)
- [AWS SDK または CLI で PutGroupPolicy を使用する](#)
- [AWS SDK または CLI で PutRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutRolePolicy を使用する](#)
- [AWS SDK または CLI で PutUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutUserPolicy を使用する](#)
- [AWS SDK または CLI で RemoveClientIdFromOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で RemoveRoleFromInstanceProfile を使用する](#)
- [AWS SDK または CLI で RemoveUserFromGroup を使用する](#)
- [AWS SDK または CLI で ResyncMfaDevice を使用する](#)
- [AWS SDK または CLI で SetDefaultPolicyVersion を使用する](#)
- [AWS SDK または CLI で TagRole を使用する](#)
- [AWS SDK または CLI で TagUser を使用する](#)
- [AWS SDK または CLI で UntagRole を使用する](#)
- [AWS SDK または CLI で UntagUser を使用する](#)
- [AWS SDK または CLI で UpdateAccessKey を使用する](#)

- [AWS SDK または CLI で UpdateAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で UpdateAssumeRolePolicy を使用する](#)
- [AWS SDK または CLI で UpdateGroup を使用する](#)
- [AWS SDK または CLI で UpdateLoginProfile を使用する](#)
- [AWS SDK または CLI で UpdateOpenIdConnectProviderThumbprint を使用する](#)
- [AWS SDK または CLI で UpdateRole を使用する](#)
- [AWS SDK または CLI で UpdateRoleDescription を使用する](#)
- [AWS SDK または CLI で UpdateSamlProvider を使用する](#)
- [AWS SDK または CLI で UpdateServerCertificate を使用する](#)
- [AWS SDK または CLI で UpdateSigningCertificate を使用する](#)
- [AWS SDK または CLI で UpdateUser を使用する](#)
- [AWS SDK または CLI で UploadServerCertificate を使用する](#)
- [AWS SDK または CLI で UploadSigningCertificate を使用する](#)
- [AWS SDK を使用する IAM のシナリオ](#)
  - [AWS SDK を使用してレジリエントなサービスを構築して管理](#)
  - [AWS SDK を使用して IAM グループを作成し、ユーザーをグループに追加します。](#)
  - [AWS SDK を使用して IAM ユーザーを作成し、AWS STS を持つロールを引き受ける](#)
  - [AWS SDK を使用して読み取り専用 IAM ユーザーおよび読み取り/書き込みできる IAM ユーザーを作成する](#)
  - [AWS SDK を使用して IAM アクセスキーを管理する](#)
  - [AWS SDK を使用して IAM ポリシーを管理する](#)
  - [AWS SDK を使用して IAM ロールを管理する](#)
  - [AWS SDK を使用して IAM アカウントを管理する](#)
  - [AWS SDK を使用して IAM ポリシーのバージョンをロールバックする](#)
  - [AWS SDK での IAM Policy Builder API の使用](#)
- [AWS SDK を使用した AWS STS のコード例](#)
  - [AWS SDK を使用した AWS STS 向けアクション](#)
    - [AWS SDK または CLI で AssumeRole を使用する](#)
    - [AWS SDK または CLI で AssumeRoleWithWebIdentity を使用する](#)
    - [AWS SDK または CLI で DecodeAuthorizationMessage を使用する](#)

- [AWS SDK または CLI で GetFederationToken を使用する](#)
- [AWS SDK または CLI で GetSessionToken を使用する](#)
- [AWS SDK を使用する AWS STS のシナリオ](#)
  - [AWS SDK を使用して AWS STS で MFA トークンを必要とする IAM ロールを割り当てる](#)
  - [AWS SDK を使用して、フェデレーションユーザー向けに AWS STS で URL を作成する](#)
  - [AWS SDK を使用して、AWS STS で MFA トークンを必要とするセッショントークンを取得する](#)

## AWS SDK を使用した IAM のコード例

以下は、AWS Software Development Kit (SDK) で IAM を使用方法を説明するコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

IAM へようこそ

次のコード例は、IAM の使用を開始する方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
{policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[ListPolicies](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

## CMakeLists.txt CMake ファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
  string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
    "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
  list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.

  # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
  may need to uncomment this
  # and set the proper subdirectory to the executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
#{AWSSDK_LINK_LIBRARIES})
```

iam.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 * A "Hello IAM" starter application which initializes an AWS Identity and
 * Access Management (IAM) client
 * and lists the IAM policies.
 *
 * main function
 *
 * Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
```

```
        outcome.GetError().GetMessage() << std::endl;
        result = 1;
        break;
    }

    if (!header) {
        std::cout << std::left << std::setw(55) << "Name" <<
            std::setw(30) << "ID" << std::setw(80) << "Arn" <<
            std::setw(64) << "Description" << std::setw(12) <<
            "CreateDate" << std::endl;
        header = true;
    }

    const auto &policies = outcome.GetResult().GetPolicies();
    for (const auto &policy: policies) {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) <<
policy.GetArn() <<
            std::setw(64) << policy.GetDescription() <<
std::setw(12) <<
            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
<<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    } else {
        done = true;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[ListPolicies](#)」を参照してください。

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
// Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    sdkConfig, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    iamClient := iam.NewFromConfig(sdkConfig)
    const maxPols = 10
    fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
    result, err := iamClient.ListPolicies(context.TODO(), &iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPols),
    })
    if err != nil {
```



```
    fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
    return
}
if len(result.Policies) == 0 {
    fmt.Println("You don't have any policies!")
} else {
    for _, policy := range result.Policies {
        fmt.Printf("\t\t%v\n", *policy.PolicyName)
    }
}
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListPolicies](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListPolicies](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
    /**
```

```
* In v3, the clients expose paginateOperationName APIs that are written using
async generators so that you can use async iterators in a for await..of loop.
* https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
*/
const paginator = paginateListPolicies(
  { client, pageSize: 10 },
  // List only customer managed policies.
  { Scope: "Local" },
);

console.log("IAM policies defined in your account:");
let policyCount = 0;
for await (const page of paginator) {
  if (page.Policies) {
    page.Policies.forEach((p) => {
      console.log(`${p.PolicyName}`);
      policyCount++;
    });
  }
}
console.log(`Found ${policyCount} policies.`);
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListPolicies](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

From src/bin/hello.rs.

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
```

```

use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}

```

From src/iam-service-lib.rs.

```

pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {

```

```
        let name = p
            .policy_name
            .unwrap_or_else(|| "Missing Policy Name".to_string());
        println!("{}", name);
        name
    })
    .collect();

Ok(policy_names)
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[ListPolicies](#)」を参照してください。

## コードの例

- [AWS SDK を使用した IAM 向けアクション](#)
  - [AWS SDK または CLI で AddClientIdToOpenIdConnectProvider を使用する](#)
  - [AWS SDK または CLI で AddRoleToInstanceProfile を使用する](#)
  - [AWS SDK または CLI で AddUserToGroup を使用する](#)
  - [AWS SDK または CLI で AttachGroupPolicy を使用する](#)
  - [AWS SDK または CLI で AttachRolePolicy を使用する](#)
  - [AWS SDK または CLI で AttachUserPolicy を使用する](#)
  - [AWS SDK または CLI で ChangePassword を使用する](#)
  - [AWS SDK または CLI で CreateAccessKey を使用する](#)
  - [AWS SDK または CLI で CreateAccountAlias を使用する](#)
  - [AWS SDK または CLI で CreateGroup を使用する](#)
  - [AWS SDK または CLI で CreateInstanceProfile を使用する](#)
  - [AWS SDK または CLI で CreateLoginProfile を使用する](#)
  - [AWS SDK または CLI で CreateOpenIdConnectProvider を使用する](#)
  - [AWS SDK または CLI で CreatePolicy を使用する](#)
  - [AWS SDK または CLI で CreatePolicyVersion を使用する](#)
  - [AWS SDK または CLI で CreateRole を使用する](#)
  - [AWS SDK または CLI で CreateSAMLProvider を使用する](#)

- [AWS SDK または CLI で CreateServiceLinkedRole を使用する](#)
- [AWS SDK または CLI で CreateUser を使用する](#)
- [AWS SDK または CLI で CreateVirtualMfaDevice を使用する](#)
- [AWS SDK または CLI で DeactivateMfaDevice を使用する](#)
- [AWS SDK または CLI で DeleteAccessKey を使用する](#)
- [AWS SDK または CLI で DeleteAccountAlias を使用する](#)
- [AWS SDK または CLI で DeleteAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で DeleteGroup を使用する](#)
- [AWS SDK または CLI で DeleteGroupPolicy を使用する](#)
- [AWS SDK または CLI で DeleteInstanceProfile を使用する](#)
- [AWS SDK または CLI で DeleteLoginProfile を使用する](#)
- [AWS SDK または CLI で DeleteOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で DeletePolicy を使用する](#)
- [AWS SDK または CLI で DeletePolicyVersion を使用する](#)
- [AWS SDK または CLI で DeleteRole を使用する](#)
- [AWS SDK または CLI で DeleteRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteRolePolicy を使用する](#)
- [AWS SDK または CLI で DeleteSAMLProvider を使用する](#)
- [AWS SDK または CLI で DeleteServerCertificate を使用する](#)
- [AWS SDK または CLI で DeleteServiceLinkedRole を使用する](#)
- [AWS SDK または CLI で DeleteSigningCertificate を使用する](#)
- [AWS SDK または CLI で DeleteUser を使用する](#)
- [AWS SDK または CLI で DeleteUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteUserPolicy を使用する](#)
- [AWS SDK または CLI で DeleteVirtualMfaDevice を使用する](#)
- [AWS SDK または CLI で DetachGroupPolicy を使用する](#)
- [AWS SDK または CLI で DetachRolePolicy を使用する](#)
- [AWS SDK または CLI で DetachUserPolicy を使用する](#)
- [AWS SDK または CLI で EnableMfaDevice を使用する](#)
- [AWS SDK または CLI で GenerateCredentialReport を使用する](#)

- [AWS SDK または CLI で GenerateServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetAccessKeyLastUsed を使用する](#)
- [AWS SDK または CLI で GetAccountAuthorizationDetails を使用する](#)
- [AWS SDK または CLI で GetAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で GetAccountSummary を使用する](#)
- [AWS SDK または CLI で GetContextKeysForCustomPolicy を使用する](#)
- [AWS SDK または CLI で GetContextKeysForPrincipalPolicy を使用する](#)
- [AWS SDK または CLI で GetCredentialReport を使用する](#)
- [AWS SDK または CLI で GetGroup を使用する](#)
- [AWS SDK または CLI で GetGroupPolicy を使用する](#)
- [AWS SDK または CLI で GetInstanceProfile を使用する](#)
- [AWS SDK または CLI で GetLoginProfile を使用する](#)
- [AWS SDK または CLI で GetOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で GetPolicy を使用する](#)
- [AWS SDK または CLI で GetPolicyVersion を使用する](#)
- [AWS SDK または CLI で GetRole を使用する](#)
- [AWS SDK または CLI で GetRolePolicy を使用する](#)
- [AWS SDK または CLI で GetSamlProvider を使用する](#)
- [AWS SDK または CLI で GetServerCertificate を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetailsWithEntities を使用する](#)
- [AWS SDK または CLI で GetServiceLinkedRoleDeletionStatus を使用する](#)
- [AWS SDK または CLI で GetUser を使用する](#)
- [AWS SDK または CLI で GetUserPolicy を使用する](#)
- [AWS SDK または CLI で ListAccessKeys を使用する](#)
- [AWS SDK または CLI で ListAccountAliases を使用する](#)
- [AWS SDK または CLI で ListAttachedGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedRolePolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedUserPolicies を使用する](#)
- [AWS SDK または CLI で ListEntitiesForPolicy を使用する](#)

- [AWS SDK または CLI で ListGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListGroups を使用する](#)
- [AWS SDK または CLI で ListGroupsForUser を使用する](#)
- [AWS SDK または CLI で ListInstanceProfiles を使用する](#)
- [AWS SDK または CLI で ListInstanceProfilesForRole を使用する](#)
- [AWS SDK または CLI で ListMfaDevices を使用する](#)
- [AWS SDK または CLI で ListOpenIdConnectProviders を使用する](#)
- [AWS SDK または CLI で ListPolicies を使用する](#)
- [AWS SDK または CLI で ListPolicyVersions を使用する](#)
- [AWS SDK または CLI で ListRolePolicies を使用する](#)
- [AWS SDK または CLI で ListRoleTags を使用する](#)
- [AWS SDK または CLI で ListRoles を使用する](#)
- [AWS SDK または CLI で ListSAMLProviders を使用する](#)
- [AWS SDK または CLI で ListServerCertificates を使用する](#)
- [AWS SDK または CLI で ListSigningCertificates を使用する](#)
- [AWS SDK または CLI で ListUserPolicies を使用する](#)
- [AWS SDK または CLI で ListUserTags を使用する](#)
- [AWS SDK または CLI で ListUsers を使用する](#)
- [AWS SDK または CLI で ListVirtualMfaDevices を使用する](#)
- [AWS SDK または CLI で PutGroupPolicy を使用する](#)
- [AWS SDK または CLI で PutRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutRolePolicy を使用する](#)
- [AWS SDK または CLI で PutUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutUserPolicy を使用する](#)
- [AWS SDK または CLI で RemoveClientIdFromOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で RemoveRoleFromInstanceProfile を使用する](#)
- [AWS SDK または CLI で RemoveUserFromGroup を使用する](#)
- [AWS SDK または CLI で ResyncMfaDevice を使用する](#)
- [AWS SDK または CLI で SetDefaultPolicyVersion を使用する](#)
- [AWS SDK または CLI で TagRole を使用する](#)



- [AWS SDK または CLI で TagUser を使用する](#)
- [AWS SDK または CLI で UntagRole を使用する](#)
- [AWS SDK または CLI で UntagUser を使用する](#)
- [AWS SDK または CLI で UpdateAccessKey を使用する](#)
- [AWS SDK または CLI で UpdateAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で UpdateAssumeRolePolicy を使用する](#)
- [AWS SDK または CLI で UpdateGroup を使用する](#)
- [AWS SDK または CLI で UpdateLoginProfile を使用する](#)
- [AWS SDK または CLI で UpdateOpenIdConnectProviderThumbprint を使用する](#)
- [AWS SDK または CLI で UpdateRole を使用する](#)
- [AWS SDK または CLI で UpdateRoleDescription を使用する](#)
- [AWS SDK または CLI で UpdateSamlProvider を使用する](#)
- [AWS SDK または CLI で UpdateServerCertificate を使用する](#)
- [AWS SDK または CLI で UpdateSigningCertificate を使用する](#)
- [AWS SDK または CLI で UpdateUser を使用する](#)
- [AWS SDK または CLI で UploadServerCertificate を使用する](#)
- [AWS SDK または CLI で UploadSigningCertificate を使用する](#)
- [AWS SDK を使用する IAM のシナリオ](#)
  - [AWS SDK を使用してレジリエントなサービスを構築して管理](#)
  - [AWS SDK を使用して IAM グループを作成し、ユーザーをグループに追加します。](#)
  - [AWS SDK を使用して IAM ユーザーを作成し、AWS STS を持つロールを引き受ける](#)
  - [AWS SDK を使用して読み取り専用 IAM ユーザーおよび読み取り/書き込みできる IAM ユーザーを作成する](#)
  - [AWS SDK を使用して IAM アクセスキーを管理する](#)
  - [AWS SDK を使用して IAM ポリシーを管理する](#)
  - [AWS SDK を使用して IAM ロールを管理する](#)
  - [AWS SDK を使用して IAM アカウントを管理する](#)
  - [AWS SDK を使用して IAM ポリシーのバージョンをロールバックする](#)
  - [AWS SDK での IAM Policy Builder API の使用](#)

## AWS SDK を使用した IAM 向けアクション

以下は、AWS SDK を使用して個々の IAM アクションを実行する方法を説明するコード例です。これらは IAM API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「[AWS Identity and Access Management \(IAM\) API リファレンス](#)」を参照してください。

### 例

- [AWS SDK または CLI で AddClientIdToOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で AddRoleToInstanceProfile を使用する](#)
- [AWS SDK または CLI で AddUserToGroup を使用する](#)
- [AWS SDK または CLI で AttachGroupPolicy を使用する](#)
- [AWS SDK または CLI で AttachRolePolicy を使用する](#)
- [AWS SDK または CLI で AttachUserPolicy を使用する](#)
- [AWS SDK または CLI で ChangePassword を使用する](#)
- [AWS SDK または CLI で CreateAccessKey を使用する](#)
- [AWS SDK または CLI で CreateAccountAlias を使用する](#)
- [AWS SDK または CLI で CreateGroup を使用する](#)
- [AWS SDK または CLI で CreateInstanceProfile を使用する](#)
- [AWS SDK または CLI で CreateLoginProfile を使用する](#)
- [AWS SDK または CLI で CreateOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で CreatePolicy を使用する](#)
- [AWS SDK または CLI で CreatePolicyVersion を使用する](#)
- [AWS SDK または CLI で CreateRole を使用する](#)
- [AWS SDK または CLI で CreateSAMLProvider を使用する](#)
- [AWS SDK または CLI で CreateServiceLinkedRole を使用する](#)
- [AWS SDK または CLI で CreateUser を使用する](#)
- [AWS SDK または CLI で CreateVirtualMfaDevice を使用する](#)
- [AWS SDK または CLI で DeactivateMfaDevice を使用する](#)
- [AWS SDK または CLI で DeleteAccessKey を使用する](#)

- [AWS SDK または CLI で DeleteAccountAlias を使用する](#)
- [AWS SDK または CLI で DeleteAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で DeleteGroup を使用する](#)
- [AWS SDK または CLI で DeleteGroupPolicy を使用する](#)
- [AWS SDK または CLI で DeleteInstanceProfile を使用する](#)
- [AWS SDK または CLI で DeleteLoginProfile を使用する](#)
- [AWS SDK または CLI で DeleteOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で DeletePolicy を使用する](#)
- [AWS SDK または CLI で DeletePolicyVersion を使用する](#)
- [AWS SDK または CLI で DeleteRole を使用する](#)
- [AWS SDK または CLI で DeleteRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteRolePolicy を使用する](#)
- [AWS SDK または CLI で DeleteSAMLProvider を使用する](#)
- [AWS SDK または CLI で DeleteServerCertificate を使用する](#)
- [AWS SDK または CLI で DeleteServiceLinkedRole を使用する](#)
- [AWS SDK または CLI で DeleteSigningCertificate を使用する](#)
- [AWS SDK または CLI で DeleteUser を使用する](#)
- [AWS SDK または CLI で DeleteUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で DeleteUserPolicy を使用する](#)
- [AWS SDK または CLI で DeleteVirtualMfaDevice を使用する](#)
- [AWS SDK または CLI で DetachGroupPolicy を使用する](#)
- [AWS SDK または CLI で DetachRolePolicy を使用する](#)
- [AWS SDK または CLI で DetachUserPolicy を使用する](#)
- [AWS SDK または CLI で EnableMfaDevice を使用する](#)
- [AWS SDK または CLI で GenerateCredentialReport を使用する](#)
- [AWS SDK または CLI で GenerateServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetAccessKeyLastUsed を使用する](#)
- [AWS SDK または CLI で GetAccountAuthorizationDetails を使用する](#)
- [AWS SDK または CLI で GetAccountPasswordPolicy を使用する](#)
- [AWS SDK または CLI で GetAccountSummary を使用する](#)

- [AWS SDK または CLI で GetContextKeysForCustomPolicy を使用する](#)
- [AWS SDK または CLI で GetContextKeysForPrincipalPolicy を使用する](#)
- [AWS SDK または CLI で GetCredentialReport を使用する](#)
- [AWS SDK または CLI で GetGroup を使用する](#)
- [AWS SDK または CLI で GetGroupPolicy を使用する](#)
- [AWS SDK または CLI で GetInstanceProfile を使用する](#)
- [AWS SDK または CLI で GetLoginProfile を使用する](#)
- [AWS SDK または CLI で GetOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で GetPolicy を使用する](#)
- [AWS SDK または CLI で GetPolicyVersion を使用する](#)
- [AWS SDK または CLI で GetRole を使用する](#)
- [AWS SDK または CLI で GetRolePolicy を使用する](#)
- [AWS SDK または CLI で GetSamlProvider を使用する](#)
- [AWS SDK または CLI で GetServerCertificate を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetails を使用する](#)
- [AWS SDK または CLI で GetServiceLastAccessedDetailsWithEntities を使用する](#)
- [AWS SDK または CLI で GetServiceLinkedRoleDeletionStatus を使用する](#)
- [AWS SDK または CLI で GetUser を使用する](#)
- [AWS SDK または CLI で GetUserPolicy を使用する](#)
- [AWS SDK または CLI で ListAccessKeys を使用する](#)
- [AWS SDK または CLI で ListAccountAliases を使用する](#)
- [AWS SDK または CLI で ListAttachedGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedRolePolicies を使用する](#)
- [AWS SDK または CLI で ListAttachedUserPolicies を使用する](#)
- [AWS SDK または CLI で ListEntitiesForPolicy を使用する](#)
- [AWS SDK または CLI で ListGroupPolicies を使用する](#)
- [AWS SDK または CLI で ListGroups を使用する](#)
- [AWS SDK または CLI で ListGroupsForUser を使用する](#)
- [AWS SDK または CLI で ListInstanceProfiles を使用する](#)
- [AWS SDK または CLI で ListInstanceProfilesForRole を使用する](#)

- [AWS SDK または CLI で ListMfaDevices を使用する](#)
- [AWS SDK または CLI で ListOpenIdConnectProviders を使用する](#)
- [AWS SDK または CLI で ListPolicies を使用する](#)
- [AWS SDK または CLI で ListPolicyVersions を使用する](#)
- [AWS SDK または CLI で ListRolePolicies を使用する](#)
- [AWS SDK または CLI で ListRoleTags を使用する](#)
- [AWS SDK または CLI で ListRoles を使用する](#)
- [AWS SDK または CLI で ListSAMLProviders を使用する](#)
- [AWS SDK または CLI で ListServerCertificates を使用する](#)
- [AWS SDK または CLI で ListSigningCertificates を使用する](#)
- [AWS SDK または CLI で ListUserPolicies を使用する](#)
- [AWS SDK または CLI で ListUserTags を使用する](#)
- [AWS SDK または CLI で ListUsers を使用する](#)
- [AWS SDK または CLI で ListVirtualMfaDevices を使用する](#)
- [AWS SDK または CLI で PutGroupPolicy を使用する](#)
- [AWS SDK または CLI で PutRolePermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutRolePolicy を使用する](#)
- [AWS SDK または CLI で PutUserPermissionsBoundary を使用する](#)
- [AWS SDK または CLI で PutUserPolicy を使用する](#)
- [AWS SDK または CLI で RemoveClientIdFromOpenIdConnectProvider を使用する](#)
- [AWS SDK または CLI で RemoveRoleFromInstanceProfile を使用する](#)
- [AWS SDK または CLI で RemoveUserFromGroup を使用する](#)
- [AWS SDK または CLI で ResyncMfaDevice を使用する](#)
- [AWS SDK または CLI で SetDefaultPolicyVersion を使用する](#)
- [AWS SDK または CLI で TagRole を使用する](#)
- [AWS SDK または CLI で TagUser を使用する](#)
- [AWS SDK または CLI で UntagRole を使用する](#)
- [AWS SDK または CLI で UntagUser を使用する](#)
- [AWS SDK または CLI で UpdateAccessKey を使用する](#)
- [AWS SDK または CLI で UpdateAccountPasswordPolicy を使用する](#)

- [AWS SDK または CLI で UpdateAssumeRolePolicy を使用する](#)
- [AWS SDK または CLI で UpdateGroup を使用する](#)
- [AWS SDK または CLI で UpdateLoginProfile を使用する](#)
- [AWS SDK または CLI で UpdateOpenIdConnectProviderThumbprint を使用する](#)
- [AWS SDK または CLI で UpdateRole を使用する](#)
- [AWS SDK または CLI で UpdateRoleDescription を使用する](#)
- [AWS SDK または CLI で UpdateSamlProvider を使用する](#)
- [AWS SDK または CLI で UpdateServerCertificate を使用する](#)
- [AWS SDK または CLI で UpdateSigningCertificate を使用する](#)
- [AWS SDK または CLI で UpdateUser を使用する](#)
- [AWS SDK または CLI で UploadServerCertificate を使用する](#)
- [AWS SDK または CLI で UploadSigningCertificate を使用する](#)

## AWS SDK または CLI で **AddClientIdToOpenIdConnectProvider** を使用する

以下のコード例は、AddClientIdToOpenIdConnectProvider の使用方法を示しています。

### CLI

#### AWS CLI

Open-ID Connect (OIDC) プロバイダーにクライアント ID (対象者) を追加するには

次の add-client-id-to-open-id-connect-provider コマンドは、クライアント ID my-application-ID を server.example.com という名前の OIDC プロバイダーに追加します。

```
aws iam add-client-id-to-open-id-connect-provider \  
  --client-id my-application-ID \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
server.example.com
```

このコマンドでは何も出力されません。

OIDC プロバイダーを作成するには、create-open-id-connect-provider コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AddClientIdToOpenIdConnectProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、クライアント ID (または対象者) **my-application-ID** を **server.example.com** という名前の既存の OIDC プロバイダーに追加します。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
-OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AddClientIdToOpenIdConnectProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **AddRoleToInstanceProfile** を使用する

以下のコード例は、AddRoleToInstanceProfile の使用方法を示しています。

### CLI

#### AWS CLI

ロールをインスタンスプロファイルに追加するには

次の add-role-to-instance-profile コマンドは、S3Access という名前のロールを Webserver という名前のインスタンスプロファイルに追加します。

```
aws iam add-role-to-instance-profile \
  --role-name S3Access \
  --instance-profile-name Webserver
```



このコマンドでは何も出力されません。

インスタンスプロファイルを作成するには、`create-instance-profile` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AddRoleToInstanceProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、**S3Access** という名前のロールを **webserver** という名前の既存のインスタンスプロファイルに追加します。インスタンスプロファイルを作成するには、**New-IAMInstanceProfile** コマンドを使用します。このコマンドを使用してインスタンスプロファイルを作成し、ロールに関連付けると、EC2 インスタンスにアタッチできます。そのためには、**New-EC2Instance** コマンドレットを **InstanceProfile\_Arn** または **InstanceProfile-Name** パラメータと共に使用して、新しいインスタンスを起動します。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName  
"webserver"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AddRoleToInstanceProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **AddUserToGroup** を使用する

以下のコード例は、`AddUserToGroup` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)



## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Add an existing IAM user to an existing IAM group.
/// </summary>
/// <param name="userName">The username of the user to add.</param>
/// <param name="groupName">The name of the group to add the user to.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
{
    var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
    {
        GroupName = groupName,
        UserName = userName,
    });

    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[AddUserToGroup](#)」を参照してください。

## CLI

### AWS CLI

IAM グループにユーザーを追加するには

次の `add-user-to-group` コマンドは、Bob という名前の IAM ユーザーを Admins という名前の IAM グループに追加します。

```
aws iam add-user-to-group \  
  --user-name Bob \  
  --group-name Admins
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループへのユーザーの追加と削除](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AddUserToGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、**Bob** という名前のユーザーを **Admins** という名前のグループに追加します。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AddUserToGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **AttachGroupPolicy** を使用する

以下のコード例は、`AttachGroupPolicy` の使用方法を示しています。

### CLI

#### AWS CLI

管理ポリシーを IAM グループにアタッチするには

次の `attach-group-policy` コマンドは、`ReadOnlyAccess` という名前の AWS 管理ポリシーを `Finance` という名前の IAM グループにアタッチします。

```
aws iam attach-group-policy \  
  --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \  
  --group-name Finance
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[管理ポリシーとインラインポリシー](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachGroupPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**TesterPolicy** という名前のカスタマー管理ポリシーを IAM グループ **Testers** にアタッチします。そのグループのユーザーは、そのポリシーのデフォルトバージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterPolicy
```

例 2: この例では、**AdministratorAccess** という名前の AWS 管理ポリシーを IAM グループ **Admins** にアタッチします。そのグループのユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/  
AdministratorAccess
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AttachGroupPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **AttachRolePolicy** を使用する

以下のコード例は、AttachRolePolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [ロールの管理](#)

### .NET

#### AWS SDK for .NET

##### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Attach an IAM policy to a role.
/// </summary>
/// <param name="policyArn">The policy to attach.</param>
/// <param name="roleName">The role that the policy will be attached to.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AttachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_attach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo " -n role_name    The name of the IAM role."
    echo " -p policy_arn -- The IAM policy document ARN."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) role_name="${OPTARG}" ;;
        p) policy_arn="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")
```

```
local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[AttachRolePolicy](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
```

```
        roleName << ": " << list_outcome.GetError().GetMessage() <<
        std::endl;
    return false;
}

const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
if (std::any_of(policies.cbegin(), policies.cend(),
    [=](const Aws::IAM::Model::AttachedPolicy &policy) {
        return policy.GetPolicyArn() == policyArn;
    })) {
    std::cout << "Policy " << policyArn <<
    " is already attached to role " << roleName << std::endl;
    return true;
}

done = !list_outcome.GetResult().GetIsTruncated();
list_request.SetMarker(list_outcome.GetResult().GetMarker());
}

Aws::IAM::Model::AttachRolePolicyRequest request;
request.SetRoleName(roleName);
request.SetPolicyArn(policyArn);

Aws::IAM::Model::AttachRolePolicyOutcome outcome =
iam.AttachRolePolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to attach policy " << policyArn << " to role " <<
    roleName << ": " << outcome.GetError().GetMessage() <<
std::endl;
}
else {
    std::cout << "Successfully attached policy " << policyArn << " to role "
<<
    roleName << std::endl;
}

return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[AttachRolePolicy](#)」を参照してください。



## CLI

### AWS CLI

管理ポリシーを IAM ロールにアタッチするには

次の `attach-role-policy` コマンドは、`ReadOnlyAccess` と呼ばれる AWS 管理ポリシー IAM ロールを `ReadOnlyRole` と呼ばれる IAM ロールにアタッチします。

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \  
  --role-name ReadOnlyRole
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[管理ポリシーとインラインポリシー](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Go

### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {  
  iamClient *iam.Client  
}  
  
// AttachRolePolicy attaches a policy to a role.
```

```
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
error {
    _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
&iam.AttachRolePolicyInput{
    PolicyArn: aws.String(policyArn),
    RoleName:  aws.String(roleName),
})
    if err != nil {
        log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
roleName, err)
    }
    return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import
    software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
```

```
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーをアタッチします。

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
  }
  var params = {
    PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
    RoleName: process.argv[2],
  };
  iam.attachRolePolicy(params, function (err, data) {
    if (err) {
      console.log("Unable to attach policy to role", err);
    } else {
      console.log("Role attached successfully");
    }
  });
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
    }
}
```

```
        println("Successfully attached policy $policyArnVal to role
        $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
```



```
        \"Principal\": {\"AWS\": \"${user['Arn']}\"},
        \"Action\": \"sts:AssumeRole\"
    }]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_${uuid}",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3:*:*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_${uuid}",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

public function attachRolePolicy($roleName, $policyArn)
{
    return $this->customWaiter(function () use ($roleName, $policyArn) {
        $this->iamClient->attachRolePolicy([
            'PolicyArn' => $policyArn,
            'RoleName' => $roleName,
        ]);
    });
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**SecurityAudit** という名前の AWS 管理ポリシーを IAM ロール **CoSecurityAuditors** にアタッチします。そのロールを引き受けるユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
arn:aws:iam::aws:policy/SecurityAudit
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

Boto3 Policy オブジェクトを使用して、ロールにポリシーをアタッチします。

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
            role_name)
        raise
```

Boto3 Role オブジェクトを使用して、ロールにポリシーをアタッチします。

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. Note this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
            role_name)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
    #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
    exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
    #{e.message}")
  end
end
```

```
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
  end
end
```

```
    false
  end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func attachRolePolicy(role: String, policyArn: String) async throws {
    let input = AttachRolePolicyInput(
        policyArn: policyArn,
        roleName: role
    )
    do {
        _ = try await client.attachRolePolicy(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **AttachUserPolicy** を使用する

以下のコード例は、AttachUserPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

## CLI

### AWS CLI

管理ポリシーを IAM ユーザーにアタッチするには

次の `attach-user-policy` コマンドは、AdministratorAccess という名前の AWS 管理ポリシーを Alice という名前の IAM ユーザーにアタッチします。

```
aws iam attach-user-policy \  
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
  --user-name Alice
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[管理ポリシーとインラインポリシー](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachUserPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**AmazonCognitoPowerUser** という名前の AWS 管理ポリシーを IAM ユーザー **Bob** にアタッチします。ユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/  
AmazonCognitoPowerUser
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AttachUserPolicy](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
            user_name)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AttachUserPolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Attaches a policy to a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_user(user_name, policy_arn)
  @iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error attaching policy to user: #{e.message}")
  false
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AttachUserPolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn attach_user_policy(
  client: &iamClient,
  user_name: &str,
  policy_arn: &str,
) -> Result<(), iamError> {
  client
    .attach_user_policy()
    .user_name(user_name)
    .policy_arn(policy_arn)
    .send()
    .await?;
```

```
    Ok(())  
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[AttachUserPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ChangePassword** を使用する

以下のコード例は、ChangePassword の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーのパスワードを変更するには

IAM ユーザーのパスワードを変更するには、`--cli-input-json` パラメータを使用して、古いパスワードと新しいパスワードを含む JSON ファイルを渡すことをお勧めします。この方法を使用すると、英数字以外の文字を含む強力なパスワードを使用できます。英数字以外の文字を含むパスワードをコマンドラインパラメータとして渡す場合、そのパスワードの使用が難しい場合があります。`--cli-input-json` パラメータを使用するには、まず次の例のように、`change-password` コマンドと `--generate-cli-skeleton` パラメータを組み合わせ使用します。

```
aws iam change-password \  
  --generate-cli-skeleton > change-password.json
```

前のコマンドは `change-password.json` という名前の JSON ファイルを作成し、これを使用して古いパスワードと新しいパスワードを入力できます。例えば、ファイルは次のようになります。

```
{  
  "OldPassword": "3s0K_;xh4~8XXI",  
  "NewPassword": "]35d/{pB9Fo9wJ"
```

```
}
```

次に、パスワードを変更するには、change-password コマンドをもう一度使用します。今度は、JSON ファイルを指定する --cli-input-json パラメータを渡します。次の change-password コマンドでは、change-password.json という名前の JSON ファイルと共に --cli-input-json パラメータを使用します。

```
aws iam change-password \  
  --cli-input-json file://change-password.json
```

このコマンドでは何も出力されません。

このコマンドは、IAM ユーザーのみが呼び出すことができます。AWS アカウント (ルート) 認証情報を使用してこのコマンドを呼び出すと、コマンドは InvalidUserType エラーを返します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーが自分のパスワードを変更する方法](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ChangePassword](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、コマンドを実行しているユーザーのパスワードを変更します。このコマンドは、IAM ユーザーのみが呼び出すことができます。AWS アカウント (ルート) 認証情報を使用してサインインしているときにこのコマンドが呼び出されると、コマンドは **InvalidUserType** エラーを返します。

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ChangePassword](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `CreateAccessKey` を使用する

以下のコード例は、`CreateAccessKey` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)
- [アクセスキーの管理](#)

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create an IAM access key for a user.
/// </summary>
/// <param name="userName">The username for which to create the IAM access
/// key.</param>
/// <returns>The AccessKey.</returns>
public async Task<AccessKey> CreateAccessKeyAsync(string userName)
{
    var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
    {
        UserName = userName,
    });

    return response.AccessKey;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#     -u user_name -- The name of the IAM user.
#     [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#     [access_key_id access_key_secret]
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user_access_key() {
    local user_name file_name response
```

```
local option OPTARG # Required to use getopt command in a function.

# bashsupport disable=BP5008
function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name    The name of the IAM user."
    echo "  [-f file_name]  Optional file name for the access key output."
    echo ""
}

# Retrieve the calling parameters.
while getopt "u:f:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        f) file_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
```

```
    return 1
fi

if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
fi

local key_id key_secret
# shellcheck disable=SC2086
key_id=$(echo $response | cut -f 2 -d ' ')
# shellcheck disable=SC2086
key_secret=$(echo $response | cut -f 4 -d ' ')

echo "$key_id $key_secret"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateAccessKey](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
```



```
Aws::IAM::Model::CreateAccessKeyOutcome outcome =
iam.CreateAccessKey(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating access key for IAM user " << userName
                << ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    const auto &accessKey = outcome.GetResult().GetAccessKey();
    std::cout << "Successfully created access key for IAM user " <<
                userName << std::endl << "  aws_access_key_id = " <<
                accessKey.GetAccessKeyId() << std::endl <<
                "  aws_secret_access_key = " << accessKey.GetSecretAccessKey()
    <<
                std::endl;
    result = accessKey.GetAccessKeyId();
}

return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーのアクセスキーを作成するには

次の `create-access-key` コマンドは、Bob という名前の IAM ユーザーのためにアクセスキー (アクセスキー ID とシークレットアクセスキー) を作成します。

```
aws iam create-access-key \
  --user-name Bob
```

出力:

```
{
  "AccessKey": {
    "UserName": "Bob",
    "Status": "Active",
```

```
    "CreateDate": "2015-03-09T18:39:23.411Z",
    "SecretAccessKey": "wJa1rXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```


シークレットアクセスキーを安全な場所に保管します。紛失した場合は回復できないため、新しいアクセスキーを作成する必要があります。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateAccessKey](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    IamClient *iam.Client
}

// CreateAccessKeyPair creates an access key for a user. The returned access key
// contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
(*types.AccessKey, error) {
    var key *types.AccessKey
    result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
    &iam.CreateAccessKeyInput{
```

```
    UserName: aws.String(userName)}})
if err != nil {
    log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
        userName, err)
} else {
    key = result.AccessKey
}
return key, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CreateAccessKey {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
        <user>\s

        Where:
        user - An AWS IAM user that you can obtain from the AWS
Management Console.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String user = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String keyId = createIAMAccessKey(iam, user);
    System.out.println("The Key Id is " + keyId);
    iam.close();
}

public static String createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アクセスキーを作成します。

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ Username: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## Kotlin

### SDK for Kotlin

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、新しいアクセスキーとシークレットアクセスキーのペアを作成し、それをユーザー **David** に割り当てます。**SecretAccessKey** を取得できるのはこのときだけなので、**AccessKeyId** と **SecretAccessKey** の値は必ずファイルに保存してください。後で取得することはできません。シークレットアクセスキーを紛失した場合は、新しいアクセスキーペアを作成する必要があります。

```
New-IAMAccessKey -UserName David
```

出力:

```
AccessKeyId      : AKIAIOSFODNN7EXAMPLE
CreateDate       : 4/13/2015 1:00:42 PM
SecretAccessKey  : wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Status           : Active
UserName        : David
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateAccessKey](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateAccessKey](#)」を参照してください。



## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
```

```
# @return [Boolean]
def create_access_key(user_name)
  response = @iam_client.create_access_key(user_name: user_name)
  access_key = response.access_key
  @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
  access_key
rescue Aws::IAM::Errors::LimitExceeded => e
  @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
  nil
rescue StandardError => e
  @logger.error("Error creating access key: #{e.message}")
  nil
end

# Deactivates an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def deactivate_access_key(user_name, access_key_id)
  @iam_client.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,
    status: "Inactive"
  )
  true
rescue StandardError => e
  @logger.error("Error deactivating access key: #{e.message}")
  false
end

# Deletes an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def delete_access_key(user_name, access_key_id)
  @iam_client.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  true
end
```

```
rescue StandardError => e
  @logger.error("Error deleting access key: #{e.message}")
  false
end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
Result<AccessKey, iamError> {
  let mut tries: i32 = 0;
  let max_tries: i32 = 10;

  let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
loop {
  match client.create_access_key().user_name(user_name).send().await {
    Ok(inner_response) => {
      break Ok(inner_response);
    }
    Err(e) => {
      tries += 1;
      if tries > max_tries {
        break Err(e);
      }
      sleep(Duration::from_secs(2)).await;
    }
  }
};
```

```
Ok(response.unwrap().access_key.unwrap())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[CreateAccessKey](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func createAccessKey(userName: String) async throws ->
IAMClientTypes.AccessKey {
    let input = CreateAccessKeyInput(
        userName: userName
    )
    do {
        let output = try await iamClient.createAccessKey(input: input)
        guard let accessKey = output.accessKey else {
            throw ServiceHandlerError.keyError
        }
        return accessKey
    } catch {
        throw error
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[CreateAccessKey](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateAccountAlias** を使用する

以下のコード例は、CreateAccountAlias の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

C++

SDK for C++

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
```

```
std::cout << "Successfully created account alias " << aliasName <<
    std::endl;
}

return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## CLI

### AWS CLI

アカウントエイリアスを作成するには

次の `create-account-alias` コマンドは、AWS アカウントのエイリアス `examplecorp` を作成します。

```
aws iam create-account-alias \
    --account-alias examplecorp
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウント ID とそのエイリアス](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateAccountAlias](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example,
myawsaccount).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
```

```
        CreateAccountAliasRequest request =
CreateAccountAliasRequest.builder()
    .accountAlias(alias)
    .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを作成します。

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
```



```
const command = new CreateAccountAliasCommand({
  AccountAlias: alias,
});

return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウントのアカウントエイリアスを **mycompanyaws** に変更します。ユーザーログインページのアドレスが、<https://mycompanyaws.signin.aws.amazon.com/console> に変わります。エイリアスの代わりにアカウント ID 番号を使用する元の URL (<https://<accountidnumber>.signin.aws.amazon.com/console>) は引き続き機能します。ただし、以前に定義したエイリアススペースの URL は機能しなくなります。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateAccountAlias](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
    the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end

    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Error listing account aliases: #{e.message}")
    end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  end
end
```

```
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating account alias: #{e.message}")
  false
end

# Deletes an AWS account alias.
#
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
def delete_account_alias(account_alias)
  @iam_client.delete_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateGroup** を使用する

以下のコード例は、CreateGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create an IAM group.
/// </summary>
/// <param name="groupName">The name to give the IAM group.</param>
/// <returns>The IAM group that was created.</returns>
public async Task<Group> CreateGroupAsync(string groupName)
{
    var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
    { GroupName = groupName });
    return response.Group;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[CreateGroup](#)」を参照してください。

## CLI

### AWS CLI

IAM グループを作成するには

次の create-group コマンドは、Admins という名前の IAM グループを作成します。

```
aws iam create-group \
  --group-name Admins
```

出力:

```
{
```

```
"Group": {
  "Path": "/",
  "CreateDate": "2015-03-09T20:30:24.940Z",
  "GroupId": "AIDGPMS9R04H3FEXAMPLE",
  "Arn": "arn:aws:iam::123456789012:group/Admins",
  "GroupName": "Admins"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateGroup](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Developers** という名前の新しい IAM グループを作成します。

```
New-IAMGroup -GroupName Developers
```

出力:

```
Arn          : arn:aws:iam::123456789012:group/Developers
CreateDate   : 4/14/2015 11:21:31 AM
GroupId      : QNEJ5PM4NFSQCEXAMPLE1
GroupName    : Developers
Path         : /
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateInstanceProfile** を使用する

以下のコード例は、CreateInstanceProfile の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)



## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create a policy, role, and profile that is associated with instances with
a specified name.
/// An instance's associated profile defines a role that is assumed by the
/// instance.The role has attached policies that specify the AWS permissions
granted to
/// clients that run on the instance.
/// </summary>
/// <param name="policyName">Name to use for the policy.</param>
/// <param name="roleName">Name to use for the role.</param>
/// <param name="profileName">Name to use for the profile.</param>
/// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
/// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
/// <returns>The Arn of the profile.</returns>
public async Task<string> CreateInstanceProfileWithName(
    string policyName,
    string roleName,
    string profileName,
    string ssmOnlyPolicyFile,
    List<string>? awsManagedPolicies = null)
{
    var assumeRoleDoc = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": [" +
                "\"ec2.amazonaws.com\"" +
            "]" +
        "}], " +
    "}," +
```

```
                "\"Action\": \"sts:AssumeRole\"\" +
                "]" +
            "});

var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

var policyArn = "";

try
{
    var createPolicyResult = await _amazonIam.CreatePolicyAsync(
        new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
    policyArn = createPolicyResult.Policy.Arn;
}
catch (EntityAlreadyExistsException)
{
    // The policy already exists, so we look it up to get the Arn.
    var policiesPaginator = _amazonIam.Paginators.ListPolicies(
        new ListPoliciesRequest()
        {
            Scope = PolicyScopeType.Local
        });
    // Get the entire list using the paginator.
    await foreach (var policy in policiesPaginator.Policies)
    {
        if (policy.PolicyName.Equals(policyName))
        {
            policyArn = policy.Arn;
        }
    }

    if (policyArn == null)
    {
        throw new InvalidOperationException("Policy not found");
    }
}

try
{
    await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
```

```
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = assumeRoleDoc,
        });
        await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
        {
            RoleName = roleName,
            PolicyArn = policyArn
        });
        if (awsManagedPolicies != null)
        {
            foreach (var awsPolicy in awsManagedPolicies)
            {
                await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                {
                    PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                    RoleName = roleName
                });
            }
        }
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Role already exists.");
    }

    string profileArn = "";
    try
    {
        var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
            new CreateInstanceProfileRequest()
            {
                InstanceProfileName = profileName
            });
        // Allow time for the profile to be ready.
        profileArn = profileCreateResponse.InstanceProfile.Arn;
        Thread.Sleep(10000);
        await _amazonIam.AddRoleToInstanceProfileAsync(
            new AddRoleToInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
    }
}
```

```
        });

    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Policy already exists.");
        var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
            new GetInstanceProfileRequest()
            {
                InstanceProfileName = profileName
            });
        profileArn = profileGetResponse.InstanceProfile.Arn;
    }
    return profileArn;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[CreateInstanceProfile](#)」を参照してください。

## CLI

### AWS CLI

インスタンスプロファイルを作成するには

次の `create-instance-profile` コマンドは、`Webserver` という名前のインスタンスプロファイルを作成します。

```
aws iam create-instance-profile \  
    --instance-profile-name Webserver
```

出力:

```
{  
  "InstanceProfile": {  
    "InstanceId": "AIPAJMBC7DLSPEXAMPLE",  
    "Roles": [],  
    "CreateDate": "2015-03-09T20:33:19.626Z",  
    "InstanceProfileName": "Webserver",  
    "Path": "/",
```

```
    "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
  }
}
```

インスタンスプロファイルにロールを追加するには、`add-role-to-instance-profile` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateInstanceProfile](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateInstanceProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**ProfileForDevEC2Instance** という名前の新しい IAM インスタンスプロファイルを作成します。**Add-IAMRoleToInstanceProfile** コマンドを個別に実行して、インスタンスにアクセス許可を与える既存の IAM ロールにインスタンスプロファイルを関連付ける必要があります。最後に、EC2 インスタンスを起動する際に、インスタンスプロファイル EC2 インスタンスにアタッチします。そのためには、**New-EC2Instance** コマンドレットを **InstanceProfile\_Arn** または **InstanceProfile\_Name** パラメータと共に使用します。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

出力:

```
Arn                : arn:aws:iam::123456789012:instance-profile/  
ProfileForDevEC2Instance  
CreateDate         : 4/14/2015 11:31:39 AM  
InstanceProfileId  : DYMFXL556EY46EXAMPLE1  
InstanceProfileName : ProfileForDevEC2Instance  
Path               : /  
Roles              : {}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateInstanceProfile](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

この例では、ポリシー、ロール、インスタンスプロファイルを作成し、それらをすべてリンクします。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
            created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"
```

```
def create_instance_profile(
    self, policy_file, policy_name, role_name, profile_name,
    aws_managed_policies=()
):
    """
    Creates a policy, role, and profile that is associated with instances
    created by
    this class. An instance's associated profile defines a role that is
    assumed by the
    instance. The role has attached policies that specify the AWS permissions
    granted to
    clients that run on the instance.

    :param policy_file: The name of a JSON file that contains the policy
    definition to
        create and attach to the role.
    :param policy_name: The name to give the created policy.
    :param role_name: The name to give the created role.
    :param profile_name: The name to the created profile.
    :param aws_managed_policies: Additional AWS-managed policies that are
    attached to
        the role, such as
    AmazonSSMManagedInstanceCore to grant
        use of Systems Manager to send commands to
    the instance.
    :return: The ARN of the profile that is created.
    """
    assume_role_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": "ec2.amazonaws.com"},
                "Action": "sts:AssumeRole",
            }
        ],
    }
    with open(policy_file) as file:
        instance_policy_doc = file.read()

    policy_arn = None
    try:
        pol_response = self.iam_client.create_policy(
```



```
        PolicyName=policy_name, PolicyDocument=instance_policy_doc
    )
    policy_arn = pol_response["Policy"]["Arn"]
    log.info("Created policy with ARN %s.", policy_arn)
except ClientError as err:
    if err.response["Error"]["Code"] == "EntityAlreadyExists":
        log.info("Policy %s already exists, nothing to do.", policy_name)
        list_pol_response = self.iam_client.list_policies(Scope="Local")
        for pol in list_pol_response["Policies"]:
            if pol["PolicyName"] == policy_name:
                policy_arn = pol["Arn"]
                break
    if policy_arn is None:
        raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

    try:
        self.iam_client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(assume_role_doc)
        )
        self.iam_client.attach_role_policy(RoleName=role_name,
            PolicyArn=policy_arn)
        for aws_policy in aws_managed_policies:
            self.iam_client.attach_role_policy(
                RoleName=role_name,
                PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
            )
        log.info("Created role %s and attached policy %s.", role_name,
            policy_arn)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            log.info("Role %s already exists, nothing to do.", role_name)
        else:
            raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

    try:
        profile_response = self.iam_client.create_instance_profile(
            InstanceProfileName=profile_name
        )
        waiter = self.iam_client.get_waiter("instance_profile_exists")
        waiter.wait(InstanceProfileName=profile_name)
        time.sleep(10) # wait a little longer
        profile_arn = profile_response["InstanceProfile"]["Arn"]
```

```
        self.iam_client.add_role_to_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )
        log.info("Created profile %s and added role %s.", profile_name,
role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            prof_response = self.iam_client.get_instance_profile(
                InstanceProfileName=profile_name
            )
            profile_arn = prof_response["InstanceProfile"]["Arn"]
            log.info(
                "Instance profile %s already exists, nothing to do.",
profile_name
            )
        else:
            raise AutoScalerError(
                f"Couldn't create profile {profile_name} and attach it to
role\n"
                f"{role_name}: {err}")
    return profile_arn
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateInstanceProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateLoginProfile** を使用する

以下のコード例は、CreateLoginProfile の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーのパスワードを作成するには

IAM ユーザーのパスワードを作成するには、`--cli-input-json` パラメータを使用して、パスワードを含む JSON ファイルを渡すことをお勧めします。この方法を使用すると、英数字以外の文字を含む強力なパスワードを作成できます。英数字以外の文字を含むパスワードをコマンドラインパラメータとして渡す場合、そのパスワードの作成が難しい場合があります。

`--cli-input-json` パラメータを使用するには、まず次の例のように、`create-login-profile` コマンドと `--generate-cli-skeleton` パラメータを組み合わせて使用します。

```
aws iam create-login-profile \  
  --generate-cli-skeleton > create-login-profile.json
```

前のコマンドでは、`create-login-profile.json` という名前の JSON ファイルが作成されます。このファイルを使用して、後続の `create-login-profile` コマンドの情報を入力できます。例:

```
{  
  "UserName": "Bob",  
  "Password": "&1-3a6u:RA0djs",  
  "PasswordResetRequired": true  
}
```

次に、IAM ユーザーのパスワードを作成するには、`create-login-profile` コマンドをもう一度使用します。今度は、JSON ファイルを指定する `--cli-input-json` パラメータを渡します。次の `create-login-profile` コマンドでは、`create-login-profile.json` という名前の JSON ファイルと共に `--cli-input-json` パラメータを使用します。

```
aws iam create-login-profile \  
  --cli-input-json file://create-login-profile.json
```

出力:

```
{  
  "LoginProfile": {  
    "UserName": "Bob",  
    "CreateDate": "2015-03-10T20:55:40.274Z",  
    "PasswordResetRequired": true  
  }  
}
```

新しいパスワードがアカウントのパスワードポリシーに違反している場合、コマンドは PasswordPolicyViolation エラーを返します。

既にパスワードを持っているユーザーのパスワードを変更するには、update-login-profile を使用します。アカウントのパスワードポリシーを設定するには、update-account-password-policy コマンドを使用します。

アカウントのパスワードポリシーで許可されている場合、IAM ユーザーは change-password コマンドを使用して自分のパスワードを変更できます。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのパスワードの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateLoginProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、Bob という名前の IAM ユーザーの (一時的な) パスワードを作成し、次回 **Bob** がサインインしたときに、ユーザーにパスワードを変更するように要求するフラグを設定しています。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

出力:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
4/14/2015 12:26:30 PM	True	Bob

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateLoginProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `CreateOpenIdConnectProvider` を使用する

以下のコード例は、`CreateOpenIdConnectProvider` の使用方法を示しています。

### CLI

#### AWS CLI

OpenID Connect (OIDC) プロバイダーを作成するには

OpenID Connect (OIDC) プロバイダーを作成するには、`--cli-input-json` パラメータを使用して、必要なパラメータを含む JSON ファイルを渡すことをお勧めします。OIDC プロバイダーを作成するときは、プロバイダーの URL を渡す必要があり、URL は `https://` で始まる必要があります。コマンドライン環境によっては、コロン (`:`) とフォワードスラッシュ (`/`) の文字が特別な意味を持つため、URL をコマンドラインパラメータとして渡すのは難しい場合があります。この `--cli-input-json` パラメータを使用すると、この制限を回避できます。

`--cli-input-json` パラメータを使用するには、まず次の例のように、`create-open-id-connect-provider` コマンドと `--generate-cli-skeleton` パラメータを組み合わせて使用します。

```
aws iam create-open-id-connect-provider \  
  --generate-cli-skeleton > create-open-id-connect-provider.json
```

前のコマンドでは、`create-open-id-connect-provider.json` という名前の JSON ファイルが作成されます。このファイルを使用して、後続の `create-open-id-connect-provider` コマンドの情報を入力できます。例:

```
{  
  "Url": "https://server.example.com",  
  "ClientIDList": [  
    "example-application-ID"  
  ],  
  "ThumbprintList": [  
    "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"  
  ]  
}
```

次に、OpenID Connect (OIDC) プロバイダーを作成するには、`create-open-id-connect-provider` コマンドをもう一度使用します。今度は、JSON ファイルを指定する `--cli-`

input-json パラメータを渡します。次の create-open-id-connect-provider コマンドでは、create-open-id-connect-provider.json という名前の JSON ファイルと共に --cli-input-json パラメータを使用します。

```
aws iam create-open-id-connect-provider \  
  --cli-input-json file://create-open-id-connect-provider.json
```

出力:

```
{  
  "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/  
server.example.com"  
}
```

OIDC プロバイダーの詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

OIDC プロバイダーのサムプリントの取得に関する詳細については、「AWS IAM ユーザーガイド」の「[OpenID Connect ID プロバイダーのサムプリントを取得する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateOpenIdConnectProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、URL <https://example.oidcprovider.com> とクライアント ID **my-testapp-1** にある OIDC 互換プロバイダーサービスに関連付けられた IAM OIDC プロバイダーを作成します。OIDC プロバイダーがサムプリントを提供します。サムプリントを認証するには、<http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html> にある手順に従ってください。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList  
my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

出力:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateOpenIdConnectProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreatePolicy** を使用する

以下のコード例は、CreatePolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)
- [ポリシーを管理](#)
- [IAM Policy Builder API を使用する](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create an IAM policy.
/// </summary>
/// <param name="policyName">The name to give the new IAM policy.</param>
/// <param name="policyDocument">The policy document for the new policy.</
param>
/// <returns>The new IAM policy object.</returns>
public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
```

```
{
    var response = await _IAMService.CreatePolicyAsync(new
CreatePolicyRequest
    {
        PolicyDocument = policyDocument,
        PolicyName = policyName,
    });

    return response.Policy;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
```



```

# -n policy_name -- The name of the IAM policy.
# -p policy_json -- The policy document.
#
# Returns:
# 0 - If successful.
# 1 - If it fails.
#####
function iam_create_policy() {
    local policy_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_policy"
        echo "Creates an AWS Identity and Access Management (IAM) policy."
        echo " -n policy_name The name of the IAM policy."
        echo " -p policy_json -- The policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) policy_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$policy_name" ]]; then
        errecho "ERROR: You must provide a policy name with the -n parameter."
        usage
        return 1
    fi
}

```

```
if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
fi

echo "$response"
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreatePolicy](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                       const Aws::String &srcArn,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfig) {
```

```

Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreatePolicyRequest request;
request.SetPolicyName(policyName);
request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
Aws::String result;
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating policy " << policyName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    result = outcome.GetResult().GetPolicy().GetArn();
    std::cout << "Successfully created policy " << policyName <<
        std::endl;
}

return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
        << "  \"Version\": \"2012-10-17\", "
        << "  \"Statement\": ["
        << "    {"
        << "      \"Effect\": \"Allow\", "
        << "      \"Action\": \"logs:CreateLogGroup\", "
        << "      \"Resource\": \""
        << rsrc_arn
        << "\" "
        << "    }, "
        << "    {"
        << "      \"Effect\": \"Allow\", "
        << "      \"Action\": ["
        << "        \"dynamodb:DeleteItem\", "
        << "        \"dynamodb:GetItem\", "
        << "        \"dynamodb:PutItem\", "
        << "        \"dynamodb:Scan\", "
        << "        \"dynamodb:UpdateItem\" "
        << "      ], "
        << "      \"Resource\": \""
        << rsrc_arn

```

```
        << "\\\"
        << "    }"
        << "  ]"
        << "}";

    return stringstream.str();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[CreatePolicy](#)」を参照してください。

## CLI

### AWS CLI

例 1: カスタマー管理ポリシーを作成するには

次のコマンドは、my-policy という名前でカスタマー管理ポリシーを作成します。

```
aws iam create-policy \
  --policy-name my-policy \
  --policy-document file://policy
```

このファイル policy は、現在のフォルダにある JSON ドキュメントで、my-bucket という名前の Amazon S3 バケット内の shared フォルダに対する読み取り専用アクセスを付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket/shared/*"
      ]
    }
  ]
}
```

```
}
```

出力:

```
{
  "Policy": {
    "PolicyName": "my-policy",
    "CreateDate": "2015-06-01T19:31:18.620Z",
    "AttachmentCount": 0,
    "IsAttachable": true,
    "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
    "DefaultVersionId": "v1",
    "Path": "/",
    "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
    "UpdateDate": "2015-06-01T19:31:18.620Z"
  }
}
```

文字列パラメータの入力としてファイルを使用する方法の詳細については、「AWS CLI ユーザーガイド」の「[AWS CLI のパラメータ値を指定する](#)」を参照してください。

例 2: 説明を含むカスタマー管理ポリシーを作成するには

次のコマンドは、イミュータブルな説明を使用して my-policy という名前のカスタマー管理ポリシーを作成します。

```
aws iam create-policy \
  --policy-name my-policy \
  --policy-document file://policy.json \
  --description "This policy grants access to all Put, Get, and List actions
for my-bucket"
```

このファイル policy.json は、my-bucket という名前の Amazon S3 バケットに対するすべての Put、List、および Get アクションへのアクセスを付与する、現在のフォルダにある JSON ドキュメントです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
        "Action": [
            "s3:ListBucket*",
            "s3:PutBucket*",
            "s3:GetBucket*"
        ],
        "Resource": [
            "arn:aws:s3:::my-bucket"
        ]
    }
]
```

出力:

```
{
  "Policy": {
    "PolicyName": "my-policy",
    "PolicyId": "ANPAWGSUGIDPEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:policy/my-policy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2023-05-24T22:38:47+00:00",
    "UpdateDate": "2023-05-24T22:38:47+00:00"
  }
}
```

アイデンティティベースのポリシーの詳細については、「AWS IAM ユーザーガイド」の「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。

例 3: タグを使用してカスタマー管理ポリシーを作成するには

次のコマンドは、タグを使用して my-policy という名前のカスタマー管理ポリシーを作成します。この例では、次の JSON 形式のタグを持つ --tags パラメータフラグを使用します: '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。あるいは、--tags フラグを次の短縮形式のタグとともに使用することもできます: 'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-policy \
```

```
--policy-name my-policy \  
--policy-document file://policy.json \  
--tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",  
"Value": "Seattle"}'
```

このファイル `policy.json` は、`my-bucket` という名前の Amazon S3 バケットに対するすべての Put、List、および Get アクションへのアクセスを付与する、現在のフォルダにある JSON ドキュメントです。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:ListBucket*",  
        "s3:PutBucket*",  
        "s3:GetBucket*"  
      ],  
      "Resource": [  
        "arn:aws:s3:::my-bucket"  
      ]  
    }  
  ]  
}
```

出力:

```
{  
  "Policy": {  
    "PolicyName": "my-policy",  
    "PolicyId": "ANPAWGSUGIDPEXAMPLE",  
    "Arn": "arn:aws:iam::12345678012:policy/my-policy",  
    "Path": "/",  
    "DefaultVersionId": "v1",  
    "AttachmentCount": 0,  
    "PermissionsBoundaryUsageCount": 0,  
    "IsAttachable": true,  
    "CreateDate": "2023-05-24T23:16:39+00:00",  
    "UpdateDate": "2023-05-24T23:16:39+00:00",  
    "Tags": [  
      {  
        "Key": "Department",  
        "Value": "Accounting"  
      },  
      {  
        "Key": "Location",  
        "Value": "Seattle"  
      }  
    ]  
  }  
}
```

```
        "Key": "Department",
        "Value": "Accounting"
    },
    "Key": "Location",
    "Value": "Seattle"
}
]
```

ポリシーのタグ付けの詳細については、「AWS IAM ユーザーガイド」の「[カスタマー管理ポリシーのタグ付け](#)」を参照してください。

- APIの詳細については、「AWS CLI Command Reference」の「[CreatePolicy](#)」を参照してください。

Go

SDK for Go V2

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    IamClient *iam.Client
}

// CreatePolicy creates a policy that grants a list of actions to the specified
resource.
// PolicyDocument shows how to work with a policy document as a data structure
and
```



```
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
    resourceArn string) (*types.Policy, error) {
    var policy *types.Policy
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Action: actions,
            Resource: aws.String(resourceArn),
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
            resourceArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
        &iam.CreatePolicyInput{
            PolicyDocument: aws.String(string(policyBytes)),
            PolicyName: aws.String(policyName),
        })
    if err != nil {
        log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Java

## SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"dynamodb:DeleteItem\", " +
        "        \"dynamodb:GetItem\", " +
        "        \"dynamodb:PutItem\", " +
        "        \"dynamodb:Scan\", " +
        "        \"dynamodb:UpdateItem\"" +
```

```
        ]," +
        "\"Resource\": \"*\")\" +
    }" +
    "]" +
    "}]";

public static void main(String[] args) {

    final String usage = ""
        Usage:
            CreatePolicy <policyName>\s

        Where:
            policyName - A unique policy name.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String policyName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMPolicy(iam, policyName);
    System.out.println("Successfully created a policy with this ARN value: "
+ result);
    iam.close();
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument)
            .build();

        CreatePolicyResponse response = iam.createPolicy(request);
```

```
        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreatePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーを作成します。

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
```

```
* @param {string} policyName
*/
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreatePolicy](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
```

```
Version: "2012-10-17",
Statement: [
  {
    Effect: "Allow",
    Action: "logs:CreateLogGroup",
    Resource: "RESOURCE_ARN",
  },
  {
    Effect: "Allow",
    Action: [
      "dynamodb:DeleteItem",
      "dynamodb:GetItem",
      "dynamodb:PutItem",
      "dynamodb:Scan",
      "dynamodb:UpdateItem",
    ],
    Resource: "RESOURCE_ARN",
  },
],
];

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Kotlin

## SDK for Kotlin

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "      \"Effect\": \"Allow\"," +
            "      \"Action\": [" +
            "        \"dynamodb:DeleteItem\"," +
            "        \"dynamodb:GetItem\"," +
            "        \"dynamodb:PutItem\"," +
            "        \"dynamodb:Scan\"," +
            "        \"dynamodb:UpdateItem\"" +
            "      ]," +
            "      \"Resource\": \"*\\"" +
            "    }" +
            "  ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreatePolicy](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3:*:*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_{$uuid}",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

public function createPolicy(string $policyName, string $policyDocument)
{
    $result = $this->customWaiter(function () use ($policyName,
    $policyDocument) {
        return $this->iamClient->createPolicy([
            'PolicyName' => $policyName,
            'PolicyDocument' => $policyDocument,
        ]);
    });
    return $result['Policy'];
}
```



- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[CreatePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MySamplePolicy** という名前の現在の AWS アカウントに新しい IAM ポリシーを作成します。ファイル **MySamplePolicy.json** は、ポリシーの内容を提供します。JSON ポリシーファイルを正常に処理するには、**-Raw** switch パラメータを使用する必要がありますことに注意してください。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw MySamplePolicy.json)
```

出力:

```
Arn          : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount : 0
CreateDate   : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : LD4KP6HVFE7WGEXAMPLE1
PolicyName  : MySamplePolicy
UpdateDate  : 4/14/2015 2:45:59 PM
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreatePolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
    policy
                        applies to. This ARN can contain wildcards, such as
                        'arn:aws:s3::my-bucket/*' to allow actions on all
    objects
                        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
end
```

```
# @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
def get_policy(policy_arn)
  response = @iam_client.get_policy(policy_arn: policy_arn)
  policy = response.policy
  @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

# Attaches a policy to a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_role(role_name, policy_arn)
  @iam_client.attach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error attaching policy to role: #{e.message}")
  false
end

# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

```
# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def detach_policy_from_role(role_name, policy_arn)
  @iam_client.detach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error detaching policy from role: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_policy(
  client: &iamClient,
  policy_name: &str,
  policy_document: &str,
) -> Result<Policy, iamError> {
  let policy = client
    .create_policy()
    .policy_name(policy_name)
    .policy_document(policy_document)
```

```
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[CreatePolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
    let input = CreatePolicyInput(
        policyDocument: policyDocument,
        policyName: name
    )
    do {
        let output = try await iamClient.createPolicy(input: input)
        guard let policy = output.policy else {
            throw ServiceHandlerError.noSuchPolicy
        }
        return policy
    } catch {
        throw error
    }
}
```

```
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[CreatePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `CreatePolicyVersion` を使用する

以下のコード例は、`CreatePolicyVersion` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)

### CLI

#### AWS CLI

新しいバージョンの管理ポリシーを作成するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MyPolicy` である IAM ポリシーの新しい v2 バージョンを作成し、それをデフォルトのバージョンにします。

```
aws iam create-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --policy-document file://NewPolicyVersion.json \  
  --set-as-default
```

出力:

```
{  
  "PolicyVersion": {  
    "CreateDate": "2015-06-16T18:56:03.721Z",  
    "VersionId": "v2",
```

```

    "IsDefaultVersion": true
  }
}

```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ポリシーのバージョニング](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreatePolicyVersion](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:policy/MyPolicy` である IAM ポリシーの新しい「v2」バージョンを作成し、それをデフォルトのバージョンにします。`NewPolicyVersion.json` ファイルは、ポリシーの内容を提供します。JSON ポリシーファイルを正常に処理するには、`-Raw` switch パラメータを使用する必要があることに注意してください。

```

New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true

```

出力:

CreateDate	VersionId	Document	IsDefaultVersion
-----	-----	-----	-----
4/15/2015 10:54:54 AM	v2		True

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreatePolicyVersion](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
```

```
return policy_version
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreatePolicyVersion](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateRole** を使用する

以下のコード例は、CreateRole の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [ロールの管理](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create a new IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="rolePolicyDocument">The name of the IAM policy document
/// for the new role.</param>
```

```
/// <returns>The Amazon Resource Name (ARN) of the role.</returns>
public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePolicyDocument,
    };

    var response = await _IAMService.CreateRoleAsync(request);
    return response.Role.Arn;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[CreateRole](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_role
#
```

```
# This function creates an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_json -- The assume role policy document.
#
# Returns:
#     The ARN of the role.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_role() {
    local role_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_json -- The assume role policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$role_name" ]]; then
```

```
errecho "ERROR: You must provide a role name with the -n parameter."
usage
return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi


echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateRole](#)」を参照してください。

## C++

## SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::createIamRole(
    const Aws::String &roleName,
    const Aws::String &policy,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[CreateRole](#)」を参照してください。

## CLI

## AWS CLI

例 1: IAM ロールを作成するには

次の `create-role` コマンドは、`Test-Role` という名前のロールを作成し、それに信頼ポリシーをアタッチします。

```
aws iam create-role \  
  --role-name Test-Role \  
  --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

出力:

```
{  
  "Role": {  
    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",  
    "RoleId": "AKIAIOSFODNN7EXAMPLE",  
    "CreateDate": "2013-06-07T20:43:32.821Z",  
    "RoleName": "Test-Role",  
    "Path": "/",  
    "Arn": "arn:aws:iam::123456789012:role/Test-Role"  
  }  
}
```

信頼ポリシーは、`Test-Role-Trust-Policy.json` ファイル内で JSON ドキュメントとして定義されます。(ファイル名と拡張子には意味はありません。) 信頼ポリシーはプリンシパルを指定する必要があります。

アクセス許可ポリシーをロールにアタッチするには、`put-role-policy` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

例 2: 最大セッション期間を指定して IAM ロールを作成するには

次の `create-role` コマンドは、`Test-Role` という名前のロールを作成し、最大セッション時間を 7,200 秒 (2 時間) に設定します。

```
aws iam create-role \  
  --role-name Test-Role \  
  --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

```
--role-name Test-Role \  
--assume-role-policy-document file://Test-Role-Trust-Policy.json \  
--max-session-duration 7200
```

出力:

```
{  
  "Role": {  
    "Path": "/",  
    "RoleName": "Test-Role",  
    "RoleId": "AKIAIOSFODNN7EXAMPLE",  
    "Arn": "arn:aws:iam::12345678012:role/Test-Role",  
    "CreateDate": "2023-05-24T23:50:25+00:00",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Sid": "Statement1",  
          "Effect": "Allow",  
          "Principal": {  
            "AWS": "arn:aws:iam::12345678012:root"  
          },  
          "Action": "sts:AssumeRole"  
        }  
      ]  
    }  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[ロールの最大セッション時間の変更 \(AWS API\)](#)」を参照してください。

例 3: タグを使用して IAM ロールを作成するには

次のコマンドは、タグを使用して IAM ロール Test-Role を作成します。この例では、次の JSON 形式のタグを持つ --tags パラメータフラグを使用します: '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。あるいは、--tags フラグを次の短縮形式のタグとともに使用することもできます: 'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-role \  

```



```
--role-name Test-Role \  
--assume-role-policy-document file://Test-Role-Trust-Policy.json \  
--tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",  
"Value": "Seattle"}'
```

出力:


```
{  
  "Role": {  
    "Path": "/",  
    "RoleName": "Test-Role",  
    "RoleId": "AKIAIOSFODNN7EXAMPLE",  
    "Arn": "arn:aws:iam::123456789012:role/Test-Role",  
    "CreateDate": "2023-05-25T23:29:41+00:00",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Sid": "Statement1",  
          "Effect": "Allow",  
          "Principal": {  
            "AWS": "arn:aws:iam::123456789012:root"  
          },  
          "Action": "sts:AssumeRole"  
        }  
      ]  
    },  
    "Tags": [  
      {  
        "Key": "Department",  
        "Value": "Accounting"  
      },  
      {  
        "Key": "Location",  
        "Value": "Seattle"  
      }  
    ]  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」で「[IAM ロールのタグ付け](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateRole](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// CreateRole creates a role that trusts a specified user. The trusted user can
// assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
(*types.Role, error) {
    var role *types.Role
    trustPolicy := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Principal: map[string]string{"AWS": trustedUserArn},
            Action: []string{"sts:AssumeRole"},
        }},
    }
    policyBytes, err := json.Marshal(trustPolicy)
```

```
if err != nil {
    log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
trustedUserArn, err)
    return nil, err
}
result, err := wrapper.IamClient.CreateRole(context.TODO(),
&iam.CreateRoleInput{
    AssumeRolePolicyDocument: aws.String(string(policyBytes)),
    RoleName:                  aws.String(roleName),
})
if err != nil {
    log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
} else {
    role = result.Role
}
return role, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[CreateRole](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;
```

```
/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that
represents the trust policy.\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }
}
```

```
public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
    try {
        JSONObject jsonObject = (JSONObject)
readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " +
response.role().arn());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ロールを作成します。

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});


/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- APIの詳細については、AWS SDK for JavaScriptAPI リファレンスの「[CreateRole](#)」を参照してください。

## PHP

## SDK for PHP

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"${user['Arn']}\"},
        \"Action\": \"sts:AssumeRole\"
    }]
}";

$assumeRoleRole = $service->createRole("iam_demo_role_{$uuid}",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

/**
 * @param string $roleName
 * @param string $rolePolicyDocument
 * @return array
 * @throws AwsException
 */
public function createRole(string $roleName, string $rolePolicyDocument)
{
    $result = $this->customWaiter(function () use ($roleName,
    $rolePolicyDocument) {
        return $this->iamClient->createRole([
            'AssumeRolePolicyDocument' => $rolePolicyDocument,
            'RoleName' => $roleName,
        ]);
    });
    return $result['Role'];
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[CreateRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MyNewRole** という名前の新しいロールを作成し、**NewRoleTrustPolicy.json** ファイルにあるポリシーをそのロールにアタッチします。JSON ポリシーファイルを正常に処理するには、**-Raw** switch パラメータを使用する必要がありますことに注意してください。出力に表示されるポリシードキュメントは、URL でエンコードされています。この例では、**UrlDecode** .NET メソッドを使用してデコードされています。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
  NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

出力:

```
Arn                : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                        %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                        %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                        %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                        %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                        %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                        %20%5D%0D%0A%7D
CreateDate         : 4/15/2015 11:04:23 AM
Path               : /
RoleId             : V5PAJI2KPN4EAEXAMPLE1
RoleName           : MyNewRole
```



```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateRole](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {"Service": service},
            "Action": "sts:AssumeRole",
        }
        for service in allowed_services
    ],
}

try:
    role = iam.create_role(
        RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
    )
    logger.info("Created role %s.", role.name)
except ClientError:
    logger.exception("Couldn't create role %s.", role_name)
    raise
else:
    return role
```

- APIの詳細については、「AWS SDK for Python (Boto3) API Reference」の「[CreateRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates a role and attaches policies to it.
#
# @param role_name [String] The name of the role.
# @param assume_role_policy_document [Hash] The trust relationship policy
document.
```

```
# @param policy_arns [Array<String>] The ARNs of the policies to attach.
# @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
def create_role(role_name, assume_role_policy_document, policy_arns)
  response = @iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: assume_role_policy_document.to_json
  )
  role_arn = response.role.arn

  policy_arns.each do |policy_arn|
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
  end

  role_arn
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating role: #{e.message}")
  nil
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_role(
  client: &iamClient,
  role_name: &str,
  role_policy_document: &str,
```

```
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[CreateRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func createRole(name: String, policyDocument: String) async throws ->
String {
    let input = CreateRoleInput(
        assumeRolePolicyDocument: policyDocument,
```

```
        roleName: name
    )
    do {
        let output = try await client.createRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        guard let id = role.roleId else {
            throw ServiceHandlerError.noSuchRole
        }
        return id
    } catch {
        throw error
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[CreateRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateSAMLProvider** を使用する

以下のコード例は、CreateSAMLProvider の使用方法を示しています。

### CLI

#### AWS CLI

SAML プロバイダーを作成するには

この例では、IAM に MySAMLProvider という名前の新しい SAML プロバイダーを作成します。これは、ファイル SAMLMetaData.xml 内の SAML メタデータドキュメントによって記述されます。

```
aws iam create-saml-provider \  
  --saml-metadata-document file://SAMLMetaData.xml \  
  --name MySAMLProvider
```

出力:

```
{
  "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM SAML ID プロバイダーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateSAMLProvider](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
import * as path from "path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 * see https://docs.aws.amazon.com/IAM/latest/UserGuide/
 * id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);
```

```
/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateSAMLProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM に新しい SAML プロバイダーエンティティを作成します。これは **MySAMLProvider** という名前で、SAML サービスプロバイダーのウェブサイトから個別にダウンロードされた **SAMLMetaData.xml** ファイルにある SAML メタデータドキュメントによって記述されます。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

出力:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateSAMLProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `CreateServiceLinkedRole` を使用する

以下のコード例は、`CreateServiceLinkedRole` の使用方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create an IAM service-linked role.
/// </summary>
/// <param name="serviceName">The name of the AWS Service.</param>
/// <param name="description">A description of the IAM service-linked role.</
param>
/// <returns>The IAM role that was created.</returns>
public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
string description)
{
    var request = new CreateServiceLinkedRoleRequest
    {
        AWSServiceName = serviceName,
        Description = description
    };

    var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
    return response.Role;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。



## CLI

## AWS CLI

サービスにリンクされたロールを作成するには

次の `create-service-linked-role` の例では、指定された AWS サービスのためにサービスにリンクされたロールを作成し、指定された説明をアタッチします。

```
aws iam create-service-linked-role \  
  --aws-service-name lex.amazonaws.com \  
  --description "My service-linked role to support Lex"
```

出力:


```
{  
  "Role": {  
    "Path": "/aws-service-role/lex.amazonaws.com/",  
    "RoleName": "AWSServiceRoleForLexBots",  
    "RoleId": "AROA1234567890EXAMPLE",  
    "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/  
AWSServiceRoleForLexBots",  
    "CreateDate": "2019-04-17T20:34:14+00:00",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Action": [  
            "sts:AssumeRole"  
          ],  
          "Effect": "Allow",  
          "Principal": {  
            "Service": [  
              "lex.amazonaws.com"  
            ]  
          }  
        }  
      ]  
    }  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[サービスにリンクされたロールの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    IamClient *iam.Client
}

// CreateServiceLinkedRole creates a service-linked role that is owned by the
// specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
description string) (*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
&iam.CreateServiceLinkedRoleInput{
    AWSServiceName: aws.String(serviceName),
    Description:    aws.String(description),
})
    if err != nil {
        log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
serviceName, err)
    } else {
        role = result.Role
    }
}
```

```
    return role, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サービスにリンクされたロールを作成します。

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
    // services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
    // general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
```

```
});
try {
  const response = await client.send(command);
  console.log(response);
  return response;
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidInputException" &&
    caught.message.includes(
      "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
this account",
    )
  ) {
    console.warn(caught.message);
    return client.send(
      new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
    );
  } else {
    throw caught;
  }
}
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();
```

```
public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
    $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
    $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
    >createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、自動スケーリングサービスのサービスにリンクされたロールを作成します。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
    RoleNameEndsWithThis -Description "My service-linked role to support
    autoscaling"
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
            service_name)
        raise
    else:
        return role
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates a service-linked role
#
# @param service_name [String] The service name to create the role for.
# @param description [String] The description of the service-linked role.
# @param suffix [String] Suffix for customizing role name.
# @return [String] The name of the created role
def create_service_linked_role(service_name, description, suffix)
  response = @iam_client.create_service_linked_role(
    aws_service_name: service_name, description: description, custom_suffix:
suffix,)
  role_name = response.role.role_name
  @logger.info("Created service-linked role #{role_name}.")
  role_name
rescue Aws::Errors::ServiceError => e
  @logger.error("Couldn't create service-linked role for #{service_name}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
    SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。



**Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func createServiceLinkedRole(service: String, suffix: String? = nil,
description: String?)
    async throws -> IAMClientTypes.Role {
    let input = CreateServiceLinkedRoleInput(
        awsServiceName: service,
        customSuffix: suffix,
        description: description
    )
    do {
        let output = try await client.createServiceLinkedRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        return role
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateUser** を使用する

以下のコード例は、CreateUser の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Create an IAM user.
/// </summary>
/// <param name="userName">The username for the new IAM user.</param>
/// <returns>The IAM user that was created.</returns>
public async Task<User> CreateUserAsync(string userName)
{
    var response = await _IAMService.CreateUserAsync(new CreateUserRequest
    { UserName = userName });
    return response.User;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[CreateUser](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#     -u user_name -- The name of the user to create.
#
# Returns:
#     The ARN of the user.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user() {
    local user_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user"
```

```
    echo "Creates an WS Identity and Access Management (IAM) user. You must
supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
account."
    echo ""
}

# Retrieve the calling parameters.
while getopts "u:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "  User name:  $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
  errecho "ERROR: A user with that name already exists in the account."
  return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
  --output text \
  --query 'User.Arn')
```

```
local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateUser](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
        create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}
```

```
return create_outcome.IsSuccess();
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[CreateUser](#)」を参照してください。

## CLI

### AWS CLI

例 1: IAM ユーザーを作成するには

次の `create-user` コマンドは、現在のアカウントに Bob という名前の IAM ユーザーを作成します。

```
aws iam create-user \  
  --user-name Bob
```

出力:

```
{  
  "User": {  
    "UserName": "Bob",  
    "Path": "/",  
    "CreateDate": "2023-06-08T03:20:41.270Z",  
    "UserId": "AIDAIOSFODNN7EXAMPLE",  
    "Arn": "arn:aws:iam::123456789012:user/Bob"  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。

例 2: 指定したパスに IAM ユーザーを作成するには

次の `create-user` コマンドは、指定されたパスに Bob という名前の IAM ユーザーを作成します。

```
aws iam create-user \  
  --user-name Bob \  
  --path /
```

```
--path /division_abc/subdivision_xyz/
```

出力:

```
{
  "User": {
    "Path": "/division_abc/subdivision_xyz/",
    "UserName": "Bob",
    "UserId": "AIDAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
    "CreateDate": "2023-05-24T18:20:17+00:00"
  }
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ID](#)」を参照してください。

例 3: タグを使用して IAM ユーザーを作成するには

次の `create-user` コマンドは、タグを使用して Bob という名前の IAM ユーザーを作成します。この例では、次の JSON 形式のタグを持つ `--tags` パラメータフラグを使用します: `'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`。あるいは、`--tags` フラグを次の短縮形式のタグとともに使用することもできます: `'Key=Department,Value=Accounting Key=Location,Value=Seattle'`。

```
aws iam create-user \
  --user-name Bob \
  --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
  "Value": "Seattle"}'
```

出力:

```
{
  "User": {
    "Path": "/",
    "UserName": "Bob",
    "UserId": "AIDAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::12345678012:user/Bob",
    "CreateDate": "2023-05-25T17:14:21+00:00",
    "Tags": [
      {
```

```
        "Key": "Department",
        "Value": "Accounting"
      },
      {
        "Key": "Location",
        "Value": "Seattle"
      }
    ]
  }
}
```

詳細については、「AWS IAM ユーザーガイド」で「[IAM ユーザーのタグ付け](#)」を参照してください。

例 3: アクセス許可の境界が設定された IAM ユーザーを作成するには

次の `create-user` コマンドは、AmazonS3FullAccess のアクセス許可の境界を持つ Bob という名前の IAM ユーザーを作成します。

```
aws iam create-user \
  --user-name Bob \
  --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

出力:

```
{
  "User": {
    "Path": "/",
    "UserName": "Bob",
    "UserId": "AIDAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::12345678012:user/Bob",
    "CreateDate": "2023-05-24T17:50:53+00:00",
    "PermissionsBoundary": {
      "PermissionsBoundaryType": "Policy",
      "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
    }
  }
}
```


詳細については、「AWS IAM ユーザーガイド」の「[IAM エンティティのアクセス許可境界](#)」を参照してください。



- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateUser](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.CreateUser(context.TODO(),
        &iam.CreateUserInput{
            UserName: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    } else {
        user = result.User
    }
    return user, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[CreateUser](#)」を参照してください。

## Java

## SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username>\s

            Where:
                username - The name of the user to create.\s

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String username = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMUser(iam, username);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateUser](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ユーザーを作成します。


```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateUser](#)」を参照してください。

## SDK for JavaScript (v2)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  Username: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateUser](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateUser](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();
```

```
$user = $service->createUser("iam_demo_user_${uuid}");
echo "Created user with the arn: {$user['Arn']}\n";

/**
 * @param string $name
 * @return array
 * @throws AwsException
 */
public function createUser(string $name): array
{
    $result = $this->iamClient->createUser([
        'UserName' => $name,
    ]);

    return $result['User'];
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[CreateUser](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーを作成します。Bob が AWS コンソールにサインインする必要がある場合は、**New-IAMLoginProfile** コマンドを個別に実行して、パスワード付きのサインインプロファイルを作成する必要があります。Bob が AWS PowerShell またはクロスプラットフォーム CLI コマンドを実行したり、AWS API 呼び出しを行ったりする必要がある場合は、**New-IAMAccessKey** コマンドを個別に実行して、アクセスキーを作成する必要があります。

```
New-IAMUser -UserName Bob
```

出力:

```
Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path         : /
```

```
UserId      : AIDAJWGEFDMEMEXAMPLE1
UserName    : Bob
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateUser](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(Username=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateUser](#)」を参照してください。



## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates a user and their login profile
#
# @param user_name [String] The name of the user
# @param initial_password [String] The initial password for the user
# @return [String, nil] The ID of the user if created, or nil if an error
occurred
def create_user(user_name, initial_password)
  response = @iam_client.create_user(user_name: user_name)
  @iam_client.wait_until(:user_exists, user_name: user_name)
  @iam_client.create_login_profile(
    user_name: user_name,
    password: initial_password,
    password_reset_required: true
  )
  @logger.info("User '#{user_name}' created successfully.")
  response.user.user_id
rescue Aws::IAM::Errors::EntityAlreadyExists
  @logger.error("Error creating user '#{user_name}': user already exists.")
  nil
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating user '#{user_name}': #{e.message}")
  nil
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateUser](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[CreateUser](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func createUser(name: String) async throws -> String {
    let input = CreateUserInput(
```

```
        userName: name
    )
    do {
        let output = try await client.createUser(input: input)
        guard let user = output.user else {
            throw ServiceHandlerError.noSuchUser
        }
        guard let id = user.userId else {
            throw ServiceHandlerError.noSuchUser
        }
        return id
    } catch {
        throw error
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[CreateUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **CreateVirtualMfaDevice** を使用する

以下のコード例は、CreateVirtualMfaDevice の使用方法を示しています。

### CLI

#### AWS CLI

仮想 MFA デバイスを作成するには

この例では、BobsMFADevice という新しい仮想 MFA デバイスを作成します。QRCode.png というブートストラップ情報を含むファイルを作成し、C:/ ディレクトリに配置します。この例で使用されているブートストラップメソッドは QRCodePNG です。

```
aws iam create-virtual-mfa-device \  
  --virtual-mfa-device-name BobsMFADevice \  
  --outfile C:/QRCode.png \  
  --bootstrap-method QRCodePNG
```

出力:

```
{
  "VirtualMFADevice": {
    "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
  }
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVirtualMfaDevice](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、新しい仮想 MFA デバイスを作成します。2 行目と 3 行目は、仮想 MFA ソフトウェアプログラムが (QR コードの代わりに) アカウントを作成するのに必要な **Base32StringSeed** 値を抽出します。この値でプログラムを設定したら、プログラムから 2 つの連続した認証コードを取得します。最後に、最後のコマンドを使用して仮想 MFA デバイスを IAM ユーザー **Bob** にリンクし、アカウントを 2 つの認証コードと同期します。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

出力:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

例 2: この例では、新しい仮想 MFA デバイスを作成します。2 行目と 3 行目は、**QRCodePNG** 値を抽出してファイルに書き込みます。(Base32StringSeed 値を手動で入力する代わりに) このイメージを仮想 MFA ソフトウェアプログラムでスキャンして、アカウントを作成できま

す。仮想 MFA プログラムでアカウントを作成したら、2 つの連続した認証コードを取得して最後のコマンドに入力し、仮想 MFA デバイスを IAM ユーザー **Bob** にリンクして、アカウントを同期します。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
QRCode.png
```

出力:

```
-- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[CreateVirtualMfaDevice](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeactivateMfaDevice** を使用する

以下のコード例は、DeactivateMfaDevice の使用方法を示しています。

CLI

### AWS CLI

デバイスを無効にするには

このコマンドは、ユーザー Bob に関連付けられている ARN `arn:aws:iam::210987654321:mfa/BobsMFADevice` を持つ仮想 MFA デバイスを無効にします。

```
aws iam deactivate-mfa-device \
  --user-name Bob \
  --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeactivateMfaDevice](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、シリアル番号 **123456789012** を持つユーザー **Bob** に関連付けられているハードウェア MFA デバイスを無効にします。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

例 2: このコマンドは、ARN **arn:aws:iam::210987654321:mfa/David** を持つユーザー **David** に関連付けられている仮想 MFA デバイスを無効にします。仮想 MFA デバイスはアカウントから削除されないことに注意してください。仮想デバイスはまだ存在し、**Get-IAMVirtualMFADevice** コマンドの出力に表示されます。同じユーザーに対して新しい仮想 MFA デバイスを作成する前に、**Remove-IAMVirtualMFADevice** コマンドを使用して、古い仮想 MFA デバイスを削除する必要があります。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber  
"arn:aws:iam::210987654321:mfa/David"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeactivateMfaDevice](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteAccessKey** を使用する

以下のコード例は、DeleteAccessKey の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)
- [アクセスキーの管理](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM user's access key.
/// </summary>
/// <param name="accessKeyId">The Id for the IAM access key.</param>
/// <param name="userName">The username of the user that owns the IAM
/// access key.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
{
    var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
    {
        AccessKeyId = accessKeyId,
        UserName = userName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#     -u user_name -- The name of the user.
#     -k access_key -- The access key to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_access_key() {
    local user_name access_key response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_access_key"
        echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
        echo "  -u user_name    The name of the user."
    }
}
```



```
    echo " -k access_key  The access key to delete."
    echo ""
}

# Retrieve the calling parameters.
while getopts "u:k:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  Username:  $user_name"
iecho "  Access key:  $access_key"
iecho ""

response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

local error_code=${?}
```

```
if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
fi

iecho "delete-access-key response:$response"
iecho

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteAccessKey](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                   const Aws::String &accessKeyID,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
```

```
        << userName << " : " << outcome.GetError().GetMessage() <<
        std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
        << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーのためにアクセスキーを削除するには

次の `delete-access-key` コマンドは、Bob という名前の IAM ユーザーのために指定されたアクセスキー (アクセスキー ID とシークレットアクセスキー) を削除します。

```
aws iam delete-access-key \
  --access-key-id AKIDPMS9R04H3FEXAMPLE \
  --user-name Bob
```

このコマンドでは何も出力されません。


IAM ユーザーのために定義されたアクセスキーを一覧表示するには、`list-access-keys` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
    _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
        &iam.DeleteAccessKeyInput{
            AccessKeyId: aws.String(keyId),
            Username:    aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
    }
    return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Java

## SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username> <accessKey>\s

                Where:
                username - The name of the user.\s
                accessKey - The access key ID for the secret access key you
                want to delete.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String username = args[0];
String accessKey = args[1];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();
deleteKey(iam, username, accessKey);
iam.close();
}

public static void deleteKey(IamClient iam, String username, String
accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

## アクセスキーを削除します

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
```

```
    AccessKeyId: "ACCESS_KEY_ID",
    UserName: "USER_NAME",
  };

  iam.deleteAccessKey(params, function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  });
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```



```
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前のユーザーからキー ID **AKIAIOSFODNN7EXAMPLE** の AWS アクセスキーペアを削除します。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
```

```
except ClientError:
    logger.exception("Couldn't delete key %s for %s", key_id, user_name)
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」で「[DeleteAccessKey](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  end
end
```

```
rescue Aws::IAM::Errors::NoSuchEntity => e
  @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
  []
rescue StandardError => e
  @logger.error("Error listing access keys: #{e.message}")
  []
end

# Creates an access key for a user
#
# @param user_name [String] The name of the user.
# @return [Boolean]
def create_access_key(user_name)
  response = @iam_client.create_access_key(user_name: user_name)
  access_key = response.access_key
  @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
  access_key
rescue Aws::IAM::Errors::LimitExceeded => e
  @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
  nil
rescue StandardError => e
  @logger.error("Error creating access key: #{e.message}")
  nil
end

# Deactivates an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def deactivate_access_key(user_name, access_key_id)
  @iam_client.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,
    status: "Inactive"
  )
  true
rescue StandardError => e
  @logger.error("Error deactivating access key: #{e.message}")
  false
end
```

```
# Deletes an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def delete_access_key(user_name, access_key_id)
  @iam_client.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  true
rescue StandardError => e
  @logger.error("Error deleting access key: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_access_key(
  client: &iamClient,
  user: &User,
  key: &AccessKey,
) -> Result<(), iamError> {
  loop {
    match client
      .delete_access_key()
      .user_name(user.user_name())
      .access_key_id(key.access_key_id())
```

```
        .send()
        .await
    {
        Ok(_) => {
            break;
        }
        Err(e) => {
            println!("Can't delete the access key: {:?}", e);
            sleep(Duration::from_secs(2)).await;
        }
    }
}
Ok(())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                             key: IAMClientTypes.AccessKey) async throws {
    let userName: String?

    if user != nil {
```

```
        userName = user!.userName
    } else {
        userName = nil
    }

    let input = DeleteAccessKeyInput(
        accessKeyId: key.accessKeyId,
        userName: userName
    )
    do {
        _ = try await iamClient.deleteAccessKey(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteAccountAlias** を使用する

以下のコード例は、DeleteAccountAlias の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

## C++

## SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## CLI

## AWS CLI

アカウントエイリアスを削除するには

次の `delete-account-alias` コマンドは、現在のアカウントのエイリアス `mycompany` を削除します。

```
aws iam delete-account-alias \  
  --account-alias mycompany
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウント ID とそのエイリアス](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteAccountAlias {  
    public static void main(String[] args) {  
        final String usage = ""
```



```
Usage:
    <alias>\s

Where:
    alias - The account alias to delete.\s
""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String alias = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

deleteIAMAccountAlias(iam, alias);
iam.close();
}

public static void deleteIAMAccountAlias(IamClient iam, String alias) {
    try {
        DeleteAccountAliasRequest request =
DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを削除します。

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

```
}  
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウントからアカウントエイリアスを削除します。https://mycompanyaws.signin.aws.amazon.com/console で、エイリアスを持つユーザーサインインページは機能しなくなりました。代わりに、https://<accountidnumber>.signin.aws.amazon.com/console で、AWS アカウント ID 番号を持つ元の URL を使用する必要があります。

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteAccountAlias](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def delete_alias(alias):  
    """  
    Removes the alias from the current account.  
  
    :param alias: The alias to remove.  
    """  
    try:
```

```
iam.meta.client.delete_account_alias(AccountAlias=alias)
logger.info("Removed alias '%s' from your account.", alias)
except ClientError:
    logger.exception("Couldn't remove alias '%s' from your account.", alias)
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」で「[DeleteAccountAlias](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
#{account_alias}") }
    else
```

```
    @logger.info("No account aliases found.")
  end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing account aliases: #{e.message}")
end

# Creates an AWS account alias.
#
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
def create_account_alias(account_alias)
  @iam_client.create_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating account alias: #{e.message}")
  false
end

# Deletes an AWS account alias.
#
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
def delete_account_alias(account_alias)
  @iam_client.delete_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteAccountPasswordPolicy** を使用する

以下のコード例は、DeleteAccountPasswordPolicy の使用方法を示しています。

## CLI

### AWS CLI

現在のアカウントのパスワードポリシーを削除するには

次の `delete-account-password-policy` コマンドは、現在のアカウントのパスワードポリシーを削除します。

```
aws iam delete-account-password-policy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアカウントパスワードポリシーの設定](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteAccountPasswordPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウントのパスワードポリシーを削除し、すべての値を元のデフォルトにリセットします。パスワードポリシーが現在存在しない場合、次のエラーメッセージが表示されます。The account policy with name PasswordPolicy cannot be found.

```
Remove-IAMAccountPasswordPolicy
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteAccountPasswordPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteGroup** を使用する

以下のコード例は、DeleteGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteGroupAsync(string groupName)
{
    var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteGroup](#)」を参照してください。

## CLI

### AWS CLI

IAM グループを削除するには

次の delete-group コマンドは、MyTestGroup という名前の IAM グループを削除します。



```
aws iam delete-group \  
  --group-name MyTestGroup
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの削除](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteGroup](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";  
  
const client = new IAMClient({});  
  
/**  
 *  
 * @param {string} groupName  
 */  
export const deleteGroup = async (groupName) => {  
  const command = new DeleteGroupCommand({  
    GroupName: groupName,  
  });  
  
  const response = await client.send(command);  
  console.log(response);  
  return response;  
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MyTestGroup** という名前の IAM グループを削除します。最初のコマンドはグループのメンバーであるすべての IAM ユーザーを削除し、2 番目のコマンドは IAM グループを削除します。どちらのコマンドも、確認を求めるプロンプトが表示されなくても機能します。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteGroupPolicy** を使用する

以下のコード例は、DeleteGroupPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM policy associated with an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group associated with the
/// policy.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
{
    var request = new DeleteGroupPolicyRequest()
    {
        GroupName = groupName,
        PolicyName = policyName,
    };

    var response = await _IAMService.DeleteGroupPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteGroupPolicy](#)」を参照してください。

## CLI

### AWS CLI

IAM グループからポリシーを削除するには

次の `delete-group-policy` コマンドは、Admins という名前のグループから ExamplePolicy という名前のポリシーを削除します。

```
aws iam delete-group-policy \  
  --group-name Admins \  
  --policy-name ExamplePolicy
```

このコマンドでは何も出力されません。

グループにアタッチされているポリシーを表示するには、`list-group-policies` コマンドを使用します。

IAM ポリシーの詳細については、「AWS IAM ユーザーガイド」の「[IAM ポリシーを管理する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteGroupPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**TesterPolicy** という名前のインラインポリシーを IAM グループ **Testers** から削除します。そのグループのユーザーは、そのポリシーで定義されているアクセス権限をすぐに失います。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteGroupPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteInstanceProfile** を使用する


以下のコード例は、DeleteInstanceProfile の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)

.NET

AWS SDK for .NET

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Detaches a role from an instance profile, detaches policies from the
role,
/// and deletes all the resources.
/// </summary>
/// <param name="profileName">The name of the profile to delete.</param>
/// <param name="roleName">The name of the role to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteInstanceProfile(string profileName, string roleName)
{
    try
    {
        await _amazonIam.RemoveRoleFromInstanceProfileAsync(
            new RemoveRoleFromInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
        await _amazonIam.DeleteInstanceProfileAsync(
            new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
        var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
            new ListAttachedRolePoliciesRequest() { RoleName = roleName });
        foreach (var policy in attachedPolicies.AttachedPolicies)
        {
```

```
        await _amazonIam.DetachRolePolicyAsync(
            new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policy.PolicyArn
            });
        // Delete the custom policies only.
        if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
        {
            await _amazonIam.DeletePolicyAsync(
                new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                {
                    PolicyArn = policy.PolicyArn
                });
        }
    }

    await _amazonIam.DeleteRoleAsync(
        new DeleteRoleRequest() { RoleName = roleName });
}
catch (NoSuchEntityException)
{
    Console.WriteLine($"Instance profile {profileName} does not exist.");
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteInstanceProfile](#)」を参照してください。

## CLI

### AWS CLI

インスタンスプロファイルを削除するには

次の `delete-instance-profile` コマンドは、`ExampleInstanceProfile` という名前のインスタンスプロファイルを削除します。

```
aws iam delete-instance-profile \  
    --instance-profile-name ExampleInstanceProfile
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteInstanceProfile](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
const client = new IAMClient({});
await client.send(
  new DeleteInstanceProfileCommand({
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteInstanceProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MyAppInstanceProfile** という名前の EC2 インスタンスプロファイルを削除します。最初のコマンドはインスタンスプロファイルからすべてのロールをデタッチし、2 番目のコマンドはインスタンスプロファイルを削除します。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

```
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteInstanceProfile](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

この例では、インスタンスプロファイルからロールを削除し、ロールにアタッチされているすべてのポリシーをデタッチし、すべてのリソースを削除します。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
                           created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
```



```
:param ec2_client: A Boto3 EC2 client.
:param ssm_client: A Boto3 Systems Manager client.
:param iam_client: A Boto3 IAM client.
"""

self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
self.launch_template_name = f"{resource_prefix}-template"
self.group_name = f"{resource_prefix}-group"
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

def delete_instance_profile(self, profile_name, role_name):
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
"""
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
        log.info("Deleted instance profile %s.", profile_name)
        attached_policies = self.iam_client.list_attached_role_policies(
            RoleName=role_name
        )
        for pol in attached_policies["AttachedPolicies"]:
            self.iam_client.detach_role_policy(
                RoleName=role_name, PolicyArn=pol["PolicyArn"]
            )
```

```
        if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
            self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
            log.info("Detached and deleted policy %s.", pol["PolicyName"])
        self.iam_client.delete_role(RoleName=role_name)
        log.info("Deleted role %s.", role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
                profile_name
            )
        else:
            raise AutoScalerError(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[DeleteInstanceProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteLoginProfile** を使用する

以下のコード例は、DeleteLoginProfile の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーのパスワードを削除するには

次の delete-login-profile コマンドは、Bob という名前の IAM ユーザーのパスワードを削除します。

```
aws iam delete-login-profile \  
    --user-name Bob
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのパスワードの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteLoginProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーからログインプロファイルを削除します。これにより、ユーザーは AWS コンソールにサインインできなくなります。ユーザーアカウントにまだアタッチされている可能性のある AWS アクセスキーを使用して、AWS CLI、PowerShell、または API 呼び出しを実行できなくなることはありません。

```
Remove-IAMLoginProfile -UserName Bob
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteLoginProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `DeleteOpenIdConnectProvider` を使用する

以下のコード例は、`DeleteOpenIdConnectProvider` の使用方法を示しています。

### CLI

#### AWS CLI

IAM OpenID Connect ID プロバイダーを削除するには

この例では、プロバイダー `example.oidcprovider.com` に接続する IAM OIDC プロバイダーを削除します。

```
aws iam delete-open-id-connect-provider \
```

```
--open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteOpenIdConnectProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、プロバイダー **example.oidcprovider.com** に接続する IAM OIDC プロバイダーを削除します。ロールの信頼ポリシーの **Principal** 要素で、このプロバイダーを参照するロールをすべて更新または削除してください。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteOpenIdConnectProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeletePolicy** を使用する

以下のコード例は、DeletePolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)
- [ポリシーを管理](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM policy.
/// </summary>
/// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
/// delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeletePolicyAsync(string policyArn)
{
    var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeletePolicy](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
```

```

#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#     -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_policy() {
    local policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_policy"
        echo "Deletes an WS Identity and Access Management (IAM) policy"
        echo "  -n policy_arn -- The name of the IAM policy arn."
        echo ""
    }

    # Retrieve the calling parameters.

```

```
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "  Policy arn: $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
  return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeletePolicy](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration
                               &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DeletePolicy](#)」を参照してください。



## CLI

### AWS CLI

IAM ポリシーを削除するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MySamplePolicy` であるポリシーを削除します。

```
aws iam delete-policy \  
  --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeletePolicy](#)」を参照してください。

## Go

### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy  
actions  
// used in the examples.  
// It contains an IAM service client that is used to perform policy actions.  
type PolicyWrapper struct {  
  IamClient *iam.Client  
}  
  
// DeletePolicy deletes a policy.
```

```
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
    _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
    }
    return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[DeletePolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <policyARN>\s

    Where:
        policyARN - A policy ARN value to delete.\s
""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String policyARN = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

deleteIAMPolicy(iam, policyARN);
iam.close();
}

public static void deleteIAMPolicy(IamClient iam, String policyARN) {
    try {
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(policyARN)
            .build();

        iam.deletePolicy(request);
        System.out.println("Successfully deleted the policy");

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeletePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーを削除します。

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeletePolicy](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeletePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MySamplePolicy** であるポリシーを削除します。ポリシーを削除する前に、**Remove-IAMPolicyVersion** を実行して、デフォルト以外のすべてのバージョンを削除する必要があります。また、すべての IAM ユーザー、グループ、またはロールからポリシーをデタッチする必要があります。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

例 2: この例では、最初にデフォルト以外のポリシーバージョンをすべて削除し、アタッチされているすべての IAM エンティティからデタッチして、最後にポリシー自体を削除すること

でポリシーを削除します。1 行目では、ポリシーオブジェクトを取得します。2 行目では、デフォルトバージョンとしてフラグが立てられていないすべてのポリシーバージョンをコレクションに取得し、コレクション内の各ポリシーを削除します。3 行目では、ポリシーがアタッチされているすべての IAM ユーザー、グループ、およびロールを取得します。4 行目から 6 行目では、アタッチされている各エンティティからポリシーをデタッチします。最後の行では、このコマンドを使用して管理ポリシーと残りのデフォルトバージョンを削除します。この例には、確認を求めるプロンプトを非表示にするための **-Force** switch パラメータが、このパラメータを必要とする行に含まれています。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
  Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeletePolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
```

```
logger.info("Deleted policy %s.", policy_arn)
except ClientError:
    logger.exception("Couldn't delete policy %s.", policy_arn)
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeletePolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[DeletePolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
    let input = DeletePolicyInput(
        policyArn: policy.arn
    )
    do {
        _ = try await iamClient.deletePolicy(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeletePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **DeletePolicyVersion** を使用する

以下のコード例は、DeletePolicyVersion の使用方法を示しています。



アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)
- [ポリシーを以前のバージョンにロールバックする](#)

## CLI

### AWS CLI

管理ポリシーのバージョンを削除するには

この例では、v2 と識別されたバージョンを ARN が `arn:aws:iam::123456789012:policy/MySamplePolicy` であるポリシーから削除します。

```
aws iam delete-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --version-id v2
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeletePolicyVersion](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、v2 と識別されたバージョンを ARN が `arn:aws:iam::123456789012:policy/MySamplePolicy` であるポリシーから削除します。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/  
MySamplePolicy -VersionID v2
```

例 2: この例では、最初にデフォルト以外のポリシーバージョンをすべて削除し、次にポリシー自体を削除することでポリシーを削除します。1 行目では、ポリシーオブジェクトを取得します。2 行目では、デフォルトとしてフラグが立てられていないすべてのポリシーバージョンをコレクションに取得し、このコマンドを使用してコレクション内の各ポリシーを削除します。最後の行では、ポリシー自体と残りのデフォルトバージョンを削除します。管理ポリシーを正常に削除するには、**Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy**、**Unregister-IAMRolePolicy** コマンドを使用して、ユーザー、グループ、またはロールからポリシーをデタッチする必要があることに注意してください。**Remove-IAMPolicy** コマンドレットの例を参照してください。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
  Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeletePolicyVersion](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteRole** を使用する

以下のコード例は、DeleteRole の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ユーザーを作成してロールを引き受ける](#)
- [ロールの管理](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRoleAsync(string roleName)
{
    var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteRole](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
#
```

```
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_role() {
    local role_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_role"
        echo "Deletes an WS Identity and Access Management (IAM) role"
        echo "  -n role_name -- The name of the IAM role."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
```

```
case "${option}" in
  n) role_name="${OPTARG}" ;;
  h)
    usage
    return 0
    ;;
  \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

echo "role_name:$role_name"
if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "  Role name:  $role_name"
iecho ""

response=$(aws iam delete-role \
  --role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-role operation failed.\n$response"
  return 1
fi

iecho "delete-role response:$response"
iecho

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRole](#)」を参照してください。

## CLI

### AWS CLI

IAM ロールを削除するには

次の delete-role コマンドは、Test-Role という名前のロールを削除します。

```
aws iam delete-role \  
  --role-name Test-Role
```

このコマンドでは何も出力されません。

ロールを削除する前に、インスタンスプロファイルからロールを削除し (remove-role-from-instance-profile)、管理ポリシーをデタッチして (detach-role-policy)、ロールにアタッチされているインラインポリシーを削除する (delete-role-policy) 必要があります。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」および「[インスタンスプロファイルの使用](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRole](#)」を参照してください。

## Go

### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
```

```
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
    }
    return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[DeleteRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

ロールを削除します。

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の IAM アカウントから **MyNewRole** という名前のロールを削除します。ロールを削除する前に、まず **Unregister-IAMRolePolicy** コマンドを使用して、管理ポリシーをデタッチする必要があります。インラインポリシーは、ロールと共に削除されません。

```
Remove-IAMRole -RoleName MyNewRole
```

例 2: この例では、**MyNewRole** という名前のロールから管理ポリシーをすべてデタッチして、ロールを削除します。最初の行では、ロールにアタッチされているすべての管理ポリシーをコレクションとして取得し、コレクション内の各ポリシーをロールからデタッチします。2 行目では、ロール自体を削除します。インラインポリシーは、ロールと共に削除されます。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteRole](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」で「[DeleteRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Deletes a role and its attached policies.
```

```
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  begin
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
    response.attached_policies.each do |policy|
      @iam_client.detach_role_policy({
        role_name: role_name,
        policy_arn: policy.policy_arn
      })

      # Check if the policy is a customer managed policy (not AWS managed)
      unless policy.policy_arn.include?("aws:policy/")
        @iam_client.delete_policy({ policy_arn: policy.policy_arn })
        @logger.info("Deleted customer managed policy
#{policy.policy_name}.")
      end
    end
  end

  # Delete the role
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Deleted role #{role_name}.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't detach policies and delete role #{role_name}.
Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
{
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

**Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func deleteRole(role: IAMClientTypes.Role) async throws {
    let input = DeleteRoleInput(
        roleName: role.roleName
    )
    do {
        _ = try await iamClient.deleteRole(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteRolePermissionsBoundary** を使用する

以下のコード例は、DeleteRolePermissionsBoundary の使用方法を示しています。

### CLI

#### AWS CLI

IAM ロールからアクセス許可の境界を削除するには

次の delete-role-permissions-boundary の例では、指定した IAM ロールのアクセス許可の境界を削除します。アクセス許可の境界をロールに適用するには、put-role-permissions-boundary コマンドを使用します。

```
aws iam delete-role-permissions-boundary \
```

```
--role-name lambda-application-role
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRolePermissionsBoundary](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例は、IAM ロールにアタッチされたアクセス許可の境界を削除する方法を示しています。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteRolePermissionsBoundary](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteRolePolicy** を使用する

以下のコード例は、DeleteRolePolicy の使用方法を示しています。

### .NET

#### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM role policy.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="policyName">The name of the IAM role policy to delete.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
{
    var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteRolePolicy](#)」を参照してください。

## CLI

### AWS CLI

IAM ロールからポリシーを削除するには

次の delete-role-policy コマンドは、Test-Role という名前のロールから ExamplePolicy という名前のポリシーを削除します。

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRolePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ロール **S3BackupRole** に埋め込まれているインラインポリシー **S3AccessPolicy** を削除します。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteSAMLProvider** を使用する

以下のコード例は、DeleteSAMLProvider の使用方法を示しています。

### CLI

#### AWS CLI

SAML プロバイダーを削除するには

この例では、ARN が `arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER` である IAM SAML 2.0 プロバイダーを削除します。

```
aws iam delete-saml-provider \  
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM SAML ID プロバイダーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSAMLProvider](#)」を参照してください。



## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteSAMLProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER** である IAM SAML 2.0 プロバイダーを削除します。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/  
SAMLADFSProvider
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteSAMLProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `DeleteServerCertificate` を使用する

以下のコード例は、`DeleteServerCertificate` の使用方法を示しています。

C++

SDK for C++

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,  
                                          const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::IAM::IAMClient iam(clientConfig);  
    Aws::IAM::Model::DeleteServerCertificateRequest request;  
    request.SetServerCertificateName(certificateName);  
  
    const auto outcome = iam.DeleteServerCertificate(request);  
    bool result = true;  
    if (!outcome.IsSuccess()) {  
        if (outcome.GetError().GetErrorType() !=  
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {  
            std::cerr << "Error deleting server certificate " << certificateName  
<<  
                ": " << outcome.GetError().GetMessage() << std::endl;  
            result = false;  
        }  
    }  
}
```

```
    }
    else {
        std::cout << "Certificate '" << certificateName
            << "' not found." << std::endl;
    }
}
else {
    std::cout << "Successfully deleted server certificate " <<
certificateName
        << std::endl;
}

return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DeleteServerCertificate](#)」を参照してください。

## CLI

### AWS CLI

AWS アカウントからサーバー証明書を削除するには

次の `delete-server-certificate` コマンドは、指定されたサーバー証明書を AWS アカウントから削除します。

```
aws iam delete-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

このコマンドでは何も出力されません。

AWS アカウントで使用可能なサーバー証明書を一覧表示するには、`list-server-certificates` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM でのサーバー証明書の管理](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteServerCertificate](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を削除します。

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteServerCertificate](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteServerCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell


例 1: この例では、**MyServerCert** という名前のサーバー証明書を削除します。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteServerCertificate](#)」を参照してください。

## Ruby

## SDK for Ruby

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end
  end
end
```

```
end

response.server_certificate_metadata_list.each do |certificate_metadata|
  @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing server certificates: #{e.message}")
end

# Updates the name of a server certificate.
def update_server_certificate_name(current_name, new_name)
  @iam_client.update_server_certificate(
    server_certificate_name: current_name,
    new_server_certificate_name: new_name
  )
  @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error updating server certificate name: #{e.message}")
  false
end

# Deletes a server certificate.
def delete_server_certificate(name)
  @iam_client.delete_server_certificate(server_certificate_name: name)
  @logger.info("Server certificate '#{name}' deleted.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting server certificate: #{e.message}")
  false
end
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteServerCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `DeleteServiceLinkedRole` を使用する

以下のコード例は、`DeleteServiceLinkedRole` の使用方法を示しています。

### CLI

#### AWS CLI

サービスにリンクされたロールを削除するには

次の `delete-service-linked-role` の例では、不要になったサービスにリンクされたロールのうち、指定されたものを削除します。削除は非同期で実行されます。`get-service-linked-role-deletion-status` コマンドを使用して、削除のステータスをチェックし、削除がいつ完了したかを確認できます。

```
aws iam delete-service-linked-role \  
  --role-name AWSServiceRoleForLexBots
```

出力:

```
{  
  "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/  
  AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[サービスにリンクされたロールの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

### Go

#### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。



```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
        &iam.DeleteServiceLinkedRoleInput{
            RoleName: aws.String(roleName)},
    )
    if err != nil {
        log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
            roleName, err)
    }
    return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";
const client = new IAMClient({});
```

```
/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、サービスにリンクされたロールを削除しました。サービスがこのロールをまだ使用している場合、このコマンドは失敗することに注意してください。

```
Remove-IAMServiceLinkedRole -RoleName
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Deletes a service-linked role.
#
```

```
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  unless e.code == "NoSuchEntity"
    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **DeleteSigningCertificate** を使用する

以下のコード例は、DeleteSigningCertificate の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ユーザーの署名証明書を削除するには

次の `delete-signing-certificate` コマンドは、Bob という名前の IAM ユーザーの指定された署名証明書を削除します。

```
aws iam delete-signing-certificate \  
  --user-name Bob \  
  --certificate-id TA7SMP42TDN5Z260BPJE7EXAMPLE
```

このコマンドでは何も出力されません。

署名証明書の ID を取得するには、`list-signing-certificates` コマンドを使用します。

詳細については、「Amazon EC2 ユーザーガイド」の「[署名証明書の管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSigningCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーから ID **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** の付いた署名証明書を削除します。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteSigningCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteUser** を使用する

以下のコード例は、`DeleteUser` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)
- [ユーザーを作成してロールを引き受ける](#)
- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an IAM user.
/// </summary>
/// <param name="userName">The username of the IAM user to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserAsync(string userName)
{
    var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
    { UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteUser](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user to create.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
```

```
#####  
function iam_delete_user() {  
    local user_name response  
    local option OPTARG # Required to use getopt command in a function.  
  
    # bashsupport disable=BP5008  
    function usage() {  
        echo "function iam_delete_user"  
        echo "Deletes an WS Identity and Access Management (IAM) user. You must  
supply a username:"  
        echo "  -u user_name    The name of the user."  
        echo ""  
    }  
  
    # Retrieve the calling parameters.  
    while getopt "u:h" option; do  
        case "${option}" in  
            u) user_name="${OPTARG}" ;;  
            h)  
                usage  
                return 0  
                ;;  
            \?)  
                echo "Invalid parameter"  
                usage  
                return 1  
                ;;  
        esac  
    done  
    export OPTIND=1  
  
    if [[ -z "$user_name" ]]; then  
        errecho "ERROR: You must provide a username with the -u parameter."  
        usage  
        return 1  
    fi  
  
    iecho "Parameters:\n"  
    iecho "  User name:  $user_name"  
    iecho ""  
  
    # If the user does not exist, we don't want to try to delete it.  
    if (! iam_user_exists "$user_name"); then  
        errecho "ERROR: A user with that name does not exist in the account."  
    fi  
}
```



```
    return 1
fi

response=$(aws iam delete-user \
  --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-user operation failed.$response"
  return 1
fi

iecho "delete-user response:$response"
iecho

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteUser](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ": " <<
```

```
        outcome.GetError().GetMessage() << std::endl;;
    }
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DeleteUser](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーを削除するには

次の delete-user コマンドは、現在のアカウントから Bob という名前の IAM ユーザーを削除します。

```
aws iam delete-user \  
  --user-name Bob
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーの削除](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteUser](#)」を参照してください。

## Go

### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
    _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
    }
    return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[DeleteUser](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <userName>\s

            Where:
                userName - The name of the user to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);
        }
    }
}
```

```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteUser](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ユーザーを削除。


```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
    const command = new DeleteUserCommand({ UserName: name });
    return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteUser](#)」を参照してください。

## SDK for JavaScript (v2)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  Username: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteUser](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeleteUser](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーを削除します。

```
Remove-IAMUser -UserName Bob
```

例 2: この例では、**Theresa** という名前の IAM ユーザーと、最初に削除する必要がある要素をすべて削除します。

```
$name = "Theresa"
```

```
# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
  -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
  $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
  $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
  $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
  UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
  suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
  Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
  if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
  SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteUser](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」で「[DeleteUser](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Deletes a user and their associated resources
#
# @param user_name [String] The name of the user to delete
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
  user.each do |key|
    @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end

  @iam_client.delete_user(user_name: user_name)
  @logger.info("Deleted user '#{user_name}'.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting user '#{user_name}': #{e.message}")
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteUser](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
SdkError<DeleteUserError>> {
  let user = user.clone();
  let mut tries: i32 = 0;
  let max_tries: i32 = 10;

  let response: Result<(), SdkError<DeleteUserError>> = loop {
    match client
      .delete_user()
```

```
        .user_name(user.user_name())
        .send()
        .await
    {
        Ok(_) => {
            break Ok(());
        }
        Err(e) => {
            tries += 1;
            if tries > max_tries {
                break Err(e);
            }
            sleep(Duration::from_secs(2)).await;
        }
    }
};

response
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteUser](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func deleteUser(user: IAMClientTypes.User) async throws {
    let input = DeleteUserInput(
        userName: user.userName
    )
    do {
        _ = try await iamClient.deleteUser(input: input)
    } catch {
        throw error
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteUserPermissionsBoundary** を使用する

以下のコード例は、DeleteUserPermissionsBoundary の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーからアクセス許可の境界を削除するには

次の delete-user-permissions-boundary の例では、intern という名前の IAM ユーザーにアタッチされているアクセス許可の境界を削除します。アクセス許可の境界をユーザーに適用するには、put-user-permissions-boundary コマンドを使用します。

```
aws iam delete-user-permissions-boundary \  
    --user-name intern
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteUserPermissionsBoundary](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例は、IAM ユーザーにアタッチされたアクセス許可の境界を削除する方法を示しています。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteUserPermissionsBoundary](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteUserPolicy** を使用する

以下のコード例は、DeleteUserPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ユーザーを作成してロールを引き受ける](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
///  
/// <summary>
```

```
/// Delete an IAM user policy.  
/// </summary>  
/// <param name="policyName">The name of the IAM policy to delete.</param>  
/// <param name="userName">The username of the IAM user.</param>  
/// <returns>A Boolean value indicating the success of the action.</returns>  
public async Task<bool> DeleteUserPolicyAsync(string policyName, string  
userName)  
{  
    var response = await _IAMService.DeleteUserPolicyAsync(new  
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });  
  
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーからポリシーを削除するには

次の delete-user-policy コマンドは、指定されたポリシーを Bob という名前の IAM ユーザーから削除します。

```
aws iam delete-user-policy \  
  --user-name Bob \  
  --policy-name ExamplePolicy
```

このコマンドでは何も出力されません。


IAM ユーザーのポリシーのリストを取得するには、list-user-policies コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
error {
    _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
&iam.DeleteUserPolicyInput{
    PolicyName: aws.String(policyName),
    UserName:   aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
err)
    }
    return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーに埋め込まれている **AccessToEC2Policy** という名前のインラインポリシーを削除します。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

例 2: この例では、**Theresa** という名前の IAM ユーザーに埋め込まれているすべてのインラインポリシーを検索し、削除します。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
  Theresa -Force}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Deletes a user and their associated resources
#
# @param user_name [String] The name of the user to delete
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
  user.each do |key|
    @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end
end
```



```
end

@iam_client.delete_user(user_name: user_name)
@logger.info("Deleted user '#{user_name}'.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting user '#{user_name}': #{e.message}")
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
throws {
    let input = DeleteUserPolicyInput(
        policyName: policyName,
        userName: user.userName
    )
    do {
        _ = try await iamClient.deleteUserPolicy(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **DeleteVirtualMfaDevice** を使用する

以下のコード例は、DeleteVirtualMfaDevice の使用方法を示しています。

## CLI

### AWS CLI

仮想 MFA デバイスを削除するには

次の `delete-virtual-mfa-device` コマンドは、指定した MFA デバイスを現在のアカウントから削除します。

```
aws iam delete-virtual-mfa-device \  
  --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[MFA デバイスの無効化](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVirtualMfaDevice](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:mfa/bob` である IAM 仮想 MFA デバイスを削除します。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

例 2: この例では、IAM ユーザー Theresa に MFA デバイスが割り当てられているかどうかを確認します。割り当てられているデバイスが見つかった場合、そのデバイスはその IAM ユーザーに対して無効になります。デバイスが仮想の場合は、そのデバイスも削除されます。

```
$mfa = Get-IAMMFADevice -UserName Theresa  
if ($mfa) {  
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name  
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -  
SerialNumber $mfa.SerialNumber }  
}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DeleteVirtualMfaDevice](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DetachGroupPolicy** を使用する

以下のコード例は、DetachGroupPolicy の使用方法を示しています。

### CLI

#### AWS CLI

グループからポリシーをデタッチするには

この例では、ARN `arn:aws:iam::123456789012:policy/TesterAccessPolicy` を持つ管理ポリシーを Testers というグループから削除します。

```
aws iam detach-group-policy \  
  --group-name Testers \  
  --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachGroupPolicy](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:policy/TesterAccessPolicy` である管理グループポリシーを Testers という名前のグループからデタッチします。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

例 2: この例では、**Testers** という名前のグループにアタッチされているすべての管理ポリシーを検索し、グループからデタッチします。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DetachGroupPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DetachRolePolicy** を使用する

以下のコード例は、DetachRolePolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ユーザーを作成してロールを引き受ける](#)
- [ロールの管理](#)

### .NET

#### AWS SDK for .NET

##### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Detach an IAM policy from an IAM role.
/// </summary>
```

```

    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

```

```
#####
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a role.
#
# Parameters:
#   -n role_name -- The name of the IAM role.
#   -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_detach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_detach_role_policy"
        echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_ARN -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1
}
```

```
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"


return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DetachRolePolicy](#)」を参照してください。



## C++

## SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
detachRequest.SetRoleName(roleName);
detachRequest.SetPolicyArn(policyArn);

auto detachOutcome = iam.DetachRolePolicy(detachRequest);
if (!detachOutcome.IsSuccess()) {
    std::cerr << "Failed to detach policy " << policyArn << " from role "
              << roleName << ": " << detachOutcome.GetError().GetMessage() <<
              std::endl;
}
else {
    std::cout << "Successfully detached policy " << policyArn << " from role "
              << roleName << std::endl;
}

return detachOutcome.IsSuccess();
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## CLI

## AWS CLI

ロールからポリシーをデタッチするには

この例では、ARN `arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy` を持つ管理ポリシーを `FedTesterRole` というロールから削除します。

```
aws iam detach-role-policy \  
  --role-name FedTesterRole \  
  --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachRolePolicy](#)」を参照してください。

Go

SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {  
  iamClient *iam.Client  
}  
  
// DetachRolePolicy detaches a policy from a role.  
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)  
  error {  
  _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),  
    &iam.DetachRolePolicyInput{  
      PolicyArn: aws.String(policyArn),
```

```
    RoleName:  aws.String(roleName),
  })
  if err != nil {
    log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
err)
  }
  return err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <roleName> <policyArn>\s

Where:
    roleName - A role name that you can obtain from the AWS
Management Console.\s
    policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String roleName = args[0];
String policyArn = args[1];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();
detachPolicy(iam, roleName, policyArn);
System.out.println("Done");
iam.close();
}

public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーをデタッチします。

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";


const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## SDK for JavaScript (v2)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

```
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
        $roleNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** である管理グループポリシーを **FedTesterRole** という名前のロールからデタッチします。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

例 2: この例では、**FedTesterRole** という名前のロールにアタッチされているすべての管理ポリシーを検索し、ロールからデタッチします。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
-Rolename FedTesterRole
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

Boto3 Policy オブジェクトを使用して、ロールからポリシーをデタッチします。

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
```



```
"""
try:
    iam.Policy(policy_arn).detach_role(RoleName=role_name)
    logger.info("Detached policy %s from role %s.", policy_arn, role_name)
except ClientError:
    logger.exception(
        "Couldn't detach policy %s from role %s.", policy_arn, role_name
    )
    raise
```

Boto3 Role オブジェクトを使用して、ロールからポリシーをデタッチします。

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. Note this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- API の詳細については、「[AWSSDK for Python \(Boto3\) API リファレンス](#)」の「[DetachRolePolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
end
```

```
# @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
def get_policy(policy_arn)
  response = @iam_client.get_policy(policy_arn: policy_arn)
  policy = response.policy
  @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

# Attaches a policy to a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_role(role_name, policy_arn)
  @iam_client.attach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error attaching policy to role: #{e.message}")
  false
end

# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

```
# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def detach_policy_from_role(role_name, policy_arn)
  @iam_client.detach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error detaching policy from role: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn detach_role_policy(
  client: &iamClient,
  role_name: &str,
  policy_arn: &str,
) -> Result<(), iamError> {
  client
    .detach_role_policy()
    .role_name(role_name)
    .policy_arn(policy_arn)
```

```
        .send()
        .await?;

    Ok(())
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
IAMClientTypes.Role) async throws {
    let input = DetachRolePolicyInput(
        policyArn: policy.arn,
        roleName: role.roleName
    )

    do {
        _ = try await iamClient.detachRolePolicy(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DetachUserPolicy** を使用する

以下のコード例は、DetachUserPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

### CLI

#### AWS CLI

ユーザーからポリシーをデタッチするには

この例では、ARN `arn:aws:iam::123456789012:policy/TesterPolicy` を持つ管理ポリシーをユーザー Bob から削除します。

```
aws iam detach-user-policy \  
  --user-name Bob \  
  --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセス許可の変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachUserPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/TesterPolicy** である管理ポリシーを **Bob** という名前の IAM ユーザーからデタッチします。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
arn:aws:iam::123456789012:policy/TesterPolicy
```

例 2: この例では、**Theresa** という名前の IAM ユーザーにアタッチされているすべての管理ポリシーを検索し、それらのポリシーをユーザーからデタッチします。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DetachUserPolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
```

```
logger.exception(  
    "Couldn't detach policy %s from user %s.", policy_arn, user_name  
)  
raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DetachUserPolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Detaches a policy from a user  
#  
# @param user_name [String] The name of the user  
# @param policy_arn [String] The ARN of the policy to detach  
# @return [Boolean] true if the policy was successfully detached, false  
# otherwise  
def detach_user_policy(user_name, policy_arn)  
  @iam_client.detach_user_policy(  
    user_name: user_name,  
    policy_arn: policy_arn  
  )  
  @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'  
successfully.")  
  true  
rescue Aws::IAM::Errors::NoSuchEntity  
  @logger.error("Error detaching policy: Policy or user does not exist.")  
  false  
rescue Aws::IAM::Errors::ServiceError => e  
  @logger.error("Error detaching policy from user '#{user_name}':  
#{e.message}")
```



```
false
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DetachUserPolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DetachUserPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `EnableMfaDevice` を使用する

以下のコード例は、`EnableMfaDevice` の使用方法を示しています。

### CLI

#### AWS CLI

MFA デバイスを有効にするには

`create-virtual-mfa-device` コマンドを使用して新しい仮想 MFA デバイスを作成したら、その MFA デバイスをユーザーに割り当てることができます。次の `enable-mfa-device` の例では、シリアル番号 `arn:aws:iam::210987654321:mfa/BobsMFADevice` を持つ MFA デバイスをユーザー Bob に割り当てます。また、このコマンドは、仮想 MFA デバイスからの最初の 2 つのコードを順番に含めることで、デバイスを AWS と同期します。

```
aws iam enable-mfa-device \  
  --user-name Bob \  
  --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \  
  --authentication-code1 123456 \  
  --authentication-code2 789012
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableMfaDevice](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: このコマンドは、シリアル番号 `987654321098` を持つハードウェア MFA デバイスを有効にし、このデバイスをユーザー **Bob** に関連付けます。また、このコマンドは、デバイスからの最初の 2 つのコードを順番に含めます。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -  
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

例 2: この例では、仮想 MFA デバイスを作成して有効にします。最初のコマンドは仮想デバイスを作成し、そのデバイスのオブジェクト表現を変数 `$MFADevice` に返します。`.Base32StringSeed` または `QRCodePng` プロパティを使用して、ユーザーのソフトウェアアプリケーションを設定できます。最後のコマンドはデバイスをユーザー **David** に割り当て、デバイスをシリアル番号で識別します。また、このコマンドは、仮想 MFA デバイスからの最初の 2 つのコードを順番に含めることで、デバイスを AWS と同期します。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"  
# see example for New-IAMVirtualMFADevice to see how to configure the software  
# program with PNG or base32 seed code  
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber  
$MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2  
"13572468"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[EnableMfaDevice](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GenerateCredentialReport` を使用する

以下のコード例は、`GenerateCredentialReport` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

### CLI

#### AWS CLI

認証情報レポートを生成するには

次の例では、AWS アカウントについての認証情報レポートの生成を試みます。

```
aws iam generate-credential-report
```

出力:

```
{
  "State": "STARTED",
  "Description": "No report exists. Starting a new report generation task"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウントの認証情報レポートの取得](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GenerateCredentialReport](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、4 時間ごとに実行できる新しいレポートの生成を要求しています。最新のレポートがまだ新しい場合、[状態] フィールドには **COMPLETE** と表示されます。**Get-IAMCredentialReport** を使用して、完成したレポートを表示します。

```
Request-IAMCredentialReport
```

出力:

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GenerateCredentialReport](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
    hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
%s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
account.")
        raise
    else:
        return response
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GenerateCredentialReport](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GenerateServiceLastAccessedDetails` を使用する

以下のコード例は、`GenerateServiceLastAccessedDetails` の使用方法を示しています。

### CLI

#### AWS CLI

例 1: カスタムポリシーのサービスアクセスレポートを生成するには

次の `generate-service-last-accessed-details` の例では、バックグラウンドジョブを開始し、`intern-boundary` という名前のカスタムポリシーを使用して IAM ユーザーやその他のエンティティがアクセスしたサービスを一覧表示するレポートを生成します。`get-service-last-accessed-details` コマンドを実行すると、レポートを作成した後に表示できます。

```
aws iam generate-service-last-accessed-details \  
  --arn arn:aws:iam::123456789012:policy/intern-boundary
```

出力:

```
{  
  "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"  
}
```

例 2: AWS 管理 AdministratorAccess ポリシーのサービスアクセスレポートを生成するには

次の `generate-service-last-accessed-details` の例では、バックグラウンドジョブを開始し、AWS 管理 AdministratorAccess ポリシーを使用して IAM ユーザーやその他のエンティティがアクセスしたサービスを一覧表示するレポートを生成します。`get-service-last-accessed-details` コマンドを実行すると、レポートを作成した後に表示できます。

```
aws iam generate-service-last-accessed-details \  
  --arn arn:aws:iam::aws:policy/AdministratorAccess
```

出力:

```
{
```

```
"JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GenerateServiceLastAccessedDetails](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例は GenerateServiceLastAccessedDetails API と同等のコマンドレットです。これにより、Get-IAMServiceLastAccessedDetail と Get-IAMServiceLastAccessedDetailWithEntity で使用できるジョブ ID が提供されます。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GenerateServiceLastAccessedDetails](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetAccessKeyLastUsed** を使用する

以下のコード例は、GetAccessKeyLastUsed の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスキーの管理](#)

## C++

## SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
iam.GetAccessKeyLastUsed(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
            secretKeyID << ":" << outcome.GetError().GetMessage() <<
std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
            outcome.GetResult()
                .GetAccessKeyLastUsed()
                .GetLastUsedDate()
                .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
            lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```



- API の詳細については、「AWS SDK for C++ SDK for Rust API リファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

## CLI

### AWS CLI

指定されたアクセスキーの最後の使用時の情報を取得するには

次の例では、アクセスキー ABCDEXAMPLE が最後に使用されたときに関する情報を取得します。

```
aws iam get-access-key-last-used \  
  --access-key-id ABCDEXAMPLE
```

出力:

```
{  
  "UserName": "Bob",  
  "AccessKeyLastUsed": {  
    "Region": "us-east-1",  
    "ServiceName": "iam",  
    "LastUsedDate": "2015-06-16T22:45:00Z"  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アクセスキーを取得します。

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });


  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Rust API リファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

SDK for JavaScript (v2)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Rust API リファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 指定されたアクセスキーの所有者名と最終使用情報を返します。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetAccessKeyLastUsed](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GetAccountAuthorizationDetails` を使用する

以下のコード例は、`GetAccountAuthorizationDetails` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

### CLI

#### AWS CLI

AWS アカウントの IAM ユーザー、グループ、ロール、ポリシーを一覧表示するには

次の `get-account-authorization-details` コマンドは、AWS アカウント内のすべての IAM ユーザー、グループ、ロール、ポリシーに関する情報を返します。

```
aws iam get-account-authorization-details
```

出力:

```
{
  "RoleDetailList": [
    {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
    }
  ],
}
```

```
"RoleId": "ARO1234567890EXAMPLE",
"CreateDate": "2014-07-30T17:09:20Z",
"InstanceProfileList": [
  {
    "InstanceProfileId": "AIPA1234567890EXAMPLE",
    "Roles": [
      {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "",
              "Effect": "Allow",
              "Principal": {
                "Service": "ec2.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        },
        "RoleId": "ARO1234567890EXAMPLE",
        "CreateDate": "2014-07-30T17:09:20Z",
        "RoleName": "EC2role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/EC2role"
      }
    ],
    "CreateDate": "2014-07-30T17:09:20Z",
    "InstanceProfileName": "EC2role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
  }
],
"RoleName": "EC2role",
"Path": "/",
"AttachedManagedPolicies": [
  {
    "PolicyName": "AmazonS3FullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
  },
  {
    "PolicyName": "AmazonDynamoDBFullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
```

```
    }
  ],
  "RoleLastUsed": {
    "Region": "us-west-2",
    "LastUsedDate": "2019-11-13T17:30:00Z"
  },
  "RolePolicyList": [],
  "Arn": "arn:aws:iam::123456789012:role/EC2role"
}
],
"GroupDetailList": [
  {
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    "GroupName": "Admins",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "CreateDate": "2013-10-14T18:32:24Z",
    "GroupPolicyList": []
  },
  {
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": {
      "PolicyName": "PowerUserAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
    },
    "GroupName": "Dev",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Dev",
    "CreateDate": "2013-10-14T18:33:55Z",
    "GroupPolicyList": []
  },
  {
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": [],
    "GroupName": "Finance",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Finance",
    "CreateDate": "2013-10-14T18:57:48Z",
    "GroupPolicyList": [
      {
```

```
        "PolicyName": "policygen-201310141157",
        "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "aws-portal:*",
                    "Sid": "Stmt1381777017000",
                    "Resource": "*",
                    "Effect": "Allow"
                }
            ]
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
```



```
        "aws-portal:*",
        "iam:*"
    ],
    "Resource": "*"
}
}
},
"Path": "/",
"AttachedManagedPolicies": [],
"Arn": "arn:aws:iam::123456789012:user/Bob"
},
{
    "UserName": "Charlie",
    "GroupList": [
        "Dev"
    ],
    "CreateDate": "2013-10-14T18:33:56Z",
    "UserId": "AIDA1234567890EXAMPLE",
    "UserPolicyList": [],
    "Path": "/",
    "AttachedManagedPolicies": [],
    "Arn": "arn:aws:iam::123456789012:user/Charlie"
}
],
"Policies": [
    {
        "PolicyName": "create-update-delete-set-managed-policies",
        "CreateDate": "2015-02-06T19:58:34Z",
        "AttachmentCount": 1,
        "IsAttachable": true,
        "PolicyId": "ANPA1234567890EXAMPLE",
        "DefaultVersionId": "v1",
        "PolicyVersionList": [
            {
                "CreateDate": "2015-02-06T19:58:34Z",
                "VersionId": "v1",
                "Document": {
                    "Version": "2012-10-17",
                    "Statement": {
                        "Effect": "Allow",
                        "Action": [
                            "iam:CreatePolicy",
                            "iam:CreatePolicyVersion",
```

```

        "iam:DeletePolicy",
        "iam:DeletePolicyVersion",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListPolicies",
        "iam:ListPolicyVersions",
        "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
}
},
"IsDefaultVersion": true
}
],
"Path": "/",
"Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
"UpdateDate": "2015-02-06T19:58:34Z"
},
{
    "PolicyName": "S3-read-only-specific-bucket",
    "CreateDate": "2015-01-21T21:39:41Z",
    "AttachmentCount": 1,
    "IsAttachable": true,
    "PolicyId": "ANPA1234567890EXAMPLE",
    "DefaultVersionId": "v1",
    "PolicyVersionList": [
        {
            "CreateDate": "2015-01-21T21:39:41Z",
            "VersionId": "v1",
            "Document": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": [
                            "s3:Get*",
                            "s3:List*"
                        ],
                        "Resource": [
                            "arn:aws:s3:::example-bucket",
                            "arn:aws:s3:::example-bucket/*"
                        ]
                    }
                ]
            }
        }
    ]
}
}

```

```
        ],
        },
        "IsDefaultVersion": true
    }
],
"Path": "/",
"Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
"UpdateDate": "2015-01-21T23:39:41Z"
},
{
    "PolicyName": "AmazonEC2FullAccess",
    "CreateDate": "2015-02-06T18:40:15Z",
    "AttachmentCount": 1,
    "IsAttachable": true,
    "PolicyId": "ANPA1234567890EXAMPLE",
    "DefaultVersionId": "v1",
    "PolicyVersionList": [
        {
            "CreateDate": "2014-10-30T20:59:46Z",
            "VersionId": "v1",
            "Document": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Action": "ec2:*",
                        "Effect": "Allow",
                        "Resource": "*"
                    },
                    {
                        "Effect": "Allow",
                        "Action": "elasticloadbalancing:*",
                        "Resource": "*"
                    },
                    {
                        "Effect": "Allow",
                        "Action": "cloudwatch:*",
                        "Resource": "*"
                    },
                    {
                        "Effect": "Allow",
                        "Action": "autoscaling:*",
                        "Resource": "*"
                    }
                ]
            }
        }
    ]
}
```

```
        ],
        },
        "IsDefaultVersion": true
    }
],
"Path": "/",
"Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
"UpdateDate": "2015-02-06T18:40:15Z"
}
],
"Marker": "EXAMPLEkakov9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
"IsTruncated": true
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS セキュリティ監査ガイドライン](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetAccountAuthorizationDetails](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウント内の ID に関する認証の詳細を取得し、返されたオブジェクトの要素リスト (ユーザー、グループ、ロールを含む) を表示します。例えば、**UserDetailList** プロパティには、ユーザーに関する詳細が表示されます。同様の情報は、**RoleDetailList** および **GroupDetailList** プロパティで入手可能です。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

出力:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker         :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

出力:

```
Arn          : arn:aws:iam::123456789012:user/Administrator
CreateDate   : 10/16/2014 9:03:09 AM
GroupList    : {Administrators}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE1
UserName     : Administrator
UserPolicyList : {}

Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/6/2015 12:54:42 PM
GroupList    : {Developers}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE2
UserName     : bab
UserPolicyList : {}

Arn          : arn:aws:iam::123456789012:user/BackupToS3
CreateDate   : 1/27/2015 10:15:08 AM
GroupList    : {Backup}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE3
UserName     : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetAccountAuthorizationDetails](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
    such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetAccountAuthorizationDetails](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetAccountPasswordPolicy** を使用する

以下のコード例は、GetAccountPasswordPolicy の使用方法を示しています。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Gets the IAM password policy for an AWS account.
/// </summary>
/// <returns>The PasswordPolicy for the AWS account.</returns>
public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
{
    var response = await _IAMService.GetAccountPasswordPolicyAsync(new
    GetAccountPasswordPolicyRequest());
    return response.PasswordPolicy;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## CLI

### AWS CLI

現在のアカウントのパスワードポリシーを表示するには

次の `get-account-password-policy` コマンドは、現在のアカウントのパスワードポリシーに関する詳細を表示します。

```
aws iam get-account-password-policy
```

出力:

```
{
  "PasswordPolicy": {
```

```
    "AllowUsersToChangePassword": false,  
    "RequireLowercaseCharacters": false,  
    "RequireUppercaseCharacters": false,  
    "MinimumPasswordLength": 8,  
    "RequireNumbers": true,  
    "RequireSymbols": true  
  }  
}
```


アカウントのためにパスワードポリシーが定義されていない場合、コマンドは `NoSuchEntity` エラーを返します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアカウントパスワードポリシーの設定](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account  
// actions  
// used in the examples.  
// It contains an IAM service client that is used to perform account actions.  
type AccountWrapper struct {  
    iamClient *iam.Client  
}  
  
// GetAccountPasswordPolicy gets the account password policy for the current  
// account.
```



```
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
error) {
    var pwPolicy *types.PasswordPolicy
    result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
    &iam.GetAccountPasswordPolicyInput{})
    if err != nil {
        log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
    } else {
        pwPolicy = result.PasswordPolicy
    }
    return pwPolicy, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントのパスワードポリシーを取得します。

```
import {
    GetAccountPasswordPolicyCommand,
    IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
    const command = new GetAccountPasswordPolicyCommand({});

    const response = await client.send(command);
```

```
console.log(response.PasswordPolicy);
return response;
};
```

- APIの詳細については、「AWS SDK for JavaScript SDK for Kotlin API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function getAccountPasswordPolicy()
{
    return $this->iamClient->getAccountPasswordPolicy();
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在のアカウントのパスワードポリシーに関する詳細を返します。アカウントのためにパスワードポリシーが定義されていない場合、コマンドは **NoSuchEntity** エラーを返します。

```
Get-IAMAccountPasswordPolicy
```


## 出力:

```
AllowUsersToChangePassword : True
ExpirePasswords             : True
HardExpiry                  : False
MaxPasswordAge              : 90
MinimumPasswordLength       : 8
PasswordReusePrevention     : 20
RequireLowercaseCharacters  : True
RequireNumbers               : True
RequireSymbols               : False
RequireUppercaseCharacters  : True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## Python

## SDK for Python (Boto3)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
{pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
```

```
print(f"\tpassword_reuse_prevention:
{pw_policy.password_reuse_prevention}")
print(
    f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}"
)
print(f"\trequire_numbers: {pw_policy.require_numbers}")
print(f"\trequire_symbols: {pw_policy.require_symbols}")
print(
    f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}"
)
printed = True
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchEntity":
        print("The account does not have a password policy set.")
    else:
        logger.exception("Couldn't get account password policy.")
        raise
else:
    return printed
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger
```

```
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
  @logger.progname = "IAMPolicyManager"
end

# Retrieves and logs the account password policy
def print_account_password_policy
  begin
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is:
#{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code}
- #{e.message}")
      raise
    end
  end
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn get_account_password_policy(
  client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
SdkError<GetAccountPasswordPolicyError>> {
```

```
let response = client.get_account_password_policy().send().await?;  
  
Ok(response)  
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetAccountSummary** を使用する

以下のコード例は、GetAccountSummary の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

### CLI

#### AWS CLI

現在のアカウントの IAM エンティティの使用状況と IAM クォータに関する情報を取得するには

次の `get-account-summary` コマンドは、アカウント内の現在の IAM エンティティの使用状況と現在の IAM エンティティのクォータに関する情報を返します。

```
aws iam get-account-summary
```

出力:

```
{  
  "SummaryMap": {  
    "UsersQuota": 5000,  
    "GroupsQuota": 100,  
  }  
}
```

```
"InstanceProfiles": 6,  
"SigningCertificatesPerUserQuota": 2,  
"AccountAccessKeysPresent": 0,  
"RolesQuota": 250,  
"RolePolicySizeQuota": 10240,  
"AccountSigningCertificatesPresent": 0,  
"Users": 27,  
"ServerCertificatesQuota": 20,  
"ServerCertificates": 0,  
"AssumeRolePolicySizeQuota": 2048,  
"Groups": 7,  
"MFADevicesInUse": 1,  
"Roles": 3,  
"AccountMFAEnabled": 1,  
"MFADevices": 3,  
"GroupsPerUserQuota": 10,  
"GroupPolicySizeQuota": 5120,  
"InstanceProfilesQuota": 100,  
"AccessKeysPerUserQuota": 2,  
"Providers": 0,  
"UserPolicySizeQuota": 2048  
}  
}
```

エンティティ制限の詳細については、「AWS IAM ユーザーガイド」の「[IAM および AWS STS クォータ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetAccountSummary](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウント内の現在の IAM エンティティの使用状況と現在の IAM エンティティのクォータに関する情報を返します。

```
Get-IAMAccountSummary
```

出力:

Key	Value
Users	7
GroupPolicySizeQuota	5120
PolicyVersionsInUseQuota	10000
ServerCertificatesQuota	20
AccountSigningCertificatesPresent	0
AccountAccessKeysPresent	0
Groups	3
UsersQuota	5000
RolePolicySizeQuota	10240
UserPolicySizeQuota	2048
GroupsPerUserQuota	10
AssumeRolePolicySizeQuota	2048
AttachedPoliciesPerGroupQuota	2
Roles	9
VersionsPerPolicyQuota	5
GroupsQuota	100
PolicySizeQuota	5120
Policies	5
RolesQuota	250
ServerCertificates	0
AttachedPoliciesPerRoleQuota	2
MFADevicesInUse	2
PoliciesQuota	1000
AccountMFAEnabled	1
Providers	2
InstanceProfilesQuota	100
MFADevices	4
AccessKeysPerUserQuota	2
AttachedPoliciesPerUserQuota	2
SigningCertificatesPerUserQuota	2
PolicyVersionsInUse	4
InstanceProfiles	1
...	

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetAccountSummary](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetAccountSummary](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **GetContextKeysForCustomPolicy** を使用する

以下のコード例は、GetContextKeysForCustomPolicy の使用方法を示しています。

## CLI

## AWS CLI

例 1: コマンドラインでパラメータとして指定された 1 つ以上のカスタム JSON ポリシーによって参照されるコンテキストキーを一覧表示するには

次の `get-context-keys-for-custom-policy` コマンドは、指定された各ポリシーを解析し、それらのポリシーが使用するコンテキストキーを一覧表示します。このコマンドを使用して、ポリシーシミュレータコマンド `simulate-custom-policy` と `simulate-custom-policy` を正常に使用するために指定する必要があるコンテキストキー値を特定します。また、`get-context-keys-for-custom-policy` コマンドを使用して、IAM ユーザーまたはロールに関連付けられたすべてのポリシーで使用されるコンテキストキーのリストを取得することもできます。 `file://` で始まるパラメータ値は、ファイルを読み取り、ファイル名自体ではなくその内容をパラメータの値として使用するようにコマンドに指示します。

```
aws iam get-context-keys-for-custom-policy \  
  --policy-input-list '{"Version":"2012-10-17","Statement":  
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-  
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":  
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

出力:

```
{  
  "ContextKeyNames": [  
    "aws:username",  
    "aws:CurrentTime"  
  ]  
}
```

例 2: ファイル入力として提供される 1 つ以上のカスタム JSON ポリシーによって参照されるコンテキストキーを一覧表示するには

次の `get-context-keys-for-custom-policy` コマンドは前の例と同じですが、ポリシーがパラメータとしてではなくファイルで提供される点が異なります。このコマンドは JSON 構造のリストではなく JSON 文字列のリストを想定しているため、ファイルは次のような構造にする必要があります。ただし、1 つにまとめることはできます。

```
[  
  "Policy1",
```

```
"Policy2"  
]
```

そのため、例えば、前の例のポリシーを含むファイルは次のようになっている必要があります。ポリシー文字列内に埋め込まれている各二重引用符の前に「バックスラッシュ」を付けてエスケープする必要があります。

```
[ {"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Action": "dynamodb:*", "Resource": "arn:aws:dynamodb:us-west-2:128716708097:table/${aws:username}", "Condition": {"DateGreaterThan": {"aws:CurrentTime": "2015-08-16T12:00:00Z"}}}] ]
```

その後、このファイルを次のコマンドに送信できます。

```
aws iam get-context-keys-for-custom-policy \  
--policy-input-list file://policyfile.json
```

出力:

```
{  
  "ContextKeyNames": [  
    "aws:username",  
    "aws:CurrentTime"  
  ]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM Policy Simulator の使用 \(AWS CLI および AWS API\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetContextKeysForCustomPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、提供されたポリシー JSON に含まれるすべてのコンテキストキーを取得します。複数のポリシーを指定するには、値のカンマ区切りリストとして指定できます。

```
$policy1 = '{"Version":"2012-10-17","Statement":  
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
```

```
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}]}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetContextKeysForCustomPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GetContextKeysForPrincipalPolicy` を使用する

以下のコード例は、`GetContextKeysForPrincipalPolicy` の使用方法を示しています。

### CLI

#### AWS CLI

IAM プリンシパルに関連付けられているすべてのポリシーが参照するコンテキストキーを一覧表示するには

次の `get-context-keys-for-principal-policy` コマンドは、ユーザー `saanvi` とそのユーザーがメンバーとなっているグループにアタッチされているすべてのポリシーを取得します。次に、このコマンドは各ポリシーを解析し、それらのポリシーで使用されているコンテキストキーを一覧表示します。このコマンドを使用して、`simulate-custom-policy` および `simulate-principal-policy` コマンドを正常に使用するために指定する必要があるコンテキストキー値を特定します。`get-context-keys-for-custom-policy` コマンドを使用して、任意の JSON ポリシーで使用されているコンテキストキーのリストを取得することもできます。

```
aws iam get-context-keys-for-principal-policy \
  --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

出力:

```
{
```

```
"ContextKeyNames": [  
  "aws:username",  
  "aws:CurrentTime"  
]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM Policy Simulator の使用 \(AWS CLI および AWS API\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetContextKeysForPrincipalPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、提供されたポリシー JSON に含まれるすべてのコンテキストキーと IAM エンティティ (ユーザー/ロールなど) にアタッチされたポリシーを取得します。-PolicyInputList では、複数の値リストをカンマ区切り値として指定できます。

```
$policy1 = '{"Version":"2012-10-17","Statement":  
{  
  "Effect":"Allow",  
  "Action":"dynamodb:*",  
  "Resource":"arn:aws:dynamodb:us-west-2:123456789012:table/",  
  "Condition":{"DateGreaterThan":  
    {"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'  
$policy2 = '{"Version":"2012-10-17","Statement":  
{  
  "Effect":"Allow",  
  "Action":"dynamodb:*",  
  "Resource":"arn:aws:dynamodb:us-west-2:123456789012:table/"}}'  
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -  
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetContextKeysForPrincipalPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetCredentialReport** を使用する

以下のコード例は、GetCredentialReport の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

## CLI

### AWS CLI

認証情報レポートを取得するには

この例では、返されたレポートを開き、それをテキスト行の配列としてパイプラインに出力します。

```
aws iam get-credential-report
```

出力:

```
{
  "GeneratedTime": "2015-06-17T19:11:50Z",
  "ReportFormat": "text/csv"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウントの認証情報レポートの取得](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetCredentialReport](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、返されたレポートを開き、それをテキスト行の配列としてパイプラインに出力します。最初の行は、カンマで区切られた列名のヘッダーです。連続する各行は 1 人のユーザーの詳細行で、各フィールドはカンマで区切られています。レポートを表示するには、**Request-IAMCredentialReport** コマンドレットを使用してレポートを生成する必要があります。レポートを 1 つの文字列として取得するには、**-AsTextArray** ではなく **-Raw** を使用します。**-AsTextArray** スイッチには、エイリアス **-SplitLines** も使用できます。出力の列の完全なリストについては、サービス API リファレンスを参照してください。-

**AsTextArray** または **-SplitLines** を使用しない場合は、**.NET StreamReader** クラスを使用して、**.Content** プロパティからテキストを抽出する必要があることに注意してください。

```
Request-IAMCredentialReport
```

出力:

Description	State
----- No report exists. Starting a new report generation task	----- STARTED

```
Get-IAMCredentialReport -AsTextArray
```

出力:

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,password_last_changed,pa
root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04-20T15:18:32+00:00,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00,2014-10-16T16:06:00,A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00,2014-10-16T16:06:00,A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetCredentialReport](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
    account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetCredentialReport](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetGroup** を使用する

以下のコード例は、GetGroup の使用方法を示しています。

### CLI

#### AWS CLI

IAM グループを取得するには

この例では、IAM グループ Admins に関する詳細を返します。

```
aws iam get-group \  
  --group-name Admins
```

出力:



```
{
  "Group": {
    "Path": "/",
    "CreateDate": "2015-06-16T19:41:48Z",
    "GroupId": "AIDGPMS9R04H3FEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "GroupName": "Admins"
  },
  "Users": []
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ID \(ユーザー、ユーザーグループ、ロール\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、グループに属するすべての IAM ユーザーのコレクションを含む、IAM グループ **Testers** に関する詳細を返します。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

出力:

Group	IsTruncated	Marker
Users		
-----	-----	-----
-----		
Amazon.IdentityManagement.Model.Group {Theresa, David}	False	

```
$results.Group
```

出力:

```
Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName : Testers
Path      : /
```

```
$results.Users
```

出力:

```
Arn      : arn:aws:iam::123456789012:user/Theresa
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path      : /
UserId    : 40SVDDJJTF4XEEXAMPLE2
UserName  : Theresa

Arn      : arn:aws:iam::123456789012:user/David
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path      : /
UserId    : Y4FKWQCXTA52QEXAMPLE3
UserName  : David
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetGroupPolicy** を使用する

以下のコード例は、GetGroupPolicy の使用方法を示しています。

CLI

AWS CLI

IAM グループにアタッチされたポリシーに関する情報を取得するには

次の `get-group-policy` コマンドは、`Test-Group` という名前のグループにアタッチされている指定されたポリシーに関する情報を取得します。

```
aws iam get-group-policy \  
  --group-name Test-Group \  
  --policy-name S3-ReadOnly-Policy
```

出力:

```
{  
  "GroupName": "Test-Group",  
  "PolicyDocument": {  
    "Statement": [  
      {  
        "Action": [  
          "s3:Get*",  
          "s3:List*"  
        ],  
        "Resource": "*",  
        "Effect": "Allow"  
      }  
    ]  
  },  
  "PolicyName": "S3-ReadOnly-Policy"  
}
```

IAM ポリシーの詳細については、「AWS IAM ユーザーガイド」の「[IAM ポリシーを管理する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetGroupPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、グループ `Testers` の `PowerUserAccess-Testers` という名前の埋め込みインラインポリシーに関する詳細を返します。`PolicyDocument` プロパティは URL エンコードされています。この例では、`UrlDecode` .NET メソッドを使用してデコードされています。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

出力:

```
GroupName      PolicyDocument
PolicyName
-----
-----
Testers        %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetGroupPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetInstanceProfile** を使用する

以下のコード例は、GetInstanceProfile の使用方法を示しています。

## CLI

## AWS CLI

インスタンスプロファイルの情報を取得するには

次の `get-instance-profile` コマンドは、`ExampleInstanceProfile` という名前のインスタンスプロファイルに関する情報を取得します。

```
aws iam get-instance-profile \
  --instance-profile-name ExampleInstanceProfile
```

出力:

```
{
  "InstanceProfile": {
    "InstanceId": "AID2MAB8DPLSRHEXAMPLE",
    "Roles": [
      {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AIDGPM9R04H3FEXAMPLE",
        "CreateDate": "2013-01-09T06:33:26Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:role/Test-Role"
      }
    ],
    "CreateDate": "2013-06-12T23:52:02Z",
    "InstanceProfileName": "ExampleInstanceProfile",
    "Path": "/",
    "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
  }
}
```

詳細については、「AWS IAM ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetInstanceProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントで定義されている **ec2instancerole** という名前のインスタンスプロファイルの詳細を返します。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

出力:

```
Arn          : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate   : 2/17/2015 2:49:04 PM
InstanceProfileId : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles       : {ec2instancerole}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetInstanceProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **GetLoginProfile** を使用する

以下のコード例は、GetLoginProfile の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ユーザーのパスワード情報を取得するには

次の get-login-profile コマンドは、Bob という名前の IAM ユーザーのパスワードに関する情報を取得します。

```
aws iam get-login-profile \  
  --user-name Bob
```

出力:

```
{
  "LoginProfile": {
    "UserName": "Bob",
    "CreateDate": "2012-09-21T23:03:39Z"
  }
}
```

get-login-profile コマンドを使用して、IAM ユーザーがパスワードを持っていることを確認できます。ユーザーのためにパスワードが定義されていない場合、コマンドは NoSuchEntity エラーを返します。

このコマンドを使用してパスワードを表示することはできません。パスワードを忘れた場合は、ユーザーのパスワードをリセットできます (update-login-profile)。または、ユーザーのログインプロファイルを削除し (delete-login-profile)、新しいログインプロファイルを作成することもできます (create-login-profile)。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのパスワードの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetLoginProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、パスワードの作成日と、IAM ユーザー **David** のパスワードのリセットが必要かどうかを返します。

```
Get-IAMLoginProfile -UserName David
```

出力:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
12/10/2014 3:39:44 PM	False	David

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetLoginProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GetOpenIdConnectProvider` を使用する

以下のコード例は、`GetOpenIdConnectProvider` の使用方法を示しています。

### CLI

#### AWS CLI

指定した OpenID Connect プロバイダーに関する情報を返すには

この例では、ARN が `arn:aws:iam::123456789012:oidc-provider/server.example.com` である OpenID Connect プロバイダーに関する詳細を返します。

```
aws iam get-open-id-connect-provider \
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
  server.example.com
```

出力:

```
{
  "Url": "server.example.com"
  "CreateDate": "2015-06-16T19:41:48Z",
  "ThumbprintList": [
    "12345abcdefghijk67890lmnopqrst987example"
  ],
  "ClientIDList": [
    "example-application-ID"
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetOpenIdConnectProvider](#)」を参照してください。



## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** である OpenID Connect プロバイダーに関する詳細を返します。**ClientIDList** プロパティは、このプロバイダーに定義されているすべてのクライアント ID を含むコレクションです。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

出力:

ClientIDList Url	CreateDate	ThumbprintList
-----	-----	-----
---		
{MyOIDCApp}	2/3/2015 3:00:30 PM	
{12345abcdefghijk67890lmnopqrst98765uvwxyz}		oidc.example.com

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetOpenIdConnectProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **GetPolicy** を使用する

以下のコード例は、GetPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [IAM Policy Builder API を使用する](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Get information about an IAM policy.
/// </summary>
/// <param name="policyArn">The IAM policy to retrieve information for.</
param>
/// <returns>The IAM policy.</returns>
public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
{
    var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
    { PolicyArn = policyArn });
    return response.Policy;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[GetPolicy](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
```

```
const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
            "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
            policy.GetArn() << std::endl << "Description: " <<
            policy.GetDescription() << std::endl << "CreateDate: " <<
            policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[GetPolicy](#)」を参照してください。

## CLI

### AWS CLI

指定した管理ポリシーに関する情報を取得するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MySamplePolicy` である管理ポリシーに関する詳細を返します。

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

出力:


```
{
  "Policy": {
    "PolicyName": "MySamplePolicy",
    "CreateDate": "2015-06-17T19:23:32Z",
    "AttachmentCount": 0,
    "IsAttachable": true,
    "PolicyId": "Z27SI6FQMG2EXAMPLE1",
    "DefaultVersionId": "v1",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
    "UpdateDate": "2015-06-17T19:23:32Z"
  }
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetPolicy](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
// actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
  IamClient *iam.Client
}
```

```
// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
    var policy *types.Policy
    result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[GetPolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーを取得します。

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
    const command = new GetPolicyCommand({
```

```
    PolicyArn: policyArn,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetPolicy](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetPolicy](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[GetPolicy](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function getPolicy($policyArn)
{
    return $this->customWaiter(function () use ($policyArn) {
        return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
    });
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[GetPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MySamplePolicy** である管理ポリシーに関する詳細を返します。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

出力:

```
Arn          : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount : 0
CreateDate   : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : Z27SI6FQMGNQ2EXAMPLE1
PolicyName  : MySamplePolicy
UpdateDate  : 2/6/2015 10:40:08 AM
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetPolicy](#)」を参照してください。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetPolicy](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Fetches an IAM policy by its ARN
# @param policy_arn [String] the ARN of the IAM policy to retrieve
# @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
def get_policy(policy_arn)
  response = @iam_client.get_policy(policy_arn: policy_arn)
  policy = response.policy
  @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetPolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
    let input = GetPolicyInput(
        policyArn: arn
    )
    do {
        let output = try await client.getPolicy(input: input)
        guard let policy = output.policy else {
            throw ServiceHandlerError.noSuchPolicy
        }
        return policy
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[GetPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GetPolicyVersion` を使用する

以下のコード例は、`GetPolicyVersion` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)
- [IAM Policy Builder API を使用する](#)

### CLI

#### AWS CLI

指定された管理ポリシーの指定されたバージョンに関する情報を取得するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MyManagedPolicy` であるポリシーの `v2` バージョンのポリシードキュメントを返します。

```
aws iam get-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --version-id v2
```

出力:

```
{  
  "PolicyVersion": {  
    "Document": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Effect": "Allow",  
          "Action": "iam:*",  
          "Resource": "*" }  
      ]  
    },  
    "VersionId": "v2",  
    "IsDefaultVersion": true,  
    "CreateDate": "2023-04-11T00:22:54+00:00"  
  }  
}
```

```
}

```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetPolicyVersion](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:policy/MyManagedPolicy` であるポリシーの v2 バージョンのポリシードキュメントを返します。Document プロパティ内のポリシードキュメントは URL でエンコードされ、この例では `UrlDecode` .NET メソッドを使用してデコードされます。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy -VersionId v2
$results
```

出力:

```
CreateDate          Document
-----
IsDefaultVersion    VersionId
-----
2/12/2015 9:39:53 AM %7B%0A%20%20%22Version%22%3A%20%222012-10... True
                    v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Effect": "Allow",
    "Action": "*",
    "Resource": "*"
  }
}
```

```
}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetPolicyVersion](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetPolicyVersion](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetRole** を使用する

以下のコード例は、GetRole の使用方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });

    return response.Role;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[GetRole](#)」を参照してください。

## CLI

## AWS CLI

IAM ロールに関する情報を取得するには

次の `get-role` コマンドは、`Test-Role` という名前のロールに関する情報を取得します。

```
aws iam get-role \  
  --role-name Test-Role
```

出力:

```
{  
  "Role": {  
    "Description": "Test Role",  
    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",  
    "MaxSessionDuration": 3600,  
    "RoleId": "AROA1234567890EXAMPLE",  
    "CreateDate": "2019-11-13T16:45:56Z",  
    "RoleName": "Test-Role",  
    "Path": "/",  
    "RoleLastUsed": {  
      "Region": "us-east-1",  
      "LastUsedDate": "2019-11-13T17:14:00Z"  
    },  
    "Arn": "arn:aws:iam::123456789012:role/Test-Role"  
  }  
}
```

このコマンドは、ロールにアタッチされている信頼ポリシーを表示します。ロールにアタッチされているアクセス許可ポリシーを一覧表示するには、`list-role-policies` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetRole](#)」を参照してください。



## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    IamClient *iam.Client
}

// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.GetRole(context.TODO(),
        &iam.GetRoleInput{RoleName: aws.String(roleName)})
    if err != nil {
        log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[GetRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ロールを取得します。

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetRole](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function getRole($roleName)
{
    return $this->customWaiter(function () use ($roleName) {
        return $this->iamClient->getRole(['RoleName' => $roleName]);
    });
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[GetRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**lamnda\_exec\_role** の詳細を返します。これには、このロールの引き受け先を指定する信頼ポリシードキュメントが含まれています。ポリシードキュメントは URL でエンコードされており、**.NET UriDecode** メソッドを使用してデコードできます。この例では、元のポリシーは、ポリシーにアップロードされる前にすべての空白が削除されています。ロールを引き受けるユーザーが実行できる操作を決定するアクセス許可ポリシードキュメントを確認するには、インラインポリシーには **Get-IAMRolePolicy** を使用し、アタッチされた管理ポリシーには **Get-IAMPolicyVersion** を使用します。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

出力:

```
Arn : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22Service%22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%7D%5D%7D
CreateDate : 4/2/2015 9:16:11 AM
```

```
Path           : /
RoleId         : 2YBIKAIBHNKB4EXAMPLE1
RoleName       : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

出力:

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetRole](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load() # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
```

```
else:
    return role
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                                role_name: name,
                              }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[GetRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[GetRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func getRole(name: String) async throws -> IAMClientTypes.Role {
```

```
let input = GetRoleInput(  
    roleName: name  
)  
do {  
    let output = try await client.getRole(input: input)  
    guard let role = output.role else {  
        throw ServiceHandlerError.noSuchRole  
    }  
    return role  
} catch {  
    throw error  
}  
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[GetRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetRolePolicy** を使用する

以下のコード例は、GetRolePolicy の使用方法を示しています。

### CLI

#### AWS CLI

IAM ロールにアタッチされたポリシーに関する情報を取得するには

次の `get-role-policy` コマンドは、Test-Role という名前のロールにアタッチされている指定されたポリシーに関する情報を取得します。

```
aws iam get-role-policy \  
    --role-name Test-Role \  
    --policy-name ExamplePolicy
```

出力:

```
{
```

```
"RoleName": "Test-Role",
"PolicyDocument": {
  "Statement": [
    {
      "Action": [
        "s3:ListBucket",
        "s3:Put*",
        "s3:Get*",
        "s3:*MultipartUpload*"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "1"
    }
  ]
}
"PolicyName": "ExamplePolicy"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ロール `lambda_exec_role` に埋め込まれている `oneClick_lambda_exec_role_policy` という名前のポリシーのアクセス許可ポリシードキュメントを返します。結果のポリシードキュメントは URL エンコードされます。この例では、`UrlDecode` .NET メソッドを使用してデコードされています。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
oneClick_lambda_exec_role_policy
$results
```

出力:

PolicyDocument	PolicyName
UserName	



```

-----
-----
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
oneClick_lambda_exec_role_policy    lambda_exec_role

```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

出力:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3::*:*"
      ]
    }
  ]
}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetSamlProvider** を使用する

以下のコード例は、GetSamlProvider の使用方法を示しています。

### CLI

#### AWS CLI

SAML プロバイダーのメタドキュメントを取得するには

この例では、ARM が `arn:aws:iam::123456789012:saml-provider/SAMLADFS` である SAML 2.0 プロバイダーに関する詳細を取得します。レスポンスには、AWS SAML プロバイダーエンティティを作成するために ID プロバイダーから取得したメタデータドキュメントと、作成日と有効期限が含まれます。

```
aws iam get-saml-provider \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

出力:

```
{  
  "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",  
  "CreateDate": "2017-03-06T22:29:46+00:00",  
  "ValidUntil": "2117-03-06T22:29:46.433000+00:00",  
  "Tags": [  
    {  
      "Key": "DeptID",  
      "Value": "123456"  
    },  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    }  
  ]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM SAML ID プロバイダーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetSamlProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARM が `arn:aws:iam::123456789012:saml-provider/SAMLADFS` である SAML 2.0 プロバイダーに関する詳細を取得します。レスポンスには、AWS SAML プロバイダーエンティティを作成するために ID プロバイダーから取得したメタデータドキュメントと、作成日と有効期限が含まれます。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

出力:

```
CreateDate                SAMLMetadataDocument
      ValidUntil
-----
-----
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...      12/23/2114 12:16:54 PM
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetSamlProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で `GetServerCertificate` を使用する

以下のコード例は、`GetServerCertificate` の使用方法を示しています。

C++

SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
            <<
                " : " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
            << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<
            certificate.GetServerCertificateMetadata().GetServerCertificateName()
            << std::endl << "Body: " << certificate.GetCertificateBody() <<
            std::endl << "Chain: " << certificate.GetCertificateChain() <<
            std::endl;
    }

    return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[GetServerCertificate](#)」を参照してください。



```

TC01BTSBDb25zb2x1MRIwEAYDVsQQDEw1UZlXN0Q21sYWMxHzAdBgkqhkiG9w0BCQ
jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
MCMVMMxkCzAJBgNVBAGTAldBMRAwDgsYDVQHEwdTZWF0dGx1MQ8wDQYDVQKQEWZBb
WF6b24xFDASBgNVBAwTC01BTSBDb2d5zb2x1MRIwEAYDVQKQDEw1UZlXN0Q21sYWMx
HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
k60CpiwsZ3G93vUEI03IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBzZswY6786m86gjpEIbb30hjZnzcVQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUHVvXyUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0F1kbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
}
}

```

AWS アカウントで使用可能なサーバー証明書を一覧表示するには、`list-server-certificates` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM でのサーバー証明書の管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetServerCertificate](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を取得します。

```

import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**

```

```
*
* @param {string} certName
* @returns
*/
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetServerCertificate](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

```
    }  
  }  
);
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetServerCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MyServerCertificate** という名前のサーバー証明書に関する詳細を取得します。証明書の詳細は、**CertificateBody** および **ServerCertificateMetadata** プロパティで確認できます。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate  
$result | format-list
```

出力:

```
CertificateBody          : -----BEGIN CERTIFICATE-----  
  
MIICiTCCAfICCD6m7oRw0uX0jANBgqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC  
  
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6  
  
b24xFDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAd  
  
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN  
  
MTIwNDI1MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD  
  
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstC01BTSBDb25z  
  
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGft  
  
YXpvbi5jb20wZG8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ  
21uUSfwfEvySWtC2XADZ4nB  
+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
```



```

                                rDHudUZg3qX4waLG5M43q7Wgc/
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE

    Ibb30hjZnzcvcQAaRHhdLQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                                nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

    FFBjvSfpJiI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
                                NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                                -----END CERTIFICATE-----
CertificateChain      :
ServerCertificateMetadata :
    Amazon.IdentityManagement.Model.ServerCertificateMetadata

```

```
$result.ServerCertificateMetadata
```

出力:

```

Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path               : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM

```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetServerCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetServiceLastAccessedDetails** を使用する

以下のコード例は、GetServiceLastAccessedDetails の使用方法を示しています。

## CLI

### AWS CLI

サービスアクセスレポートを取得するには

次の `get-service-last-accessed-details` の例では、IAM エンティティがアクセスしたサービスを一覧表示する、以前に生成したレポートを取得します。レポートを生成するには、`generate-service-last-accessed-details` コマンドを使用します。

```
aws iam get-service-last-accessed-details \  
  --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

出力:

```
{  
  "JobStatus": "COMPLETED",  
  "JobCreationDate": "2019-10-01T03:50:35.929Z",  
  "ServicesLastAccessed": [  
    ...  
    {  
      "ServiceName": "AWS Lambda",  
      "LastAuthenticated": "2019-09-30T23:02:00Z",  
      "ServiceNamespace": "lambda",  
      "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",  
      "TotalAuthenticatedEntities": 6  
    },  
  ]  
}
```

詳細については、「AWS IAM ユーザーガイド」の [「最終アクセス情報を使用した AWS のアクセス許可の調整」](#) を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の [「GetServiceLastAccessedDetails」](#) を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、リクエスト呼び出しに関連付けられた IAM エンティティ (ユーザー、グループ、ロール、またはポリシー) が最後にアクセスしたサービスの詳細が表示されます。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

出力:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetServiceLastAccessedDetails](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetServiceLastAccessedDetailsWithEntities** を使用する

以下のコード例は、`GetServiceLastAccessedDetailsWithEntities` の使用方法を示しています。

CLI

### AWS CLI

サービスの詳細を含むサービスアクセスレポートを取得するには

次の `get-service-last-accessed-details-with-entities` の例では、指定されたサービスにアクセスした IAM ユーザーやその他のエンティティに関する詳細を含むレポートを取得します。レポートを生成するには、`generate-service-last-accessed-details` コマンドを使用します。名前空間でアクセスされるサービスのリストを取得するには、`get-service-last-accessed-details` を使用します。

```
aws iam get-service-last-accessed-details-with-entities \  
  --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \  
  --service-namespace lambda
```

出力:

```
{
  "JobStatus": "COMPLETED",
  "JobCreationDate": "2019-10-01T03:55:41.756Z",
  "JobCompletionDate": "2019-10-01T03:55:42.533Z",
  "EntityDetailsList": [
    {
      "EntityInfo": {
        "Arn": "arn:aws:iam::123456789012:user/admin",
        "Name": "admin",
        "Type": "USER",
        "Id": "AIDAI02XMPLNQEXAMPLE",
        "Path": "/"
      },
      "LastAuthenticated": "2019-09-30T23:02:00Z"
    },
    {
      "EntityInfo": {
        "Arn": "arn:aws:iam::123456789012:user/developer",
        "Name": "developer",
        "Type": "USER",
        "Id": "AIDAIBEYXMPL2YEXAMPLE",
        "Path": "/"
      },
      "LastAuthenticated": "2019-09-16T19:34:00Z"
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetServiceLastAccessedDetailsWithEntities](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、それぞれの IAM エンティティによるリクエスト内のサービスの最終アクセスタイムスタンプを提供します。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-  
dc26-ca598911cb9f -ServiceNamespace ec2  
$results
```

出力:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}  
Error             :  
IsTruncated       : False  
JobCompletionDate : 12/29/19 11:19:31 AM  
JobCreationDate   : 12/29/19 11:19:31 AM  
JobStatus         : COMPLETED  
Marker           :
```

```
$results.EntityDetailsList
```

出力:

```
EntityInfo                               LastAuthenticated  
-----                               -  
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

出力:

```
Arn   : arn:aws:iam::123456789012:user/TestUser  
Id    : AIDA4NBK5CXF5TZHU1234  
Name  : TestUser  
Path  : /  
Type  : USER
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetServiceLastAccessedDetailsWithEntities](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `GetServiceLinkedRoleDeletionStatus` を使用する

以下のコード例は、`GetServiceLinkedRoleDeletionStatus` の使用方法を示しています。

### CLI

#### AWS CLI

サービスにリンクされたロールの削除リクエストのステータスを確認するには

次の `get-service-linked-role-deletion-status` の例では、サービスにリンクされたロールを削除するという以前のリクエストのステータスが表示されます。削除オペレーションは非同期で実行されます。リクエストを実行すると、このコマンドのパラメータとして指定した `DeletionTaskId` の値を取得します。

```
aws iam get-service-linked-role-deletion-status \
  --deletion-task-id task/aws-service-role/lex.amazonaws.com/
  AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

出力:

```
{
  "Status": "SUCCEEDED"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[サービスにリンクされたロールの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetServiceLinkedRoleDeletionStatus](#)」を参照してください。

### JavaScript

#### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[GetServiceLinkedRoleDeletionStatus](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetUser** を使用する

以下のコード例は、GetUser の使用方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

/// <summary>
/// Get information about an IAM user.
/// </summary>
/// <param name="userName">The username of the user.</param>
/// <returns>An IAM user object.</returns>
public async Task<User> GetUserAsync(string userName)
{
    var response = await _IAMService.GetUserAsync(new GetUserRequest
{ Username = userName });
    return response.User;
}

```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[GetUser](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
(IAM) user already exists.

```



```
#
# Parameters:
#     $1 - The name of the IAM user to check.
#
# Returns:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_user_exists() {
    local user_name
    user_name=$1

    # Check whether the IAM user already exists.
    # We suppress all output - we're interested only in the return code.

    local errors
    errors=$(aws iam get-user \
        --user-name "$user_name" 2>&1 >/dev/null)

    local error_code=${?}

    if [[ $error_code -eq 0 ]]; then
        return 0 # 0 in Bash script means true.
    else
        if [[ $errors != *"error"*(NoSuchEntity)* ]]; then
            aws_cli_error_log $error_code
            errecho "Error calling iam get-user $errors"
        fi

        return 1 # 1 in Bash script means false.
    fi
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[GetUser](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーに関する情報を取得するには

次の `get-user` コマンドは、Paulo という名前の IAM ユーザーに関する情報を取得します。

```
aws iam get-user \  
  --user-name Paulo
```

出力:


```
{  
  "User": {  
    "UserName": "Paulo",  
    "Path": "/",  
    "CreateDate": "2019-09-21T23:03:13Z",  
    "UserId": "AIDA123456789EXAMPLE",  
    "Arn": "arn:aws:iam::123456789012:user/Paulo"  
  }  
}
```

IAM ポリシーの詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーを管理する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetUser](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.  
// It contains an IAM service client that is used to perform user actions.  
type UserWrapper struct {  
  IamClient *iam.Client  
}
```

```
// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NoSuchEntityException:
                log.Printf("User %v does not exist.\n", userName)
                err = nil
            default:
                log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
            }
        }
    } else {
        user = result.User
    }
    return user, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[GetUser](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**David** という名前のユーザーに関する詳細を取得します。

```
Get-IAMUser -UserName David
```

出力:

```
Arn           : arn:aws:iam::123456789012:user/David
CreateDate    : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
```

```
Path           : /
UserId         : Y4FKWQCXTA52QEXAMPLE1
UserName       : David
```

例 2: この例では、現在サインインしている IAM ユーザーに関する詳細を取得します。

```
Get-IAMUser
```

出力:

```
Arn           : arn:aws:iam::123456789012:user/Bob
CreateDate    : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path         : /
UserId       : 7K3GJEANSKZF2EXAMPLE2
UserName     : Bob
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetUser](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Retrieves a user's details
#
# @param user_name [String] The name of the user to retrieve
# @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
def get_user(user_name)
  response = @iam_client.get_user(user_name: user_name)
  response.user
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("User '#{user_name}' not found.")
```

```
nil
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error retrieving user '#{user_name}': #{e.message}")
  nil
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[GetUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetUserPolicy** を使用する

以下のコード例は、GetUserPolicy の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーのポリシーの詳細を一覧表示するには

次の `get-user-policy` コマンドは、Bob という名前の IAM ユーザーにアタッチされている指定されたポリシーの詳細を一覧表示します。

```
aws iam get-user-policy \
  --user-name Bob \
  --policy-name ExamplePolicy
```

出力:

```
{
  "UserName": "Bob",
  "PolicyName": "ExamplePolicy",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "*",
        "Resource": "*",
```

```
        "Effect": "Allow"
      }
    ]
  }
}
```

IAM ユーザーのポリシーのリストを取得するには、`list-user-policies` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetUserPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**David** という名前の IAM ユーザーに埋め込まれている **Dauids\_IAM\_Admin\_Policy** という名前のインラインポリシーの詳細を取得します。ポリシードキュメントは URL エンコードされています。

```
$results = Get-IAMUserPolicy -PolicyName Dauids_IAM_Admin_Policy -UserName David
$results
```

出力:

```
PolicyDocument                                     PolicyName
-----
-----
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...  Dauids_IAM_Admin_Policy
David
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
    "Action": [
      "iam:*"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetUserPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListAccessKeys** を使用する

以下のコード例は、ListAccessKeys の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスキーの管理](#)

### Bash

Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#     -u user_name -- The name of the IAM user.
#
# Returns:
#     access_key_ids
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_list_access_keys() {

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_list_access_keys"
        echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
the specified user."
        echo " -u user_name    The name of the IAM user."
        echo ""
    }

    local user_name response
    local option OPTARG # Required to use getopt command in a function.
    # Retrieve the calling parameters.
    while getopt "u:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
        esac
    done
}
```



```
;;
esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[ListAccessKeys](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                const Aws::Client::ClientConfiguration
                                &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                    key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                statusString << std::setw(20) <<
                key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
}
```

```
    }  
  }  
  
  return true;  
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーのアクセスキー ID を一覧表示するには

次の `list-access-keys` コマンドは、Bob という名前の IAM ユーザーのアクセスキー ID を一覧表示します。

```
aws iam list-access-keys \  
  --user-name Bob
```

出力:

```
{  
  "AccessKeyMetadata": [  
    {  
      "UserName": "Bob",  
      "Status": "Active",  
      "CreateDate": "2013-06-04T18:17:34Z",  
      "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"  
    },  
    {  
      "UserName": "Bob",  
      "Status": "Inactive",  
      "CreateDate": "2013-06-06T20:42:26Z",  
      "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"  
    }  
  ]  
}
```


IAM ユーザーのシークレットアクセスキーを一覧表示することはできません。シークレットアクセスキーを紛失した場合は、`create-access-keys` コマンドを使用して新しいアクセスキーを作成する必要があります。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListAccessKeys](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
([]types.AccessKeyMetadata, error) {
    var keys []types.AccessKeyMetadata
    result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
&iam.ListAccessKeysInput{
    Username: aws.String(userName),
})
    if err != nil {
        log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
err)
    } else {
        keys = result.AccessKeyMetadata
    }
}
```

```
}  
return keys, err  
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;  
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListAccessKeys {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <userName>\s
```

```
        Where:
            userName - The name of the user for which access keys are
retrieved.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    listKeys(iam, userName);
    System.out.println("Done");
    iam.close();
}

public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();
```

```
        response = iam.listAccessKeys(request);
    }

    for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.accessKeyId());
    }

    if (!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アクセスキーを一覧表示します。

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    Username: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
undefined}
   */
  let response = await client.send(command);


  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccessKeysCommand({
          Marker: response.Marker,
        })),
    );
  } else {
    break;
  }
}
}
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListAccessKeys](#)」を参照してください。



## SDK for JavaScript (v2)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、**Bob** という名前の IAM ユーザーのアクセスキーを一覧表示します。IAM ユーザーのシークレットアクセスキーを一覧表示できないことに注意してください。シークレットアクセスキーを紛失した場合は、**New-IAMAccessKey** コマンドレットを使用して新しいアクセスキーを作成する必要があります。

```
Get-IAMAccessKey -UserName "Bob"
```

出力:

AccessKeyId	CreateDate	Status	
-----	-----	-----	
-----			
AKIAIOSFODNN7EXAMPLE	12/3/2014 10:53:41 AM	Active	Bob
AKIAI44QH8DHBEXAMPLE	6/6/2013 8:42:26 PM	Inactive	Bob

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListAccessKeys](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListAccessKeys](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end
end
```

```
# Creates an access key for a user
#
# @param user_name [String] The name of the user.
# @return [Boolean]
def create_access_key(user_name)
  response = @iam_client.create_access_key(user_name: user_name)
  access_key = response.access_key
  @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
  access_key
rescue Aws::IAM::Errors::LimitExceeded => e
  @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
  nil
rescue StandardError => e
  @logger.error("Error creating access key: #{e.message}")
  nil
end

# Deactivates an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def deactivate_access_key(user_name, access_key_id)
  @iam_client.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,
    status: "Inactive"
  )
  true
rescue StandardError => e
  @logger.error("Error deactivating access key: #{e.message}")
  false
end

# Deletes an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def delete_access_key(user_name, access_key_id)
  @iam_client.delete_access_key(
```

```
        user_name: user_name,
        access_key_id: access_key_id
    )
    true
rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAccessKeys](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListAccountAliases** を使用する

以下のコード例は、ListAccountAliases の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アカウントの管理](#)

C++

SDK for C++

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
    &clientConfig) {
```

```
Aws::IAM::IAMClient iam(clientConfig);
Aws::IAM::Model::ListAccountAliasesRequest request;

bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListAccountAliases(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list account aliases: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const auto &aliases = outcome.GetResult().GetAccountAliases();
    if (!header) {
        if (aliases.size() == 0) {
            std::cout << "Account has no aliases" << std::endl;
            break;
        }
        std::cout << std::left << std::setw(32) << "Alias" << std::endl;
        header = true;
    }

    for (const auto &alias: aliases) {
        std::cout << std::left << std::setw(32) << alias << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[ListAccountAliases](#)」を参照してください。

## CLI

### AWS CLI

アカウントエイリアスを一覧表示するには

次の `list-account-aliases` コマンドは、現在のアカウントのエイリアスを一覧表示します。

```
aws iam list-account-aliases
```

出力:

```
{
  "AccountAliases": [
    "mycompany"
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウント ID とそのエイリアス](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListAccountAliases](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```



```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListAccountAliases](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示します。

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

```
}
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListAccountAliases](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListAccountAliases](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[ListAccountAliases](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、AWS アカウントのアカウントエイリアスを返します。

```
Get-IAMAccountAlias
```

出力:

```
ExampleCo
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListAccountAliases](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListAccountAliases](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  end
end
```

```
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating account alias: #{e.message}")
  false
end

# Deletes an AWS account alias.
#
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
def delete_account_alias(account_alias)
  @iam_client.delete_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAccountAliases](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListAttachedGroupPolicies** を使用する

以下のコード例は、ListAttachedGroupPolicies の使用方法を示しています。

### CLI

#### AWS CLI

指定されたグループにアタッチされているすべての管理ポリシーを一覧表示するには

この例では、AWS アカウントの Admins という名前の IAM グループにアタッチされている管理ポリシーの名前と ARN を返します。

```
aws iam list-attached-group-policies \
  --group-name Admins
```

出力:

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    {
      "PolicyName": "SecurityAudit",
      "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
    }
  ],
  "IsTruncated": false
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListAttachedGroupPolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、AWS アカウントの **Admins** という名前の IAM グループにアタッチされている管理ポリシーの名前と ARN を返します。グループに埋め込まれているインラインポリシーのリストを表示するには、**Get-IAMGroupPolicyList** コマンドを使用します。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

出力:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess	AdministratorAccess

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListAttachedGroupPolicies](#)」を参照してください。



AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListAttachedRolePolicies` を使用する

以下のコード例は、`ListAttachedRolePolicies` の使用方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List the IAM role policies that are attached to an IAM role.
/// </summary>
/// <param name="roleName">The IAM role to list IAM policies for.</param>
/// <returns>A list of the IAM policies attached to the IAM role.</returns>
public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
{
    var attachedPolicies = new List<AttachedPolicyType>();
    var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

    await foreach (var response in attachedRolePoliciesPaginator.Responses)
    {
        attachedPolicies.AddRange(response.AttachedPolicies);
    }

    return attachedPolicies;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## CLI

### AWS CLI

指定された IAM ロールにアタッチされている管理ポリシーを一覧表示するには

このコマンドは、AWS アカウントの SecurityAuditRole という名前の IAM ロールにアタッチされている管理ポリシーの名前と ARN を返します。

```
aws iam list-attached-role-policies \  
  --role-name SecurityAuditRole
```

出力:


```
{  
  "AttachedPolicies": [  
    {  
      "PolicyName": "SecurityAudit",  
      "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"  
    }  
  ],  
  "IsTruncated": false  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// ListAttachedRolePolicies lists the policies that are attached to the specified
// role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
([]types.AttachedPolicy, error) {
    var policies []types.AttachedPolicy
    result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
&iam.ListAttachedRolePoliciesInput{
    RoleName: aws.String(roleName),
})
    if err != nil {
        log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
roleName, err)
    } else {
        policies = result.AttachedPolicies
    }
    return policies, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ロールにアタッチされたポリシーを一覧表示します。

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
        })
      );
    }
  }
}
```

```
    }),  
    );  
  } else {  
    break;  
  }  
}  
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();  
$service = new IAMService();  
  
public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker  
= "", $maxItems = 0)  
{  
    $listAttachRolePoliciesArguments = ['RoleName' => $roleName];  
    if ($pathPrefix) {  
        $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;  
    }  
    if ($marker) {  
        $listAttachRolePoliciesArguments['Marker'] = $marker;  
    }  
    if ($maxItems) {  
        $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;  
    }  
    return $this->iamClient->  
listAttachedRolePolicies($listAttachRolePoliciesArguments);  
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、AWS アカウントの **SecurityAuditRole** という名前の IAM ロールにアタッチされている管理ポリシーの名前と ARN を返します。ロールに埋め込まれているインラインポリシーのリストを表示するには、**Get-IAMRolePolicyList** コマンドを使用します。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

出力:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_attached_policies(role_name):  
    """  
    Lists policies attached to a role.
```

```
:param role_name: The name of the role to query.
"""
try:
    role = iam.Role(role_name)
    for policy in role.attached_policies.all():
        logger.info("Got policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't list attached policies for %s.", role_name)
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
```

```
#
# @param policy_name [String] The name of the policy
# @param policy_document [Hash] The policy document
# @return [String] The policy ARN if successful, otherwise nil
def create_policy(policy_name, policy_document)
  response = @iam_client.create_policy(
    policy_name: policy_name,
    policy_document: policy_document.to_json
  )
  response.policy.arn
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating policy: #{e.message}")
  nil
end

# Fetches an IAM policy by its ARN
# @param policy_arn [String] the ARN of the IAM policy to retrieve
# @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
def get_policy(policy_arn)
  response = @iam_client.get_policy(policy_arn: policy_arn)
  policy = response.policy
  @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

# Attaches a policy to a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_role(role_name, policy_arn)
  @iam_client.attach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
end
```



```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// Returns a list of AWS Identity and Access Management (IAM) policies
/// that are attached to the role.
///
/// - Parameter role: The IAM role to return the policy list for.
///
/// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
///   describing each managed policy that's attached to the role.
public func listAttachedRolePolicies(role: String) async throws ->
[IAMClientTypes.AttachedPolicy] {
    var policyList: [IAMClientTypes.AttachedPolicy] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListAttachedRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listAttachedRolePolicies(input: input)

        guard let attachedPolicies = output.attachedPolicies else {
            return policyList
        }
    }
```

```
        for attachedPolicy in attachedPolicies {
            policyList.append(attachedPolicy)
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListAttachedUserPolicies** を使用する

以下のコード例は、ListAttachedUserPolicies の使用方法を示しています。

CLI

### AWS CLI

指定されたユーザーにアタッチされているすべての管理ポリシーを一覧表示するには

このコマンドは、AWS アカウントの Bob という名前の IAM ユーザーの管理ポリシーの名前と ARN を返します。

```
aws iam list-attached-user-policies \
    --user-name Bob
```

出力:

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
  ],
}
```

```
{
  "PolicyName": "SecurityAudit",
  "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
},
"IsTruncated": false
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListAttachedUserPolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: このコマンドは、AWS アカウントの **Bob** という名前の IAM ユーザーの管理ポリシーの名前と ARN を返します。IAM ユーザーに埋め込まれているインラインポリシーのリストを表示するには、**Get-IAMUserPolicyList** コマンドを使用します。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

出力:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/TesterPolicy	TesterPolicy

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListAttachedUserPolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListEntitiesForPolicy** を使用する

以下のコード例は、ListEntitiesForPolicy の使用方法を示しています。

## CLI

## AWS CLI

指定された管理ポリシーがアタッチされているすべてのユーザー、グループ、ロールを一覧表示するには

この例では、ポリシー `arn:aws:iam::123456789012:policy/TestPolicy` がアタッチされている IAM グループ、ロール、ユーザーのリストを返します。

```
aws iam list-entities-for-policy \  
  --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

出力:

```
{  
  "PolicyGroups": [  
    {  
      "GroupName": "Admins",  
      "GroupId": "AGPACKCEVSQ6C2EXAMPLE"  
    }  
  ],  
  "PolicyUsers": [  
    {  
      "UserName": "Alice",  
      "UserId": "AIDACKCEVSQ6C2EXAMPLE"  
    }  
  ],  
  "PolicyRoles": [  
    {  
      "RoleName": "DevRole",  
      "RoleId": "AROADBQP57FF2AEXAMPLE"  
    }  
  ],  
  "IsTruncated": false  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListEntitiesForPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ポリシー `arn:aws:iam::123456789012:policy/TestPolicy` がアタッチされている IAM グループ、ロール、ユーザーのリストを返します。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

出力:

```
IsTruncated   : False
Marker        :
PolicyGroups  : {}
PolicyRoles   : {testRole}
PolicyUsers   : {Bob, Theresa}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListEntitiesForPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で `ListGroupPolicies` を使用する

以下のコード例は、`ListGroupPolicies` の使用方法を示しています。

#### CLI

##### AWS CLI

指定されたグループにアタッチされているすべてのインラインポリシーを一覧表示するには

次の `list-group-policies` コマンドは、現在のアカウントの `Admins` という名前の IAM グループにアタッチされているインラインポリシーの名を一覧表示します。

```
aws iam list-group-policies \  
  --group-name Admins
```

出力:

```
{
  "PolicyNames": [
    "AdminRoot",
    "ExamplePolicy"
  ]
}
```

IAM ポリシーの詳細については、「AWS IAM ユーザーガイド」の「[IAM ポリシーを管理する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListGroupPolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、グループ **Testers** に埋め込まれているインラインポリシーのリストを返します。グループにアタッチされている管理ポリシーを取得するには、コマンド **Get-IAMAttachedGroupPolicyList** を使用します。

```
Get-IAMGroupPolicyList -GroupName Testers
```

出力:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListGroupPolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListGroups** を使用する

以下のコード例は、ListGroups の使用方法を示しています。



## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List IAM groups.
/// </summary>
/// <returns>A list of IAM groups.</returns>
public async Task<List<Group>> ListGroupsAsync()
{
    var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
    var groups = new List<Group>();

    await foreach (var response in groupsPaginator.Responses)
    {
        groups.AddRange(response.Groups);
    }

    return groups;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListGroup](#)」を参照してください。

## CLI

### AWS CLI

現在のアカウントの IAM グループを一覧表示するには

次の `list-groups` コマンドは、現在のアカウントの IAM グループを一覧表示します。

```
aws iam list-groups
```

出力:

```
{
  "Groups": [
    {
      "Path": "/",
      "CreateDate": "2013-06-04T20:27:27.972Z",
      "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/Admins",
      "GroupName": "Admins"
    },
    {
      "Path": "/",
      "CreateDate": "2013-04-16T20:30:42Z",
      "GroupId": "AIDGPMS9R04H3FEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
      "GroupName": "S3-Admins"
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListGroup](#)s」を参照してください。

Go

SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
    iamClient *iam.Client
}

// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(maxGroups int32) ([]types.Group, error) {
    var groups []types.Group
    result, err := wrapper.IamClient.ListGroups(context.TODO(),
        &iam.ListGroupsInput{
            MaxItems: aws.Int32(maxGroups),
        })
    if err != nil {
        log.Printf("Couldn't list groups. Here's why: %v\n", err)
    } else {
        groups = result.Groups
    }
    return groups, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListGroup](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

グループを一覧表示します。

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListGroup](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listGroupsArguments = [];
    if ($pathPrefix) {
        $listGroupsArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listGroupsArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listGroupsArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listGroups($listGroupsArguments);
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントで定義されているすべての IAM グループのコレクションを返します。

## Get-IAMGroupList

出力:

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
GroupName : Administrators
Path      : /

Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZ0EXAMPLE2
GroupName : Developers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE3
GroupName : Testers
Path      : /
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListGroups](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
```

```
"""
try:
    for group in iam.groups.limit(count):
        logger.info("Group: %s", group.name)
except ClientError:
    logger.exception("Couldn't list groups for the account.")
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListGroups](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
  end
end
```

```
response
rescue Aws::Errors::ServiceError => e
  @logger.error("Couldn't list groups for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListGroup](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_groups(
  client: &iamClient,
  path_prefix: Option<String>,
  marker: Option<String>,
  max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
  let response = client
    .list_groups()
    .set_path_prefix(path_prefix)
    .set_marker(marker)
    .set_max_items(max_items)
    .send()
    .await?;

  Ok(response)
}
```



- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListGroup](#)s」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func listGroups() async throws -> [String] {
    var groupList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListGroupsInput(marker: marker)
        let output = try await client.listGroups(input: input)

        guard let groups = output.groups else {
            return groupList
        }

        for group in groups {
            if let name = group.groupName {
                groupList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
}
```

```
    return groupList
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListGroups](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListGroupsForUser** を使用する

以下のコード例は、ListGroupsForUser の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーが属する IAM グループを一覧表示するには

次の list-groups-for-user コマンドは、Bob という IAM ユーザーが属するグループを表示します。

```
aws iam list-groups-for-user \  
  --user-name Bob
```

出力:

```
{  
  "Groups": [  
    {  
      "Path": "/",  
      "CreateDate": "2013-05-06T01:18:08Z",  
      "GroupId": "AKIAIOSFODNN7EXAMPLE",  
      "Arn": "arn:aws:iam::123456789012:group/Admin",  
      "GroupName": "Admin"  
    },  
    {  
      "Path": "/",  
      "CreateDate": "2013-05-06T01:37:28Z",  
      "GroupId": "AKIAI44QH8DHBEXAMPLE",
```

```
        "Arn": "arn:aws:iam::123456789012:group/s3-Users",
        "GroupName": "s3-Users"
    }
]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListGroupForUser](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **David** が属する IAM グループのリストを返します。

```
Get-IAMGroupForUser -UserName David
```

出力:

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
GroupName : Administrators
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE2
GroupName : Testers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZOEXAMPLE3
GroupName : Developers
Path      : /
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListGroupForUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListInstanceProfiles` を使用する

以下のコード例は、`ListInstanceProfiles` の使用方法を示しています。

### CLI

#### AWS CLI

アカウントのインスタンスプロファイルを一覧表示するには

次の `list-instance-profiles` コマンドは、現在のアカウントに関連付けられているインスタンスプロファイルを一覧表示します。

```
aws iam list-instance-profiles
```

出力:

```
{
  "InstanceProfiles": [
    {
      "Path": "/",
      "InstanceProfileName": "example-dev-role",
      "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
      "CreateDate": "2023-09-21T18:17:41+00:00",
      "Roles": [
        {
          "Path": "/",
          "RoleName": "example-dev-role",
          "RoleId": "AROAJ520TH4H7LEXAMPLE",
          "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
          "CreateDate": "2023-09-21T18:17:40+00:00",
          "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Principal": {
                  "Service": "ec2.amazonaws.com"
                }
              }
            ]
          }
        }
      ]
    }
  ]
}
```

```
    },
    "Action": "sts:AssumeRole"
  }
]
}
},
{
  "Path": "/",
  "InstanceProfileName": "example-s3-role",
  "InstanceProfileId": "AIPAJVJVNRIQFEXAMPLE",
  "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
  "CreateDate": "2023-09-21T18:18:50+00:00",
  "Roles": [
    {
      "Path": "/",
      "RoleName": "example-s3-role",
      "RoleId": "AROAINUBC507XLEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
      "CreateDate": "2023-09-21T18:18:49+00:00",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListInstanceProfiles](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントで定義されているインスタンスプロファイルのコレクションを返します。

```
Get-IAMInstanceProfileList
```

出力:

```
Arn          : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate   : 2/17/2015 2:49:04 PM
InstanceId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles        : {ec2instancerole}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListInstanceProfiles](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で `ListInstanceProfilesForRole` を使用する

以下のコード例は、`ListInstanceProfilesForRole` の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ロールのインスタンスプロファイルを一覧表示するには

次の `list-instance-profiles-for-role` コマンドは、ロール `Test-Role` に関連付けられているインスタンスプロファイルを一覧表示します。

```
aws iam list-instance-profiles-for-role \  
  --role-name Test-Role
```

出力:

```
{
  "InstanceProfiles": [
    {
      "InstanceId": "AIDGPMS9R04H3FEXAMPLE",
      "Roles": [
        {
          "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
          "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
          "CreateDate": "2013-06-07T20:42:15Z",
          "RoleName": "Test-Role",
          "Path": "/",
          "Arn": "arn:aws:iam::123456789012:role/Test-Role"
        }
      ],
      "CreateDate": "2013-06-07T21:05:24Z",
      "InstanceProfileName": "ExampleInstanceProfile",
      "Path": "/",
      "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListInstanceProfilesForRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ロール **ec2instancerole** に関連付けられているインスタンスプロファイルの詳細を返します。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

出力:

```
Arn           : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceProfileId : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles        : {ec2instancerole}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListInstanceProfilesForRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListMfaDevices** を使用する

以下のコード例は、ListMfaDevices の使用方法を示しています。

### CLI

#### AWS CLI

指定したユーザーのすべての MFA デバイスを一覧表示するには

この例では、IAM ユーザー Bob に割り当てられた MFA デバイスに関する詳細を返します。

```
aws iam list-mfa-devices \
  --user-name Bob
```

出力:

```
{
  "MFADevices": [
    {
      "UserName": "Bob",
      "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
      "EnableDate": "2019-10-28T20:37:09+00:00"
    },
    {
      "UserName": "Bob",
```



```

        "SerialNumber": "GAKT12345678",
        "EnableDate": "2023-02-18T21:44:42+00:00"
    },
    {
        "UserName": "Bob",
        "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
        "EnableDate": "2023-09-19T02:25:35+00:00"
    },
    {
        "UserName": "Bob",
        "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
        "EnableDate": "2023-09-19T01:49:18+00:00"
    }
]
}

```

詳細については、「AWS IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListMfaDevices](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **David** に割り当てられた MFA デバイスに関する詳細を返します。この例では、**SerialNumber** は物理デバイスの実際のシリアル番号ではなく ARN であるため、仮想デバイスであることがわかります。

```
Get-IAMMFADevice -UserName David
```

出力:

EnableDate	SerialNumber	UserName
-----	-----	-----
4/8/2015 9:41:10 AM	arn:aws:iam::123456789012:mfa/David	David

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListMfaDevices](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListOpenIdConnectProviders` を使用する

以下のコード例は、`ListOpenIdConnectProviders` の使用方法を示しています。

### CLI

#### AWS CLI

AWS アカウント内の OpenID Connect プロバイダーに関する情報を一覧表示するには

この例では、現在の AWS アカウントで定義されているすべての OpenID Connect プロバイダーの ARN のリストを返します。

```
aws iam list-open-id-connect-providers
```

出力:

```
{
  "OpenIDConnectProviderList": [
    {
      "Arn": "arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com"
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListOpenIdConnectProviders](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、現在の AWS アカウントで定義されているすべての OpenID Connect プロバイダーの ARN のリストを返します。

```
Get-IAMOpenIDConnectProviderList
```

出力:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListOpenIdConnectProviders](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListPolicies** を使用する

以下のコード例は、ListPolicies の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List IAM policies.
/// </summary>
/// <returns>A list of the IAM policies.</returns>
```

```
public async Task<List<ManagedPolicy>> ListPoliciesAsync()
{
    var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
    var policies = new List<ManagedPolicy>();

    await foreach (var response in listPoliciesPaginator.Responses)
    {
        policies.AddRange(response.Policies);
    }

    return policies;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[ListPolicies](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
&clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (!header) {
    std::cout << std::left << std::setw(55) << "Name" <<
        std::setw(30) << "ID" << std::setw(80) << "Arn" <<
        std::setw(64) << "Description" << std::setw(12) <<
        "CreateDate" << std::endl;
    header = true;
}

const auto &policies = outcome.GetResult().GetPolicies();
for (const auto &policy: policies) {
    std::cout << std::left << std::setw(55) <<
        policy.GetPolicyName() << std::setw(30) <<
        policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
        std::setw(64) << policy.GetDescription() << std::setw(12)
<<
        policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
        std::endl;
}

if (outcome.GetResult().GetIsTruncated()) {
    request.SetMarker(outcome.GetResult().GetMarker());
}
else {
    done = true;
}
}

return true;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[ListPolicies](#)」を参照してください。

## CLI

### AWS CLI

AWS アカウントで使用できる管理ポリシーを一覧表示するには

この例では、現在の AWS アカウントで使用可能な最初の 2 つの管理ポリシーのコレクションを返します。

```
aws iam list-policies \  
  --max-items 3
```

出力:

```
{  
  "Policies": [  
    {  
      "PolicyName": "AWSCloudTrailAccessPolicy",  
      "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",  
      "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",  
      "Path": "/",  
      "DefaultVersionId": "v1",  
      "AttachmentCount": 0,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
      "CreateDate": "2019-09-04T17:43:42+00:00",  
      "UpdateDate": "2019-09-04T17:43:42+00:00"  
    },  
    {  
      "PolicyName": "AdministratorAccess",  
      "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",  
      "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",  
      "Path": "/",  
      "DefaultVersionId": "v1",  
      "AttachmentCount": 6,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
      "CreateDate": "2015-02-06T18:39:46+00:00",  
      "UpdateDate": "2015-02-06T18:39:46+00:00"  
    },  
    {  
      "PolicyName": "PowerUserAccess",  
      "PolicyId": "ANPAJYRXTHIB4F0VS3ZXS",  
      "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",  
      "Path": "/",  
      "DefaultVersionId": "v5",  
      "AttachmentCount": 1,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
    }  
  ]  
}
```

```
        "CreateDate": "2015-02-06T18:39:47+00:00",
        "UpdateDate": "2023-07-06T22:04:00+00:00"
    }
],
"NextToken": "EXAMPLErZXIi0iBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQi0iA4fQ=="
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListPolicies](#)」を参照してください。

Go

SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    iamClient *iam.Client
}

// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
error) {
    var policies []types.Policy
    result, err := wrapper.IamClient.ListPolicies(context.TODO(),
&iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPolicies),
    })
```

```
if err != nil {
    log.Printf("Couldn't list policies. Here's why: %v\n", err)
} else {
    policies = result.Policies
}
return policies, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListPolicies](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーを一覧表示します。

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listPolicies() {
    const command = new ListPoliciesCommand({
        MaxItems: 10,
        OnlyAttached: false,
        // List only the customer managed policies in your Amazon Web Services
        account.
    });
```



```
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        })),
    );
  } else {
    break;
  }
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListPolicies](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();
```

```
public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listPoliciesArguments = [];
    if ($pathPrefix) {
        $listPoliciesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listPoliciesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listPoliciesArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listPolicies($listPoliciesArguments);
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[ListPolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントで使用可能な最初の 3 つの管理ポリシーのコレクションを返します。**-scope** が指定されていないため、デフォルトは **all** で、AWS 管理ポリシーとカスタマー管理ポリシーの両方が含まれます。

```
Get-IAMPolicyList -MaxItem 3
```

出力:

```
Arn          : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount : 0
CreateDate   : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : Z27SI6FQMGNO2EXAMPLE1
PolicyName  : AWSDirectConnectReadOnlyAccess
```

```
UpdateDate      : 2/6/2015 10:40:08 AM

Arn             : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount : 0
CreateDate      : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description     :
IsAttachable   : True
Path           : /
PolicyId       : NJKMU274MET4EEXAMPLE2
PolicyName     : AmazonGlacierReadOnlyAccess
UpdateDate     : 2/6/2015 10:40:27 AM

Arn             : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount : 0
CreateDate      : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description     :
IsAttachable   : True
Path           : /
PolicyId       : 5ULJS02FYVPYGEXAMPLE3
PolicyName     : AWSMarketplaceFullAccess
UpdateDate     : 2/11/2015 9:21:45 AM
```

例 2: この例では、現在の AWS アカウントで使用可能な最初の 2 つのカスタマー管理ポリシーのコレクションを返します。-**Scope local** を使用して、出力をカスタマー管理ポリシーのみに制限します。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

出力:

```
Arn             : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount : 0
CreateDate      : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description     :
IsAttachable   : True
Path           : /
PolicyId       : SQVCBLC4VA0UCEXAMPLE4
PolicyName     : MyLocalPolicy
UpdateDate     : 2/12/2015 9:39:53 AM
```

```
Arn          : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount : 1
CreateDate    : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description   :
IsAttachable  : True
Path          : /
PolicyId      : X5JPBLJH2Z2S0EXAMPLE5
PolicyName    : policyforec2instancerole
UpdateDate    : 2/18/2015 8:52:31 AM
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListPolicies](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
    returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListPolicies](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
  end
end
```

```
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
```

```
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end

# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
# @return [Boolean] true if successful, false otherwise
def detach_policy_from_role(role_name, policy_arn)
  @iam_client.detach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error detaching policy from role: #{e.message}")
  false
end
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListPolicies](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_policies(
```

```
client: iamClient,
path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[ListPolicies](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。



**Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func listPolicies() async throws -> [MyPolicyRecord] {
    var policyList: [MyPolicyRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListPoliciesInput(marker: marker)
        let output = try await client.listPolicies(input: input)

        guard let policies = output.policies else {
            return policyList
        }

        for policy in policies {
            guard let name = policy.policyName,
                  let id = policy.policyId,
                  let arn = policy.arn else {
                throw ServiceHandlerError.noSuchPolicy
            }
            policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListPolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListPolicyVersions` を使用する

以下のコード例は、`ListPolicyVersions` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)
- [ポリシーを以前のバージョンにロールバックする](#)

### CLI

#### AWS CLI

指定された管理ポリシーのバージョンに関する情報を一覧表示するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MySamplePolicy` であるポリシーの使用可能なバージョンのリストを返します。

```
aws iam list-policy-versions \  
  --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

出力:

```
{  
  "IsTruncated": false,  
  "Versions": [  
    {  
      "VersionId": "v2",  
      "IsDefaultVersion": true,  
      "CreateDate": "2015-06-02T23:19:44Z"  
    },  
    {  
      "VersionId": "v1",  
      "IsDefaultVersion": false,  
      "CreateDate": "2015-06-02T22:30:47Z"  
    }  
  ]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListPolicyVersions](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:policy/MyManagedPolicy` であるポリシーの使用可能なバージョンのリストを返します。特定のバージョンのポリシードキュメントを取得するには、`Get-IAMPolicyVersion` コマンドを使用して、必要なバージョンの `VersionId` を指定します。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

出力:

CreateDate	Document	IsDefaultVersion
VersionId		
-----	-----	-----
-----		
2/12/2015 9:39:53 AM		True
v2		
2/12/2015 9:39:09 AM		False
v1		

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListPolicyVersions](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListRolePolicies` を使用する

以下のコード例は、`ListRolePolicies` の使用方法を示しています。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List IAM role policies.
/// </summary>
/// <param name="roleName">The IAM role for which to list IAM policies.</
param>
/// <returns>A list of IAM policy names.</returns>
public async Task<List<string>> ListRolePoliciesAsync(string roleName)
{
    var listRolePoliciesPaginator =
_IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
roleName });
    var policyNames = new List<string>();

    await foreach (var response in listRolePoliciesPaginator.Responses)
    {
        policyNames.AddRange(response.PolicyNames);
    }

    return policyNames;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## CLI

### AWS CLI

IAM ロールにアタッチされているポリシーを一覧表示するには

次の `list-role-policies` コマンドは、指定された IAM ロールのアクセス許可ポリシーの名前を一覧表示します。

```
aws iam list-role-policies \  
  --role-name Test-Role
```

出力:

```
{  
  "PolicyNames": [  
    "ExamplePolicy"  
  ]  
}
```


ロールにアタッチされている信頼ポリシーを表示するには、`get-role` コマンドを使用します。アクセス許可ポリシーの詳細を表示するには、`get-role-policy` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListRolePolicies](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {
```

```
IamClient *iam.Client
}

// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
        &iam.ListRolePoliciesInput{
            RoleName: aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
            err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

ポリシーを一覧表示します。

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        })),
    );
  } else {
    break;
  }
}
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
{
    $listRolePoliciesArguments = ['RoleName' => $roleName];
    if ($marker) {
        $listRolePoliciesArguments['Marker'] = $marker;
    }
    if ($maxItems) {
        $listRolePoliciesArguments['MaxItems'] = $maxItems;
    }
    return $this->customWaiter(function () use ($listRolePoliciesArguments) {
        return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
    });
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ロール **lamda\_exec\_role** に埋め込まれているインラインポリシーの名前のリストを返します。インラインポリシーの詳細を表示するには、**Get-IAMRolePolicy** コマンドを使用します。



```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

出力:

```
oneClick_lambda_exec_role_policy
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListRolePolicies](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) APIリファレンス」の「[ListRolePolicies](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_role_policies(
  client: &iamClient,
  role_name: &str,
  marker: Option<String>,
```

```
max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
  let response = client
    .list_role_policies()
    .role_name(role_name)
    .set_marker(marker)
    .set_max_items(max_items)
    .send()
    .await?;

  Ok(response)
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[ListRolePolicies](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func listRolePolicies(role: String) async throws -> [String] {
  var policyList: [String] = []
  var marker: String? = nil
  var isTruncated: Bool

  repeat {
    let input = ListRolePoliciesInput(
```

```
        marker: marker,
        roleName: role
    )
    let output = try await client.listRolePolicies(input: input)

    guard let policies = output.policyNames else {
        return policyList
    }

    for policy in policies {
        policyList.append(policy)
    }
    marker = output.marker
    isTruncated = output.isTruncated
} while isTruncated == true
return policyList
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[ListRolePolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListRoleTags** を使用する

以下のコード例は、ListRoleTags の使用方法を示しています。

### CLI

#### AWS CLI

ロールにアタッチされたタグを一覧表示するには

次の list-role-tags コマンドは、指定されたロールに関連付けられているタグのリストを取得します。

```
aws iam list-role-tags \  
    --role-name production-role
```

出力:

```
{
  "Tags": [
    {
      "Key": "Department",
      "Value": "Accounting"
    },
    {
      "Key": "DeptID",
      "Value": "12345"
    }
  ],
  "IsTruncated": false
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListRoleTags](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ロールに関連付けられているタグを取得します。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListRoleTags](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListRoles** を使用する

以下のコード例は、ListRoles の使用方法を示しています。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List IAM roles.
/// </summary>
/// <returns>A list of IAM roles.</returns>
public async Task<List<Role>> ListRolesAsync()
{
    var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
    var roles = new List<Role>();

    await foreach (var response in listRolesPaginator.Responses)
    {
        roles.AddRange(response.Roles);
    }

    return roles;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListRoles](#)」を参照してください。

## CLI

### AWS CLI

現在のアカウントの IAM ロールを一覧表示するには

次の `list-roles` コマンドは、現在のアカウントの IAM ロールを一覧表示します。

```
aws iam list-roles
```

出力:

```
{
  "Roles": [
    {
      "Path": "/",
      "RoleName": "ExampleRole",
      "RoleId": "AROAJ520TH4H7LEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
      "CreateDate": "2017-09-12T19:23:36+00:00",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "MaxSessionDuration": 3600
    },
    {
      "Path": "/example_path/",
      "RoleName": "ExampleRoleWithPath",
      "RoleId": "AROAI4QRP7UFT7EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
      "CreateDate": "2023-09-21T20:29:38+00:00",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "MaxSessionDuration": 3600
    }
  ]
}
```

```
        "Action": "sts:AssumeRole"
    }
}
],
},
"MaxSessionDuration": 3600
}
]
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListRoles](#)」を参照してください。

Go

SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    IamClient *iam.Client
}

// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
    var roles []types.Role
    result, err := wrapper.IamClient.ListRoles(context.TODO(),
        &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
    )
    if err != nil {
```



```
    log.Printf("Couldn't list roles. Here's why: %v\n", err)
  } else {
    roles = result.Roles
  }
  return roles, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListRoles](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ロールを一覧表示します。

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}

```

```
*/
let response = await client.send(command);

while (response?.Roles?.length) {
  for (const role of response.Roles) {
    yield role;
  }

  if (response.IsTruncated) {
    response = await client.send(
      new ListRolesCommand({
        Marker: response.Marker,
      })),
  );
} else {
  break;
}
}
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListRoles](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

/**
 * @param string $pathPrefix
 * @param string $marker
 * @param int $maxItems
```

```
* @return Result
* $roles = $service->listRoles();
*/
public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listRolesArguments = [];
    if ($pathPrefix) {
        $listRolesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listRolesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listRolesArguments["MaxItems"] = $maxItems;
    }
    return $this->iamClient->listRoles($listRolesArguments);
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[ListRoles](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウント内のすべての IAM ロールのリストを取得します。

```
Get-IAMRoleList
```

例 2: このコードスニペットの例では、AWS アカウント内の IAM ロールのリストを取得して一度に 3 つずつ表示し、各グループ間で [Enter] キーが押されるのを待ちます。前の呼び出しの **Marker** 値を渡して、次のグループの開始位置を指定します。

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
}
```

```
} while ($nextMarker -ne $null)
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListRoles](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListRoles](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count
      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListRoles](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListRoles](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

**Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func listRoles() async throws -> [String] {
    var roleList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolesInput(marker: marker)
        let output = try await client.listRoles(input: input)

        guard let roles = output.roles else {
            return roleList
        }

        for role in roles {
            if let name = role.roleName {
                roleList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return roleList
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListRoles](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListSAMLProviders** を使用する

以下のコード例は、ListSAMLProviders の使用方法を示しています。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// List SAML authentication providers.
/// </summary>
/// <returns>A list of SAML providers.</returns>
public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
{
    var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
    return response.SAMLProviderList;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## CLI

### AWS CLI

AWS アカウント内の SAML プロバイダーを一覧表示するには

この例では、現在の AWS アカウントで作成された SAML 2.0 プロバイダーのリストを取得します。

```
aws iam list-saml-providers
```

出力:

```
{
```




```
"SAMLProviderList": [  
  {  
    "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",  
    "ValidUntil": "2015-06-05T22:45:14Z",  
    "CreateDate": "2015-06-05T22:45:14Z"  
  }  
]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM SAML ID プロバイダーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListSAMLProviders](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account  
actions  
// used in the examples.  
// It contains an IAM service client that is used to perform account actions.  
type AccountWrapper struct {  
  IamClient *iam.Client  
}  
  
// ListSAMLProviders gets the SAML providers for the account.  
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,  
error) {  
  var providers []types.SAMLProviderListEntry  
  result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),  
    &iam.ListSAMLProvidersInput{})
```

```
if err != nil {
    log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
} else {
    providers = result.SAMLProviderList
}
return providers, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

SAML プロバイダーを一覧表示します。

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});


export const listSamlProviders = async () => {
    const command = new ListSAMLProvidersCommand({});

    const response = await client.send(command);
    console.log(response);
    return response;
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## PHP

## SDK for PHP

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function listSAMLProviders()
{
    return $this->iamClient->listSAMLProviders();
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## PowerShell

## Tools for PowerShell

例 1: この例では、現在の AWS アカウントで作成された SAML 2.0 プロバイダーのリストを取得します。これは、各 SAML プロバイダーの ARN、作成日、有効期限を返します。

```
Get-IAMSAMLProviderList
```

出力:

Arn	CreateDate
ValidUntil	
---	-----
-----	
arn:aws:iam::123456789012:saml-provider/SAMLADFS	12/23/2014 12:16:55 PM
12/23/2114 12:16:54 PM	

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListSAMLProviders](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class SAMLProviderLister
  # Initializes the SAMLProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **ListServerCertificates** を使用する

以下のコード例は、ListServerCertificates の使用方法を示しています。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::listServerCertificates(
    const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(14) << "UploadDate" << std::setw(14) <<
                "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
            outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                certificate.GetServerCertificateName() << std::setw(30) <<
                certificate.GetServerCertificateId() << std::setw(80) <<
                certificate.GetArn() << std::setw(14) <<

certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                std::setw(14) <<

certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
    }
}
```

```
    }
    else {
        done = true;
    }
}

return true;
}
```

- APIの詳細については、「AWS SDK for C++ SDK for Rust API リファレンス」の「[ListServerCertificates](#)」を参照してください。

## CLI

### AWS CLI

AWS アカウント内のサーバー証明書を一覧表示するには

次の `list-server-certificates` コマンドは、AWS アカウントに保存され、当該アカウントで使用できるすべてのサーバー証明書を一覧表示します。

```
aws iam list-server-certificates
```

出力:

```
{
  "ServerCertificateMetadataList": [
    {
      "Path": "/",
      "ServerCertificateName": "myUpdatedServerCertificate",
      "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:server-certificate/myUpdatedServerCertificate",
      "UploadDate": "2019-04-22T21:13:44+00:00",
      "Expiration": "2019-10-15T22:23:16+00:00"
    },
    {
      "Path": "/cloudfront/",
      "ServerCertificateName": "MyTestCert",
      "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",

```



```
        "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
        "UploadDate": "2015-04-21T18:14:16+00:00",
        "Expiration": "2018-01-14T17:52:36+00:00"
    }
]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM でのサーバー証明書の管理](#)」を参照してください。

- API の詳細については、「AWS CLI API リファレンス」の「[ListServerCertificates](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

証明書を一覧表示します。

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
simplify this.
 *
 */
export async function* listServerCertificates() {
    const command = new ListServerCertificatesCommand({});
    let response = await client.send(command);
```

```
while (response.ServerCertificateMetadataList?.length) {
  for await (const cert of response.ServerCertificateMetadataList) {
    yield cert;
  }

  if (response.IsTruncated) {
    response = await client.send(new ListServerCertificatesCommand({}));
  } else {
    break;
  }
}
}
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Rust API リファレンス」の「[ListServerCertificates](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript SDK for Rust API リファレンス」の「[ListServerCertificates](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントにアップロードされたサーバー証明書のリストを取得します。

```
Get-IAMServerCertificateList
```

出力:

```
Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path               : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListServerCertificates](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```
class ServerCertificateManager
```

```
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
  @logger.progname = "ServerCertificateManager"
end

# Creates a new server certificate.
# @param name [String] the name of the server certificate
# @param certificate_body [String] the contents of the certificate
# @param private_key [String] the private key contents
# @return [Boolean] returns true if the certificate was successfully created
def create_server_certificate(name, certificate_body, private_key)
  @iam_client.upload_server_certificate({
    server_certificate_name: name,
    certificate_body: certificate_body,
    private_key: private_key,
  })

  true
rescue Aws::IAM::Errors::ServiceError => e
  puts "Failed to create server certificate: #{e.message}"
  false
end

# Lists available server certificate names.
def list_server_certificate_names
  response = @iam_client.list_server_certificates

  if response.server_certificate_metadata_list.empty?
    @logger.info("No server certificates found.")
    return
  end

  response.server_certificate_metadata_list.each do |certificate_metadata|
    @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
  end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing server certificates: #{e.message}")
end

# Updates the name of a server certificate.
def update_server_certificate_name(current_name, new_name)
  @iam_client.update_server_certificate(
    server_certificate_name: current_name,
```

```
        new_server_certificate_name: new_name
      )
      @logger.info("Server certificate name updated from '#{current_name}' to
        '#{new_name}'.")
      true
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Error updating server certificate name: #{e.message}")
      false
    end

    # Deletes a server certificate.
    def delete_server_certificate(name)
      @iam_client.delete_server_certificate(server_certificate_name: name)
      @logger.info("Server certificate '#{name}' deleted.")
      true
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Error deleting server certificate: #{e.message}")
      false
    end
  end
end
```

- APIの詳細については、「AWS SDK for Ruby SDK for Rust API リファレンス」の「[ListServerCertificates](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListSigningCertificates** を使用する

以下のコード例は、ListSigningCertificates の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーの署名証明書を一覧表示するには

次の list-signing-certificates コマンドは、Bob という名前の IAM ユーザーの署名証明書を一覧表示します。

```
aws iam list-signing-certificates \  
  --user-name Bob
```

出力:

```
{  
  "Certificates": [  
    {  
      "UserName": "Bob",  
      "Status": "Inactive",  
      "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-  
body>-----END CERTIFICATE-----",  
      "CertificateId": "TA7SMP42TDN5Z260BPJE7EXAMPLE",  
      "UploadDate": "2013-06-06T21:40:08Z"  
    }  
  ]  
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「[署名証明書の管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListSigningCertificates](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前のユーザーに関連付けられている署名証明書に関する詳細を取得します。

```
Get-IAMSigningCertificate -UserName Bob
```

出力:

```
CertificateBody : -----BEGIN CERTIFICATE-----  
  
MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC  
  
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
```

```

b24xFDASBgNVBAsTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHzAd
BqkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHzAdBqkqhkiG9w0BCQEWEG5vb25lQGFT
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/
f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE

Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAtCu4
nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status         : Active
UploadDate    : 4/20/2015 1:26:01 PM
UserName      : Bob

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListSigningCertificates](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ListUserPolicies** を使用する

以下のコード例は、ListUserPolicies の使用方法を示しています。

## CLI

### AWS CLI

IAM ユーザーのポリシーを一覧表示するには

次の `list-user-policies` コマンドは、Bob という名前の IAM ユーザーにアタッチされているポリシーを一覧表示します。

```
aws iam list-user-policies \  
  --user-name Bob
```

出力:

```
{  
  "PolicyNames": [  
    "ExamplePolicy",  
    "TestPolicy"  
  ]  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[AWS アカウントでの IAM ユーザーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI API リファレンス」の「[ListUserPolicies](#)」を参照してください。

## Go

### SDK for Go V2

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.  
// It contains an IAM service client that is used to perform user actions.  
type UserWrapper struct {
```



```
IamClient *iam.Client
}

// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
        &iam.ListUserPoliciesInput{
            UserName: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
            err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListUserPolicies](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**David** という名前の IAM ユーザーに埋め込まれているインラインポリシーの名前一覧を取得します。

```
Get-IAMUserPolicyList -UserName David
```

出力:

```
 Davids_IAM_Admin_Policy
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListUserPolicies](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `ListUserTags` を使用する

以下のコード例は、`ListUserTags` の使用方法を示しています。

### CLI

#### AWS CLI

ユーザーにアタッチされたタグを一覧表示するには

次の `list-user-tags` コマンドは、指定された IAM ユーザーに関連付けられているタグを取得します。

```
aws iam list-user-tags \  
  --user-name alice
```

出力:

```
{  
  "Tags": [  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    },  
    {  
      "Key": "DeptID",  
      "Value": "12345"  
    }  
  ],  
  "IsTruncated": false  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListUserTags](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ユーザーに関連付けられているタグを取得します。

```
Get-IAMUserTagList -UserName joe
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListUserTags](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **ListUsers** を使用する

以下のコード例は、ListUsers の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>  
/// List IAM users.  
/// </summary>  
/// <returns>A list of IAM users.</returns>  
public async Task<List<User>> ListUsersAsync()  
{
```

```
var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
var users = new List<User>();

await foreach (var response in listUsersPaginator.Responses)
{
    users.AddRange(response.Users);
}

return users;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[ListUsers](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_list_users
#
# List the IAM users in the account.
#
```

```
# Returns:
#     The list of users names
# And:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_list_users() {
    local option OPTARG # Required to use getopt command in a function.
    local error_code
    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_list_users"
        echo "Lists the AWS Identity and Access Management (IAM) user in the
account."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "h" option; do
        case "${option}" in
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    local response

    response=$(aws iam list-users \
        --output text \
        --query "Users[].UserName")
    error_code=${?}

    if [[ $error_code -ne 0 ]]; then
        aws_cli_error_log $error_code
        errecho "ERROR: AWS reports list-users operation failed.$response"
        return 1
    fi
}
```

```
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[ListUsers](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
{
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                std::setw(20) << "CreateDate" << std::endl;
        }
    }
}
```

```
        header = true;
    }

    const auto &users = outcome.GetResult().GetUsers();
    for (const auto &user: users) {
        std::cout << std::left << std::setw(32) << user.GetUserName() <<
            std::setw(30) << user.GetUserId() << std::setw(64) <<
            user.GetArn() << std::setw(20) <<
            user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
            << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[ListUsers](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーを一覧表示するには

次の `list-users` コマンドは、現在のアカウントの IAM ユーザーを一覧表示します。

```
aws iam list-users
```

出力:

```
{
  "Users": [
    {
```


```
    "UserName": "Adele",
    "Path": "/",
    "CreateDate": "2013-03-07T05:14:48Z",
    "UserId": "AKIAI44QH8DHBEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:user/Adele"
  },
  {
    "UserName": "Bob",
    "Path": "/",
    "CreateDate": "2012-09-21T23:03:13Z",
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::123456789012:user/Bob"
  }
]
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーの一覧表示](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListUsers](#)」を参照してください。

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    IamClient *iam.Client
}

// ListUsers gets up to maxUsers number of users.
```



```
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
    var users []types.User
    result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
        MaxItems: aws.Int32(maxUsers),
    })
    if err != nil {
        log.Printf("Couldn't list users. Here's why: %v\n", err)
    } else {
        users = result.Users
    }
    return users, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の「[ListUsers](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
                }
            }
        }
    }
}
```

```
        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListUsers](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ユーザーを一覧表示します。

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
    const command = new ListUsersCommand({ MaxItems: 10 });

    const response = await client.send(command);
    response.Users?.forEach(({ UserName, CreateDate }) => {
        console.log(`${UserName} created on: ${CreateDate}`);
    });
};
```

```
    return response;
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListUsers](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListUsers](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[ListUsers](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
$uuid = uniqid();
$service = new IAMService();

public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listUsersArguments = [];
    if ($pathPrefix) {
        $listUsersArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listUsersArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listUsersArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listUsers($listUsersArguments);
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[ListUsers](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、現在の AWS アカウントのユーザーのコレクションを取得します。

```
Get-IAMUserList
```

出力:

```
Arn          : arn:aws:iam::123456789012:user/Administrator
CreateDate   : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path         : /
UserId       : 7K3GJEANSKZF2EXAMPLE1
UserName     : Administrator

Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/6/2015 12:54:42 PM
```

```
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path              : /
UserId           : L3EWNONDOM3YUEXAMPLE2
UserName        : bab

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path            : /
UserId          : Y4FKWQCXTA52QEXAMPLE3
UserName        : David
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListUsers](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListUsers](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListUsers](#)」を参照してください。



## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListUsers](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

**Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func listUsers() async throws -> [MyUserRecord] {
    var userList: [MyUserRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListUsersInput(marker: marker)
        let output = try await client.listUsers(input: input)

        guard let users = output.users else {
            return userList
        }

        for user in users {
            if let id = user.userId, let name = user.userName {
                userList.append(MyUserRecord(id: id, name: name))
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return userList
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListUsers](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListVirtualMfaDevices** を使用する

以下のコード例は、ListVirtualMfaDevices の使用方法を示しています。

## CLI

### AWS CLI

仮想 MFA デバイスを一覧表示するには

次の `list-virtual-mfa-devices` コマンドは、現在のアカウントに設定されている仮想 MFA デバイスを一覧表示します。

```
aws iam list-virtual-mfa-devices
```

出力:

```
{
  "VirtualMFADevices": [
    {
      "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
    },
    {
      "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
    }
  ]
}
```

詳細については、「AWS IAM ユーザーガイド」の「[仮想 Multi-Factor Authentication \(MFA\) デバイスの有効化 \(コンソール\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ListVirtualMfaDevices](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、AWS アカウント内のユーザーに割り当てられている仮想 MFA デバイスのコレクションを取得します。それぞれの **User** プロパティは、デバイスが割り当てられている IAM ユーザーの詳細を含むオブジェクトです。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

出力:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG       :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG       :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ListVirtualMfaDevices](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutGroupPolicy** を使用する

以下のコード例は、PutGroupPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
///  
/// <summary>
```

```
/// Add or update an inline policy document that is embedded in an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
{
    var request = new PutGroupPolicyRequest
    {
        GroupName = groupName,
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutGroupPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[PutGroupPolicy](#)」を参照してください。

## CLI

### AWS CLI

グループにポリシーを追加するには

次の `put-group-policy` コマンドは、Admins という名前の IAM グループにポリシーを追加します。

```
aws iam put-group-policy \
  --group-name Admins \
  --policy-document file://AdminPolicy.json \
  --policy-name AdminRoot
```

このコマンドでは何も出力されません。

ポリシーは、AdminPolicy.json ファイル内で JSON ドキュメントとして定義されます。(ファイル名と拡張子には意味はありません。)

IAM ポリシーの詳細については、「AWS IAM ユーザーガイド」の「[IAM ポリシーを管理する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PutGroupPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**AppTesterPolicy** という名前のインラインポリシーを作成し、IAM グループ **AppTesters** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容がファイル **apptesterpolicy.json** に送られます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要があることに注意してください。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -  
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[PutGroupPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutRolePermissionsBoundary** を使用する

以下のコード例は、PutRolePermissionsBoundary の使用方法を示しています。

### CLI

#### AWS CLI

例 1: カスタムポリシーに基づくアクセス許可の境界を IAM ロールに適用するには

次の put-role-permissions-boundary の例では、指定した IAM ロールのアクセス許可の境界として、intern-boundary という名前のカスタムポリシーを適用します。

```
aws iam put-role-permissions-boundary \  
  --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \  
  --role-name lambda-application-role
```

このコマンドでは何も出力されません。

例 2: AWS 管理ポリシーに基づくアクセス許可の境界を IAM ロールに適用するには

次の `put-role-permissions-boundary` の例では、指定した IAM ロールのアクセス許可の境界として、AWS 管理 `PowerUserAccess` ポリシーを適用します。

```
aws iam put-role-permissions-boundary \  
  --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \  
  --role-name x-account-admin
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PutRolePermissionsBoundary](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例は、IAM ロールのアクセス許可の境界を設定する方法を示しています。AWS 管理ポリシーまたはカスタムポリシーをアクセス許可の境界として設定できます。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[PutRolePermissionsBoundary](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutRolePolicy** を使用する

以下のコード例は、PutRolePolicy の使用方法を示しています。

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Update the inline policy document embedded in a role.
/// </summary>
/// <param name="policyName">The name of the policy to embed.</param>
/// <param name="roleName">The name of the role to update.</param>
/// <param name="policyDocument">The policy document that defines the role.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutRolePolicyAsync(string policyName, string
roleName, string policyDocument)
{
    var request = new PutRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
        PolicyDocument = policyDocument
    };


    var response = await _IAMService.PutRolePolicyAsync(request);
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[PutRolePolicy](#)」を参照してください。



## C++

## SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::putRolePolicy(
    const Aws::String &roleName,
    const Aws::String &policyName,
    const Aws::String &policyDocument,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
    iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[PutRolePolicy](#)」を参照してください。

## CLI

### AWS CLI

アクセス許可ポリシーを IAM ロールにアタッチするには

次の `put-role-policy` コマンドは、`Test-Role` という名前のロールにアクセス許可ポリシーを追加します。

```
aws iam put-role-policy \  
  --role-name Test-Role \  
  --policy-name ExamplePolicy \  
  --policy-document file://AdminPolicy.json
```

このコマンドでは何も出力されません。

ポリシーは、`AdminPolicy.json` ファイル内で JSON ドキュメントとして定義されます。(ファイル名と拡張子には意味はありません。)

信頼ポリシーをロールにアタッチするには、`update-assume-role-policy` コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PutRolePolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";  
  
const examplePolicyDocument = JSON.stringify({  
  Version: "2012-10-17",  
  Statement: [  
    {  
      Action: "iam:PutRolePolicy",  
      Resource: "arn:aws:iam::123456789012:role/Test-Role",  
      Effect: "Allow",  
      Principal: "AWS" }  
    ]  
});
```

```
{
  Sid: "VisualEditor0",
  Effect: "Allow",
  Action: [
    "s3:ListBucketMultipartUploads",
    "s3:ListBucketVersions",
    "s3:ListBucket",
    "s3:ListMultipartUploadParts",
  ],
  Resource: "arn:aws:s3:::some-test-bucket",
},
{
  Sid: "VisualEditor1",
  Effect: "Allow",
  Action: [
    "s3:ListStorageLensConfigurations",
    "s3:ListAccessPointsForObjectLambda",
    "s3:ListAllMyBuckets",
    "s3:ListAccessPoints",
    "s3:ListJobs",
    "s3:ListMultiRegionAccessPoints",
  ],
  Resource: "*",
},
],
});

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
}
```

```
return response;
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[PutRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**FedTesterRolePolicy** という名前のインラインポリシーを作成し、IAM ロール **FedTesterRole** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容は、ファイル **FedTesterPolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要がありますことに注意してください。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[PutRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutUserPermissionsBoundary** を使用する

以下のコード例は、**PutUserPermissionsBoundary** の使用方法を示しています。

### CLI

#### AWS CLI

例 1: カスタムポリシーに基づくアクセス許可の境界を IAM ユーザーに適用するには

次の **put-user-permissions-boundary** の例では、指定した IAM ユーザーのアクセス許可の境界として、**intern-boundary** という名前のカスタムポリシーを適用します。

```
aws iam put-user-permissions-boundary \  
  --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \  
  --user-name intern
```

このコマンドでは何も出力されません。

例 2: AWS 管理ポリシーに基づくアクセス許可の境界を IAM ユーザーに適用するには

次の `put-user-permissions-boundary` の例では、指定した IAM ユーザーのアクセス許可の境界として、`PowerUserAccess` という名前の AWS 管理ポリシーを適用します。

```
aws iam put-user-permissions-boundary \  
  --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \  
  --user-name developer
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ID アクセス許可の追加および削除](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PutUserPermissionsBoundary](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例は、ユーザーのアクセス許可の境界を設定する方法を示しています。AWS 管理ポリシーまたはカスタムポリシーをアクセス許可の境界として設定できます。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[PutUserPermissionsBoundary](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutUserPolicy** を使用する

以下のコード例は、PutUserPolicy の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ユーザーを作成してロールを引き受ける](#)

### CLI

#### AWS CLI

ポリシーを IAM ユーザーにアタッチするには

次の put-user-policy コマンドは、Bob という名前の IAM ユーザーにポリシーをアタッチします。

```
aws iam put-user-policy \  
  --user-name Bob \  
  --policy-name ExamplePolicy \  
  --policy-document file://AdminPolicy.json
```

このコマンドでは何も出力されません。


ポリシーは、AdminPolicy.json ファイル内で JSON ドキュメントとして定義されます。(ファイル名と拡張子には意味はありません。)

詳細については、「AWS IAM ユーザーガイド」の「[IAM ID アクセス許可の追加および削除](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PutUserPolicy](#)」を参照してください。

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// CreateUserPolicy adds an inline policy to a user. This example creates a
// policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
    actions []string,
    roleArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Action: actions,
            Resource: aws.String(roleArn),
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn, err)
        return err
    }
}
```

```
_, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
&iam.PutUserPolicyInput{
    PolicyDocument: aws.String(string(policyBytes)),
    PolicyName:     aws.String(policyName),
    UserName:       aws.String(userName),
})
if err != nil {
    log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
err)
}
return err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[PutUserPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**EC2AccessPolicy** という名前のインラインポリシーを作成し、IAM ユーザー **Bob** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容は、ファイル **EC2AccessPolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要がありますことに注意してください。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
(Get-Content -Raw EC2AccessPolicy.json)
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[PutUserPolicy](#)」を参照してください。



## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
    user_name: username,
    policy_name: policy_name,
    policy_document: policy_document
  })
  @logger.info("Policy #{policy_name} created for user #{username}.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  false
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutUserPolicy](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
func putUserPolicy(policyDocument: String, policyName: String, user:
IAMClientTypes.User) async throws {
    let input = PutUserPolicyInput(
        policyDocument: policyDocument,
        policyName: policyName,
        userName: user.userName
    )
    do {
        _ = try await iamClient.putUserPolicy(input: input)
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[PutUserPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `RemoveClientIdFromOpenIdConnectProvider` を使用する

以下のコード例は、`RemoveClientIdFromOpenIdConnectProvider` の使用方法を示しています。

### CLI

#### AWS CLI

指定の IAM OpenID Connect プロバイダーに対して登録されているクライアント ID のリストから、指定のクライアント ID を削除するには

この例では、ARN が `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` である IAM OIDC プロバイダーに関連付けられているクライアント ID のリストから、クライアント ID `My-TestApp-3` を削除します。

```
aws iam remove-client-id-from-open-id-connect-provider
  --client-id My-TestApp-3 \
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RemoveClientIdFromOpenIdConnectProvider](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` である IAM OIDC プロバイダーに関連付けられているクライアント ID のリストから、クライアント ID `My-TestApp-3` を削除します。

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
-OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[RemoveClientIdFromOpenIdConnectProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `RemoveRoleFromInstanceProfile` を使用する

以下のコード例は、`RemoveRoleFromInstanceProfile` の使用方法を示しています。

### CLI

#### AWS CLI

ロールをインスタンスプロファイルから削除するには

次の `remove-role-from-instance-profile` コマンドは、`ExampleInstanceProfile` という名前のインスタンスプロファイルから `Test-Role` という名前のロールを削除します。

```
aws iam remove-role-from-instance-profile \  
  --instance-profile-name ExampleInstanceProfile \  
  --role-name Test-Role
```

詳細については、「AWS IAM ユーザーガイド」の「[インスタンスプロファイルの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RemoveRoleFromInstanceProfile](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、**MyNewRole** という名前の EC2 インスタンスプロファイルから **MyNewRole** という名前のロールを削除します。IAM コンソールで作成されたインスタンスプロファイルは、この例のように、常にロールと同じ名前になります。API または CLI で作成する場合、名前は異なる場合があります。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
MyNewRole -Force
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[RemoveRoleFromInstanceProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **RemoveUserFromGroup** を使用する

以下のコード例は、RemoveUserFromGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [グループを作成しユーザーを追加します。](#)

.NET

AWS SDK for .NET

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Remove a user from an IAM group.
/// </summary>
/// <param name="userName">The username of the user to remove.</param>
/// <param name="groupName">The name of the IAM group to remove the user
from.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
{
    // Remove the user from the group.
```

```
var removeUserRequest = new RemoveUserFromGroupRequest()
{
    UserName = userName,
    GroupName = groupName,
};

var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[RemoveUserFromGroup](#)」を参照してください。

## CLI

### AWS CLI

IAM グループからユーザーを削除するには

次の `remove-user-from-group` コマンドは、Admins という名前の IAM グループから Bob というユーザーを削除します。

```
aws iam remove-user-from-group \
  --user-name Bob \
  --group-name Admins
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループへのユーザーの追加と削除](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[RemoveUserFromGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **Bob** をグループ **Testers** から削除します。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

例 2: この例では、IAM ユーザー **Theresa** がメンバーとなっているグループをすべて検索し、それらのグループから **Theresa** を削除します。

```
$groups = Get-IAMGroupForUser -UserName Theresa  
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName  
-UserName Theresa -Force }
```

例 3: この例は、IAM ユーザー **Bob** を **Testers** グループから削除する別の方法を示しています。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -  
GroupName Testers -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[RemoveUserFromGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **ResyncMfaDevice** を使用する

以下のコード例は、ResyncMfaDevice の使用方法を示しています。

### CLI

#### AWS CLI

MFA デバイスを同期するには

次の `resync-mfa-device` の例では、IAM ユーザー Bob に関連付けられていて、ARN が `arn:aws:iam::123456789012:mfa/BobsMFADevice` である MFA デバイスを、2 つの認証コードを提供した認証プログラムと同期します。

```
aws iam resync-mfa-device \  
--user-name Bob \  
--serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \  

```

```
--authentication-code1 123456 \  
--authentication-code2 987654
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ResyncMfaDevice](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **Bob** に関連付けられていて、ARN が **arn:aws:iam::123456789012:mfa/bob** である MFA デバイスを、2 つの認証コードを提供した認証プログラムと同期します。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -  
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

例 2: この例では、IAM ユーザー **Theresa** に関連付けられている IAM MFA デバイスを、シリアル番号が **ABCD12345678** であり、2 つの認証コードを提供した物理デバイスと同期します。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -  
AuthenticationCode2 987654 -UserName Theresa
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[ResyncMfaDevice](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **SetDefaultPolicyVersion** を使用する

以下のコード例は、SetDefaultPolicyVersion の使用方法を示しています。



アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [ポリシーを管理](#)
- [ポリシーを以前のバージョンにロールバックする](#)

## CLI

### AWS CLI

指定されたポリシーの指定されたバージョンを、ポリシーのデフォルトバージョンとして設定するには

この例では、ARN が `arn:aws:iam::123456789012:policy/MyPolicy` であるポリシーの v2 バージョンをデフォルトのアクティブなバージョンとして設定します。

```
aws iam set-default-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --version-id v2
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM のポリシーとアクセス許可](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[SetDefaultPolicyVersion](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: ARN が `arn:aws:iam::123456789012:policy/MyPolicy` であるポリシーの v2 バージョンをデフォルトのアクティブなバージョンとして設定します。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy  
-VersionId v2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[SetDefaultPolicyVersion](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **TagRole** を使用する

以下のコード例は、TagRole の使用方法を示しています。

### CLI

#### AWS CLI

ロールにタグを追加するには

次の tag-role コマンドは、部門名を持つタグを指定したロールに追加します。

```
aws iam tag-role --role-name my-role \  
  --tags '{"Key": "Department", "Value": "Accounting"}'
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[TagRole](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、ID 管理サービスのロールにタグを追加します。

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[TagRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **TagUser** を使用する

以下のコード例は、TagUser の使用方法を示しています。

### CLI

#### AWS CLI

ユーザーにタグを追加するには

次の tag-user コマンドは、関連付けられた部門を持つタグを指定したユーザーに追加します。

```
aws iam tag-user \  
  --user-name alice \  
  --tags '{"Key": "Department", "Value": "Accounting"}'
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[TagUser](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、ID 管理サービスのユーザーにタグを追加します。

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[TagUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UntagRole` を使用する

以下のコード例は、`UntagRole` の使用方法を示しています。

### CLI

#### AWS CLI

ロールからタグを削除するには

次の `untag-role` コマンドは、キー名「Department」のタグを指定されたロールから削除します。

```
aws iam untag-role \  
  --role-name my-role \  
  --tag-keys Department
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UntagRole](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、タグキーが「abac」の「MyRoleName」という名前のロールからタグを削除します。複数のタグを削除するには、カンマで区切ったタグキーリストを指定します。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UntagRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UntagUser` を使用する

以下のコード例は、`UntagUser` の使用方法を示しています。

### CLI

#### AWS CLI

ユーザーからタグを削除するには

次の `untag-user` コマンドは、キー名が「Department」のタグを指定されたユーザーから削除します。

```
aws iam untag-user \  
  --user-name alice \  
  --tag-keys Department
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UntagUser](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、タグキーが「abac」と「xyzw」の「joe」という名前のユーザーからタグを削除します。複数のタグを削除するには、カンマで区切ったタグキーリストを指定します。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UntagUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateAccessKey` を使用する


以下のコード例は、`UpdateAccessKey` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスキーの管理](#)

C++

SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[UpdateAccessKey](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザーのためにアクセスキーをアクティブ化または非アクティブ化するには

次の `update-access-key` コマンドは、Bob という名前の IAM ユーザーのために指定されたアクセスキー (アクセスキー ID とシークレットアクセスキー) を非アクティブ化します。

```
aws iam update-access-key \  
  --access-key-id AKIAIOSFODNN7EXAMPLE \  
  --status Inactive \  
  --user-name Bob
```

このコマンドでは何も出力されません。

キーを非アクティブ化すると、そのキーをプログラムによる AWS へのアクセスに使用できなくなります。ただし、キーは引き続き使用可能であり、再アクティブ化することができます。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateAccessKey](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s

            Where:
                username - The name of the user whose key you want to update.
\s
                accessId - The access key ID of the secret access key you
want to update.\s
                status - The status you want to assign to the secret access
key.\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
```



```
IamClient iam = IamClient.builder()
    .region(region)
    .build();

updateKey(iam, username, accessId, status);
System.out.println("Done");
iam.close();
}

public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s
to" +
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[UpdateAccessKey](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

アクセスキーを更新します。

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateAccessKey](#)」を参照してください。

## SDK for JavaScript (v2)

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateAccessKey](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーのアクセスキー **AKIAIOSFODNN7EXAMPLE** のステータスを **Inactive** に変更します。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateAccessKey](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
                    key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
            key_id
        )
    raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[UpdateAccessKey](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **UpdateAccountPasswordPolicy** を使用する

以下のコード例は、UpdateAccountPasswordPolicy の使用方法を示しています。

### CLI

#### AWS CLI

現在のアカウントのパスワードポリシーを設定または変更するには

次の update-account-password-policy コマンドは、8 文字以上の長さを要求し、パスワードに 1 つ以上の数字を要求するようにパスワードポリシーを設定します。

```
aws iam update-account-password-policy \  
  --minimum-password-length 8 \  
  --require-numbers
```

このコマンドでは何も出力されません。

アカウントのパスワードポリシーを変更すると、そのアカウントの IAM ユーザー用に作成された新しいパスワードに影響します。パスワードポリシーを変更しても、既存のパスワードには影響しません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのアカウントパスワードポリシーの設定](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateAccountPasswordPolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、指定した設定でアカウントのパスワードポリシーを更新します。コマンドに含まれていないパラメータは変更されないままにはならないことに注意してください。代わりに、デフォルト値にリセットされます。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -RequireUppercaseCharacters $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateAccountPasswordPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **UpdateAssumeRolePolicy** を使用する

以下のコード例は、UpdateAssumeRolePolicy の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ロールの信頼ポリシーを更新するには

次の update-assume-role-policy コマンドは、Test-Role という名前のロールの信頼ポリシーを更新します。

```
aws iam update-assume-role-policy \  
  --role-name Test-Role \  
  --policy-document file:///Test-Role-Trust-Policy.json
```

このコマンドでは何も出力されません。

信頼ポリシーは、Test-Role-Trust-Policy.json ファイル内で JSON ドキュメントとして定義されます。(ファイル名と拡張子には意味はありません。) 信頼ポリシーはプリンシパルを指定する必要があります。

ロールのアクセス許可ポリシーを更新するには、put-role-policy コマンドを使用します。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ロールの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateAssumeRolePolicy](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**ClientRole** という名前の IAM ロールを新しい信頼ポリシーで更新します。その内容は、ファイル **ClientRolePolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** スイッチパラメータを使用する必要があることに注意してください。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw ClientRolePolicy.json)
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateAssumeRolePolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **UpdateGroup** を使用する

以下のコード例は、UpdateGroup の使用方法を示しています。

### CLI

#### AWS CLI

IAM グループの名前を変更するには

次の `update-group` コマンドは、IAM グループ `Test` の名前を `Test-1` に変更します。

```
aws iam update-group \  
  --group-name Test \  
  --new-group-name Test-1
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの名前の変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateGroup](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM グループ `Testers` の名前を `AppTesters` に変更します。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

例 2: この例では、IAM グループ `AppTesters` のパスを `/Org1/Org2/` に変更します。これにより、グループの ARN が `arn:aws:iam::123456789012:group/Org1/Org2/AppTesters` に変更されます。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateGroup](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateLoginProfile` を使用する

以下のコード例は、`UpdateLoginProfile` の使用方法を示しています。



## CLI

### AWS CLI

IAM ユーザーのパスワードを更新するには

次の `update-login-profile` コマンドは、Bob という名前の IAM ユーザーの新しいパスワードを作成します。

```
aws iam update-login-profile \  
  --user-name Bob \  
  --password <password>
```

このコマンドでは何も出力されません。

アカウントのパスワードポリシーを設定するには、`update-account-password-policy` コマンドを使用します。新しいパスワードがアカウントのパスワードポリシーに違反している場合、コマンドは `PasswordPolicyViolation` エラーを返します。

アカウントのパスワードポリシーで許可されている場合、IAM ユーザーは `change-password` コマンドを使用して自分のパスワードを変更できます。

パスワードを安全な場所に保存します。パスワードを紛失した場合は回復できないため、`create-login-profile` コマンドを使用して新しいパスワードを作成する必要があります。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーのパスワードの管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateLoginProfile](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **Bob** に新しい一時パスワードを設定し、ユーザーが次回サインインしたときにパスワードを変更するようユーザーに要求します。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -  
PasswordResetRequired $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドリファレンス」の「[UpdateLoginProfile](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateOpenIdConnectProviderThumbprint` を使用する

以下のコード例は、`UpdateOpenIdConnectProviderThumbprint` の使用方法を示しています。

### CLI

#### AWS CLI

サーバー証明書のサムプリントの既存のリストを新しいリストに置き換えるには

この例では、ARN が `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` である OIDC プロバイダーの証明書サムプリントリストを更新して、新しいサムプリントを使用します。

```
aws iam update-open-id-connect-provider-thumbprint \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com \  
  --thumbprint-list 7359755EXAMPLEEabc3060bce3EXAMPLEEec4542a3
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM で OpenID Connect \(OIDC\) ID プロバイダーを作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateOpenIdConnectProviderThumbprint](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` である OIDC プロバイダーの証明書サムプリントリストを更

新して、新しいサムプリントを使用します。OIDC プロバイダーは、プロバイダーに関連付けられている証明書が変更されると、新しい値を共有します。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateOpenIdConnectProviderThumbprint](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **UpdateRole** を使用する

以下のコード例は、UpdateRole の使用方法を示しています。

### CLI

#### AWS CLI

IAM ロールの説明またはセッション期間を変更するには

次の update-role コマンドは、IAM ロールの説明 production-role を Main production role に変更し、最大セッション期間を 12 時間に設定します。

```
aws iam update-role \  
  --role-name production-role \  
  --description 'Main production role' \  
  --max-session-duration 43200
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ロールの説明と、ロールのセッションをリクエストできる最大セッション期間の値 (秒単位) を更新します。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **UpdateRoleDescription** を使用する

以下のコード例は、UpdateRoleDescription の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ロールの説明を変更するには

次の update-role コマンドは、IAM ロール production-role の説明を Main production role に変更します。

```
aws iam update-role-description \
  --role-name production-role \
  --description 'Main production role'
```

出力:

```
{
  "Role": {
    "Path": "/",
    "RoleName": "production-role",
    "RoleId": "AROA1234567890EXAMPLE",
```

```
"Arn": "arn:aws:iam::123456789012:role/production-role",
"CreateDate": "2017-12-06T17:16:37+00:00",
"AssumeRolePolicyDocument": {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
},
"Description": "Main production role"
}
```

詳細については、「AWS IAM ユーザーガイド」の「[ロールの変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateRoleDescription](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、アカウントの IAM ロールの説明を更新します。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateRoleDescription](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **UpdateSamlProvider** を使用する

以下のコード例は、UpdateSamlProvider の使用方法を示しています。

## CLI

### AWS CLI

既存の SAML プロバイダーのメタデータドキュメントを更新するには

この例では、ARN が `arn:aws:iam::123456789012:saml-provider/SAMLADFS` である IAM の SAML プロバイダーを、ファイル `SAMLMetaData.xml` の新しい SAML メタデータドキュメントで更新します。

```
aws iam update-saml-provider \  
  --saml-metadata-document file://SAMLMetaData.xml \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

出力:

```
{  
  "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[IAM SAML ID プロバイダーの作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateSamlProvider](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、ARN が `arn:aws:iam::123456789012:saml-provider/SAMLADFS` である IAM の SAML プロバイダーを、ファイル `SAMLMetaData.xml` の新しい SAML メタデータドキュメントで更新します。JSON ファイルの内容を正常に処理するには、`-Raw` スイッチパラメータを使用する必要があることに注意してください。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/  
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateSamlProvider](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateServerCertificate` を使用する

以下のコード例は、`UpdateServerCertificate` の使用方法を示しています。

C++

SDK for C++

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
&currentCertificateName,
                                         const Aws::String &newCertificateName,
                                         const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
                      ":" <<
                      outcome.GetError().GetMessage() << std::endl;
        }
    }
}
```

```
        result = false;
    }
    else {
        std::cout << "Certificate '" << currentCertificateName
            << "' not found." << std::endl;
    }
}

return result;
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[UpdateServerCertificate](#)」を参照してください。

## CLI

### AWS CLI

AWS アカウント内のサーバー証明書のパスまたは名前を変更するには

次の `update-server-certificate` コマンドは、証明書の名前を `myServerCertificate` から `myUpdatedServerCertificate` に変更します。また、Amazon CloudFront サービスからアクセスできるように `/cloudfront/` へのパスも変更します。このコマンドでは何も出力されません。 `list-server-certificates` コマンドを実行すると、更新の結果を表示できます。

```
aws-iam update-server-certificate \  
  --server-certificate-name myServerCertificate \  
  --new-server-certificate-name myUpdatedServerCertificate \  
  --new-path /cloudfront/
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM でのサーバー証明書の管理](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[UpdateServerCertificate](#)」を参照してください。



## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を更新します。

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateServerCertificate](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateServerCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、**MyServerCertificate** という証明書の名前を **MyRenamedServerCertificate** に変更します。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

例 2: この例では、**MyServerCertificate** という証明書を **/Org1/Org2/** というパスに移動します。これにより、リソースの ARN が **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate** に変更されます。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateServerCertificate](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end
end
```

```
# Lists available server certificate names.
def list_server_certificate_names
  response = @iam_client.list_server_certificates

  if response.server_certificate_metadata_list.empty?
    @logger.info("No server certificates found.")
    return
  end

  response.server_certificate_metadata_list.each do |certificate_metadata|
    @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
  end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing server certificates: #{e.message}")
end

# Updates the name of a server certificate.
def update_server_certificate_name(current_name, new_name)
  @iam_client.update_server_certificate(
    server_certificate_name: current_name,
    new_server_certificate_name: new_name
  )
  @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error updating server certificate name: #{e.message}")
  false
end

# Deletes a server certificate.
def delete_server_certificate(name)
  @iam_client.delete_server_certificate(server_certificate_name: name)
  @logger.info("Server certificate '#{name}' deleted.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting server certificate: #{e.message}")
  false
end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateServerCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateSigningCertificate` を使用する

以下のコード例は、`UpdateSigningCertificate` の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーの署名証明書を有効または無効にするには

次の `update-signing-certificate` コマンドは、Bob という名前の IAM ユーザーの指定された署名証明書を無効にします。

```
aws iam update-signing-certificate \  
  --certificate-id TA7SMP42TDN5Z2260BPJE7EXAMPLE \  
  --status Inactive \  
  --user-name Bob
```

署名証明書の ID を取得するには、`list-signing-certificates` コマンドを使用します。

詳細については、「Amazon EC2 ユーザーガイド」の「[署名証明書の管理](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateSigningCertificate](#)」を参照してください。

### PowerShell

#### Tools for PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーに関連付けられ、証明書 ID が **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** である証明書を更新し、無効としてマークされるようにします。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateSigningCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で `UpdateUser` を使用する

以下のコード例は、`UpdateUser` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [読み取り専用ユーザーおよび読み取り/書き込みできるユーザーを作成する](#)

C++

SDK for C++

### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);
```

```
auto outcome = iam.UpdateUser(request);
if (outcome.IsSuccess()) {
    std::cout << "IAM user " << currentUserName <<
        " successfully updated with new user name " << newUserName <<
        std::endl;
}
else {
    std::cerr << "Error updating user name for IAM user " << currentUserName
<<
        ":" << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[UpdateUser](#)」を参照してください。

## CLI

### AWS CLI

IAM ユーザー名を変更するには

次の update-user コマンドは、IAM ユーザー Bob の名前を Robert に変更します。

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

このコマンドでは何も出力されません。

詳細については、「AWS IAM ユーザーガイド」の「[IAM ユーザーグループの名前の変更](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[UpdateUser](#)」を参照してください。

## Java

## SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <curName> <newName>\s

            Where:
                curName - The current user name.\s
                newName - An updated user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
```



```
String newName = args[1];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

updateIAMUser(iam, curName, newName);
System.out.println("Done");
iam.close();
}

public static void updateIAMUser(IamClient iam, String curName, String
newName) {
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[UpdateUser](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ユーザーを更新します。

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateUser](#)」を参照してください。

### SDK for JavaScript (v2)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  Username: process.argv[2],
  NewUsername: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[UpdateUser](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?
) {
```

```
val request =
    UpdateUserRequest {
        userName = curName
        newUserName = newName
    }

IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    iamClient.updateUser(request)
    println("Successfully updated user to $newName")
}
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[UpdateUser](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、IAM ユーザー **Bob** の名前を **Robert** に変更します。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

例 2: この例では、IAM ユーザー **Bob** のパスを **/Org1/Org2/** に変更します。これにより、ユーザーの ARN は実質的に **arn:aws:iam::123456789012:user/Org1/Org2/bob** に変更されます。

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UpdateUser](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[UpdateUser](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
  false
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateUser](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **UploadServerCertificate** を使用する

以下のコード例は、UploadServerCertificate の使用方法を示しています。

#### CLI

##### AWS CLI

サーバー証明書を AWS アカウントにアップロードするには

次の `upload-server-certificate` コマンドは、サーバー証明書をアカウントにアップロードします。この例では、証明書はファイル `public_key_cert_file.pem` 内に、関連付けられたプライベートキーはファイル `my_private_key.pem` 内に、認証機関 (CA) によって提供される証明書チェーンは `my_certificate_chain_file.pem` ファイル内に、それぞれ存在しています。ファイルのアップロードが完了すると、`myServerCertificate` という名前で使用できるようになります。 `file://` で始まるパラメータは、ファイルの内容を読み取り、それをファイル名自体の代わりにパラメータ値として使用するようコマンドに指示します。

```
aws iam upload-server-certificate \  
  --server-certificate-name myServerCertificate \  
  --certificate-body file://public_key_cert_file.pem \  
  --private-key file://my_private_key.pem \  
  --certificate-chain file://my_certificate_chain_file.pem
```

出力:

```
{  
  "ServerCertificateMetadata": {  
    "Path": "/",  
    "ServerCertificateName": "myServerCertificate",  
    "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",  
    "Arn": "arn:aws:iam::1234567989012:server-certificate/  
myServerCertificate",  
    "UploadDate": "2019-04-22T21:13:44+00:00",  
    "Expiration": "2019-10-15T22:23:16+00:00"  
  }  
}
```

詳細については、IAM の使用に関するガイドの「サーバー証明書の作成、アップロード、削除」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UploadServerCertificate](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
  or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }
  }

  throw err;
}
```



```
    }  
  };  
  
  /**  
   *  
   * @param {string} certificateName  
   */  
  export const uploadServerCertificate = (certificateName) => {  
    const { cert, key } = getCertAndKey();  
    const command = new UploadServerCertificateCommand({  
      ServerCertificateName: certificateName,  
      CertificateBody: cert.toString(),  
      PrivateKey: key.toString(),  
    });  
  
    return client.send(command);  
  };  
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[UploadServerCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、新しいサーバー証明書を IAM アカウントにアップロードします。証明書本文、プライベートキー、および (オプションで) 証明書チェーンを含むファイルは、すべて PEM エンコードされる必要があります。パラメータにはファイル名ではなくファイルの実際の内容が必要であることに注意してください。ファイルの内容を正常に処理するには、**-Raw** スイッチパラメータを使用する必要があります。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody  
(Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

出力:

```
Arn           : arn:aws:iam::123456789012:server-certificate/MyTestCert  
Expiration    : 1/14/2018 9:52:36 AM  
Path          : /  
ServerCertificateId : ASCAJIEXAMPLE7J7HQZYW
```

```
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UploadServerCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **UploadSigningCertificate** を使用する

以下のコード例は、UploadSigningCertificate の使用方法を示しています。

### CLI

#### AWS CLI

IAM ユーザーの署名証明書をアップロードするには

次の upload-signing-certificate コマンドは、Bob という名前の IAM ユーザーの署名証明書をアップロードします。

```
aws iam upload-signing-certificate \
  --user-name Bob \
  --certificate-body file://certificate.pem
```

出力:

```
{
  "Certificate": {
    "UserName": "Bob",
    "Status": "Active",
    "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
CERTIFICATE-----",
    "CertificateId": "TA7SMP42TDN5Z260BPJE7EXAMPLE",
    "UploadDate": "2013-06-06T21:40:08.121Z"
  }
}
```

証明書は PEM 形式の certificate.pem という名前のファイルにあります。

詳細については、「IAM の使用に関するガイド」の「Creating and Uploading a User Signing Certificate」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UploadSigningCertificate](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: この例では、新しい X.509 署名証明書をアップロードし、**Bob** という名前の IAM ユーザーに関連付けます。証明書の本文を含むファイルは PEM でエンコードされています。**CertificateBody** パラメータには、ファイル名ではなく証明書ファイルの実際の内容が必要です。ファイルを正常に処理するには、**-Raw** スイッチパラメータを使用する必要があります。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw SampleSigningCert.pem)
```

出力:

```
CertificateBody : -----BEGIN CERTIFICATE-----  
  
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC  
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6  
b24xFDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAd  
BgkqhkiG9w0BCQEWEG5vb25lQGFTtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN  
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD  
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstC01BTSBDb25z  
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFT  
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn  
+a4GmWIWJ  
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/  
f0wYK8m9T  
rDHudUZg3qX4waLG5M43q7Wgc/  
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
```

```
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate     : 4/20/2015 1:26:01 PM
UserName       : Bob
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[UploadSigningCertificate](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用する IAM のシナリオ

以下のコード例は、AWS SDK を使用して IAM で一般的なシナリオを実装する方法を示しています。これらのシナリオは、IAM 内で複数の関数を呼び出すことによって特定のタスクを実行する方法を示しています。それぞれのシナリオには、GitHub へのリンクがあり、コードを設定および実行する方法についての説明が記載されています。

### 例

- [AWS SDK を使用してレジリエントなサービスを構築して管理](#)
- [AWS SDK を使用して IAM グループを作成し、ユーザーをグループに追加します。](#)
- [AWS SDK を使用して IAM ユーザーを作成し、AWS STS を持つロールを引き受ける](#)
- [AWS SDK を使用して読み取り専用 IAM ユーザーおよび読み取り/書き込みできる IAM ユーザーを作成する](#)
- [AWS SDK を使用して IAM アクセスキーを管理する](#)
- [AWS SDK を使用して IAM ポリシーを管理する](#)
- [AWS SDK を使用して IAM ロールを管理する](#)
- [AWS SDK を使用して IAM アカウントを管理する](#)

- [AWS SDK を使用して IAM ポリシーのバージョンをロールバックする](#)
- [AWS SDK での IAM Policy Builder API の使用](#)

## AWS SDK を使用してレジリエントなサービスを構築して管理

次のコード例は、本、映画、曲のレコメンデーションを返すロードバランシングウェブサービスの作成方法を示しています。この例は、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストやヘルスチェックに対するウェブサーバーの応答を制御できます。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
static async Task Main(string[] args)
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
```

```
.AddJsonFile("settings.json") // Load settings from .json file.
.AddJsonFile("settings.local.json",
    true) // Optionally, load local settings.
.Build();

// Set up dependency injection for the AWS services.
using var host = Host.CreateDefaultBuilder(args)
    .ConfigureLogging(logging =>
        logging.AddFilter("System", LogLevel.Debug)
            .AddFilter<DebugLoggerProvider>("Microsoft",
                LogLevel.Information)
            .AddFilter<ConsoleLoggerProvider>("Microsoft",
                LogLevel.Trace))
    .ConfigureServices((_, services) =>
        services.AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonDynamoDB>()
            .AddAWSService<IAmazonElasticLoadBalancingV2>()
            .AddAWSService<IAmazonSimpleSystemsManagement>()
            .AddAWSService<IAmazonAutoScaling>()
            .AddAWSService<IAmazonEC2>()
            .AddTransient<AutoScalerWrapper>()
            .AddTransient<ElasticLoadBalancerWrapper>()
            .AddTransient<SmParameterWrapper>()
            .AddTransient<Recommendations>()
            .AddSingleton<IConfiguration>(_configuration)
    )
    .Build();

ServicesSetup(host);
ResourcesSetup();

try
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
    Console.WriteLine(new string('-', 80));
    await Deploy(true);

    Console.WriteLine("Now let's begin the scenario.");
    Console.WriteLine(new string('-', 80));
    await Demo(true);
}
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Finally, let's clean up our resources.");
        Console.WriteLine(new string('-', 80));

        await DestroyResources(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
        await DestroyResources(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Setup any common resources, also used for integration testing.
/// </summary>
public static void ResourcesSetup()
{
    _httpClient = new HttpClient();
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
    _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    _recommendations = host.Services.GetRequiredService<Recommendations>();
    _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
    _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
}
```

```
}

/// <summary>
/// Deploy necessary resources for the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Deploy(bool interactive)
{
    var protocol = "HTTP";
    var port = 80;
    var sshPort = 22;

    Console.WriteLine(
        "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
        "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
        "against various kinds of failures.\n\n" +
        "Some of the resources create by this demo are:\n");

    Console.WriteLine(
        "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
    Console.WriteLine(
        "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
    Console.WriteLine(
        "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
    Console.WriteLine(
        "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
    if (interactive)
        Console.ReadLine();

    // Create and populate the DynamoDB table.
    var databaseTableName = _configuration["databaseName"];
    var recommendationsPath = Path.Join(_configuration["resourcePath"],
        "recommendations_objects.json");
}
```



```
    Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
    await _recommendations.CreateDatabaseWithName(databaseTableName);
    await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
    Console.WriteLine(new string('-', 80));

    // Create the EC2 Launch Template.

    Console.WriteLine(
        $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
        + "\nThis script starts a Python web server defined in the
'server.py' script. The web server\n"
        + "listens to HTTP requests on port 80 and responds to requests to
 '/' and to '/healthcheck'.\n"
        + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
        + "run a web server, such as Apache, with least-privileged
credentials.");
    Console.WriteLine(
        "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
        + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
        + "that control the flow of the demo.");

    var startupScriptPath = Path.Join(_configuration["resourcePath"],
        "server_startup_script.sh");
    var instancePolicyPath = Path.Join(_configuration["resourcePath"],
        "instance_policy.json");
    await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
    Console.WriteLine(new string('-', 80));

    Console.WriteLine(
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
        + "Availability Zone.\n");
    var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
    await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
    Console.WriteLine(new string('-', 80));
```

```
    Console.WriteLine(
        "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
        + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you're ready to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("Creating variables that control the flow of the
demo.");
    await _smParameterWrapper.Reset();

    Console.WriteLine(
        "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
        + "defines how the load balancer connects to instances. The load
balancer provides a\n"
        + "single endpoint where clients connect and dispatches requests to
instances in the group.");

    var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
    var subnets = await
        _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
    var subnetIds = subnets.Select(s => s.SubnetId).ToList();
    var targetGroup = await
        _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroup
protocol, port, defaultVpc.VpcId);

    await
        _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.L
subnetIds, targetGroup);
    await
        _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
targetGroup.TargetGroupArn);
    Console.WriteLine("\nVerifying access to the load balancer endpoint...");
    var endPoint = await
        _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
var loadBalancerAccess = await
        _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

    if (!loadBalancerAccess)
```

```
{
    Console.WriteLine("\nCouldn't connect to the load balancer, verifying
that the port is open...");

    var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
    ipString = ipString.Trim();

    var defaultSecurityGroup = await
_autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
    var portIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
ipString);
    var sshPortIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
ipString);

    if (!portIsOpen)
    {
        Console.WriteLine(
            "\nFor this example to work, the default security group for
your default VPC must\n"
            + "allows access from this computer. You can either add it
automatically from this\n"
            + "example or add it yourself using the AWS Management
Console.\n");

        if (!interactive || GetYesNoResponse(
            "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
        {
            await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
        }
    }

    if (!sshPortIsOpen)
    {
        if (!interactive || GetYesNoResponse(
            "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
        {
```

```
        await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
    }
}
loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
}

if (loadBalancerAccess)
{
    Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
    Console.WriteLine($"http://{endPoint}\n");
}
else
{
    Console.WriteLine(
        "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
        + "manually verifying that your VPC and security group are
configured correctly and that\n"
        + "you can successfully make a GET request to the load balancer
endpoint:\n");
    Console.WriteLine($"http://{endPoint}\n");
}
Console.WriteLine(new string('-', 80));
Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
if (interactive)
    Console.ReadLine();
return true;
}

/// <summary>
/// Demonstrate the steps of the scenario.
/// </summary>
/// <param name="interactive">True to run as an interactive scenario.</param>
/// <returns>Async task.</returns>
public static async Task<bool> Demo(bool interactive)
{
    var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
        "ssm_only_policy.json");
```

```
Console.WriteLine(new string('-', 80));
Console.WriteLine("Resetting parameters to starting values for demo.");
await _smParameterWrapper.Reset();

Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
    "to create situations where the web service fails, and
shows how using a resilient\n" +
    "architecture can keep the web service running in spite
of these failures.");
Console.WriteLine(new string('-', 88));
Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
if (interactive)
    await DemoActionChoices();

Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
    $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
    $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
    "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
if (interactive)
    await DemoActionChoices();

Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
```

```
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
        _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
        _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
        _autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
        _autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();
```

```
    Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
    Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
    Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
    Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
    Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

    Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
    Console.WriteLine("and take that instance out of rotation.");

    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

    Console.WriteLine($"Now, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
    Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
    Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
    Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
    Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

    await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

    Console.WriteLine($"Even while the instance is terminating and the new
instance is starting, sending a GET");
    Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
    Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
```

```
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"When all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );
    }
}
```



```
        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
            _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
            _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
            _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
            _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
            _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
            _recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
}
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
```

```
private readonly IAmazonSimpleSystemsManagement _amazonSsm;
private readonly IAmazonIdentityManagementService _amazonIam;

private readonly string _instanceType = "";
private readonly string _amiParam = "";
private readonly string _launchTemplateName = "";
private readonly string _groupName = "";
private readonly string _instancePolicyName = "";
private readonly string _instanceRoleName = "";
private readonly string _instanceProfileName = "";
private readonly string _badCredsProfileName = "";
private readonly string _badCredsRoleName = "";
private readonly string _badCredsPolicyName = "";
private readonly string _keyPairName = "";

public string GroupName => _groupName;
public string KeyPairName => _keyPairName;
public string LaunchTemplateName => _launchTemplateName;
public string InstancePolicyName => _instancePolicyName;
public string BadCredsProfileName => _badCredsProfileName;
public string BadCredsRoleName => _badCredsRoleName;
public string BadCredsPolicyName => _badCredsPolicyName;

/// <summary>
/// Constructor for the AutoScalerWrapper.
/// </summary>
/// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
/// <param name="amazonEc2">The injected EC2 client.</param>
/// <param name="amazonIam">The injected IAM client.</param>
/// <param name="amazonSsm">The injected SSM client.</param>
public AutoScalerWrapper(
    IAmazonAutoScaling amazonAutoScaling,
    IAmazonEC2 amazonEc2,
    IAmazonSimpleSystemsManagement amazonSsm,
    IAmazonIdentityManagementService amazonIam,
    IConfiguration configuration)
{
    _amazonAutoScaling = amazonAutoScaling;
    _amazonEc2 = amazonEc2;
    _amazonSsm = amazonSsm;
    _amazonIam = amazonIam;

    var prefix = configuration["resourcePrefix"];
    _instanceType = configuration["instanceType"];
```

```

    _amiParam = configuration["amiParam"];

    _launchTemplateName = prefix + "-template";
    _groupName = prefix + "-group";
    _instancePolicyName = prefix + "-pol";
    _instanceRoleName = prefix + "-role";
    _instanceProfileName = prefix + "-prof";
    _badCredsPolicyName = prefix + "-bc-pol";
    _badCredsRoleName = prefix + "-bc-role";
    _badCredsProfileName = prefix + "-bc-prof";
    _keyPairName = prefix + "-key-pair";
}

/// <summary>
/// Create a policy, role, and profile that is associated with instances with
a specified name.
/// An instance's associated profile defines a role that is assumed by the
/// instance. The role has attached policies that specify the AWS permissions
granted to
/// clients that run on the instance.
/// </summary>
/// <param name="policyName">Name to use for the policy.</param>
/// <param name="roleName">Name to use for the role.</param>
/// <param name="profileName">Name to use for the profile.</param>
/// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
/// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
/// <returns>The Arn of the profile.</returns>
public async Task<string> CreateInstanceProfileWithName(
    string policyName,
    string roleName,
    string profileName,
    string ssmOnlyPolicyFile,
    List<string>? awsManagedPolicies = null)
{
    var assumeRoleDoc = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": { " +
            "\"Service\": [ " +
                "\"ec2.amazonaws.com\"" +
            "]" +

```

```
        "}," +
        "\"Action\": \"sts:AssumeRole\" +
        "]" +
        "}";

var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

var policyArn = "";

try
{
    var createPolicyResult = await _amazonIam.CreatePolicyAsync(
        new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
    policyArn = createPolicyResult.Policy.Arn;
}
catch (EntityAlreadyExistsException)
{
    // The policy already exists, so we look it up to get the Arn.
    var policiesPaginator = _amazonIam.Paginators.ListPolicies(
        new ListPoliciesRequest()
        {
            Scope = PolicyScopeType.Local
        });
    // Get the entire list using the paginator.
    await foreach (var policy in policiesPaginator.Policies)
    {
        if (policy.PolicyName.Equals(policyName))
        {
            policyArn = policy.Arn;
        }
    }

    if (policyArn == null)
    {
        throw new InvalidOperationException("Policy not found");
    }
}

try
{
```

```
        await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = assumeRoleDoc,
        });
        await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
        {
            RoleName = roleName,
            PolicyArn = policyArn
        });
        if (awsManagedPolicies != null)
        {
            foreach (var awsPolicy in awsManagedPolicies)
            {
                await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                {
                    PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                    RoleName = roleName
                });
            }
        }
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Role already exists.");
    }

    string profileArn = "";
    try
    {
        var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
            new CreateInstanceProfileRequest()
            {
                InstanceProfileName = profileName
            });
        // Allow time for the profile to be ready.
        profileArn = profileCreateResponse.InstanceProfile.Arn;
        Thread.Sleep(10000);
        await _amazonIam.AddRoleToInstanceProfileAsync(
            new AddRoleToInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
```

```
        RoleName = roleName
    });

}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Policy already exists.");
    var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
        new GetInstanceProfileRequest()
        {
            InstanceProfileName = profileName
        });
    profileArn = profileGetResponse.InstanceProfile.Arn;
}
return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="newKeyPairName">The name of the new key pair.</param>
/// <returns>Async task.</returns>
public async Task CreateKeyPair(string newKeyPairName)
{
    try
    {
        var keyResponse = await _amazonEc2.CreateKeyPairAsync(
            new CreateKeyPairRequest() { KeyName = newKeyPairName });
        await File.WriteAllTextAsync($"{newKeyPairName}.pem",
            keyResponse.KeyPair.KeyMaterial);
        Console.WriteLine($"Created key pair {newKeyPairName}.");
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine("Key pair already exists.");
    }
}

/// <summary>
/// Delete the key pair and file by name.
/// </summary>
/// <param name="deleteKeyPairName">The key pair to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteKeyPairByName(string deleteKeyPairName)
```

```
{
    try
    {
        await _amazonEc2.DeleteKeyPairAsync(
            new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
        File.Delete($"{deleteKeyPairName}.pem");
    }
    catch (FileNotFoundException)
    {
        Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
    }
}

/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    await CreateKeyPair(_keyPairName);
    await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName, _instanceProfileName, instancePolicyPath);

    var startServerText = await File.ReadAllTextAsync(startupScriptPath);
    var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(startServerText);

    var amiLatest = await _amazonSsm.GetParameterAsync(
        new GetParameterRequest() { Name = _amiParam });
    var amiId = amiLatest.Parameter.Value;
    var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
        new CreateLaunchTemplateRequest()
        {
            LaunchTemplateName = _launchTemplateName,
```

```
        LaunchTemplateData = new RequestLaunchTemplateData()
        {
            InstanceType = _instanceType,
            ImageId = amiId,
            IamInstanceProfile =
                new
LaunchTemplateIamInstanceProfileSpecificationRequest()
            {
                Name = _instanceProfileName
            },
            KeyName = _keyPairName,
            UserData = System.Convert.ToBase64String(plainTextBytes)
        }
    });
    return launchTemplateResponse.LaunchTemplate;
}

/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
        new DescribeAvailabilityZonesRequest());
    return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
}

/// <summary>
/// Create an EC2 Auto Scaling group of a specified size and name.
/// </summary>
/// <param name="groupSize">The size for the group.</param>
/// <param name="groupName">The name for the group.</param>
/// <param name="availabilityZones">The availability zones for the group.</
param>
/// <returns>Async task.</returns>
public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
{
    try
    {
```



```
        await _amazonAutoScaling.CreateAutoScalingGroupAsync(
            new CreateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                AvailabilityZones = availabilityZones,
                LaunchTemplate =
                    new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                    {
                        LaunchTemplateName = _launchTemplateName,
                        Version = "$Default"
                    },
                MaxSize = groupSize,
                MinSize = groupSize
            });
        Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
    }
}

/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
        new DescribeVpcsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("is-default", new List<string>() { "true" })
            }
        });
    return vpcResponse.Vpcs[0];
}

/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
```

```
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    var subnets = new List<Subnet>();
    var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
        new DescribeSubnetsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("vpc-id", new List<string>() { vpcId}),
                new ("availability-zone", availabilityZones),
                new ("default-for-az", new List<string>() { "true" })
            }
        });

    // Get the entire list using the paginator.
    await foreach (var subnet in subnetPaginator.Subnets)
    {
        subnets.Add(subnet);
    }

    return subnets;
}

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonClientException)
```

```
        {
            Console.WriteLine($"Unable to delete template {templateName}.");
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
            _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        }
                    );
                }
            }
        }
        catch { }
    }
}
```

```
        });
    }
}

await _amazonIam.DeleteRoleAsync(
    new DeleteRoleRequest() { RoleName = roleName });
}
catch (NoSuchEntityException)
{
    Console.WriteLine($"Instance profile {profileName} does not exist.");
}
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{
    var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
    new DescribeAutoScalingGroupsRequest()
    {
        AutoScalingGroupNames = new List<string>() { group }
    });
    var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
        g => g.Instances.Select(i => i.InstanceId));
    return instanceIds;
}

/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
    new DescribeIamInstanceProfileAssociationsRequest()
    {
```

```
        Filters = new List<Amazon.EC2.Model.Filter>()
        {
            new ("instance-id", new List<string>() { instanceId })
        },
    });
    return response.IamInstanceProfileAssociations[0];
}

/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
        new ReplaceIamInstanceProfileAssociationRequest()
        {
            AssociationId = associationId,
            IamInstanceProfile = new IamInstanceProfileSpecification()
            {
                Name = credsProfileName
            }
        }
    );
    // Allow time before resetting.
    Thread.Sleep(25000);
    var instanceReady = false;
    var retries = 5;
    while (retries-- > 0 && !instanceReady)
    {
        await _amazonEc2.RebootInstancesAsync(
            new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(10000);
    }
}
```

```
        var instancesPaginator =
    _amazonSsm.Paginators.DescribeInstanceInformation(
        new DescribeInstanceInformationRequest());
    // Get the entire list using the paginator.
    await foreach (var instance in
instancesPaginator.InstanceInformationList)
    {
        instanceReady = instance.InstanceId == instanceId;
        if (instanceReady)
        {
            break;
        }
    }
}
Console.WriteLine($"Sending restart command to instance {instanceId}");
await _amazonSsm.SendCommandAsync(
    new SendCommandRequest()
    {
        InstanceIds = new List<string>() { instanceId },
        DocumentName = "AWS-RunShellScript",
        Parameters = new Dictionary<string, List<string>>()
        {
            {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
        }
    });
Console.WriteLine($"Restarted the web server on instance {instanceId}");
}

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
    _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
```

```
        new TerminateInstanceInAutoScalingGroupRequest()
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false
        });
        stopping = true;
    }
    catch (ScalingActivityInProgressException)
    {
        Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
        Thread.Sleep(10000);
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
            Console.WriteLine($"Some instances are still running.
Waiting...");
            Thread.Sleep(10000);
        }
    }
}
```

```
    }
  }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>Async task.</returns>
public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
{
    var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
    new DescribeAutoScalingGroupsRequest()
    {
        AutoScalingGroupNames = new List<string>() { groupName }
    });
    if (describeGroupsResponse.AutoScalingGroups.Any())
    {
        // Update the size to 0.
        await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
            new UpdateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                MinSize = 0
            });
        var group = describeGroupsResponse.AutoScalingGroups[0];
        foreach (var instance in group.Instances)
        {
            await TryTerminateInstanceById(instance.InstanceId);
        }

        await TryDeleteGroupByName(groupName);
    }
    else
    {
        Console.WriteLine($"No groups found with name {groupName}.");
    }
}

/// <summary>
/// Get the default security group for a specified Vpc.
```



```
/// </summary>
/// <param name="vpc">The Vpc to search.</param>
/// <returns>The default security group.</returns>
public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
{
    var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
        new DescribeSecurityGroupsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("group-name", new List<string>() { "default" }),
                new ("vpc-id", new List<string>() { vpc.VpcId })
            }
        });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.
/// This can be done by allowing ingress from this computer's IP address.
/// In some situations, such as connecting from a corporate network, you must
instead specify
/// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
/// If you do, be sure to remove public access when you're done.
/// </summary>
/// <param name="vpc">The group to check.</param>
/// <param name="port">The port to verify.</param>
/// <param name="ipAddress">This computer's IP address.</param>
/// <returns>True if the ip address is allowed on the group.</returns>
public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
{
    var portIsOpen = false;
    foreach (var ipPermission in group.IpPermissions)
    {
        if (ipPermission.FromPort == port)
        {
            foreach (var ipRange in ipPermission.Ipv4Ranges)
            {
                var cidr = ipRange.CidrIp;
                if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                {
```

```
        portIsOpen = true;
    }
}

if (ipPermission.PrefixListIds.Any())
{
    portIsOpen = true;
}

if (!portIsOpen)
{
    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                    "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
}
else
{
    break;
}
}
}

return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
{
    await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest()
        {
            GroupId = groupId,
            IpPermissions = new List<IpPermission>()
            {
                new IpPermission()
            }
        }
    );
}
```

```

        {
            FromPort = port,
            ToPort = port,
            IpProtocol = "tcp",
            Ipv4Ranges = new List<IpRange>()
            {
                new IpRange() { CidrIp = $"{ipAddress}/32" }
            }
        }
    });
}

/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
/// <param name="targetGroupArn">The Arn for the target group.</param>
/// <returns>Async task.</returns>
public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
{
    await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
        new AttachLoadBalancerTargetGroupsRequest()
        {
            AutoScalingGroupName = autoScalingGroupName,
            TargetGroupARNs = new List<string>() { targetGroupArn }
        });
}
}

```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```

/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{

```

```
private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
private string? _endpoint = null;
private readonly string _targetGroupName = "";
private readonly string _loadBalancerName = "";
HttpClient _httpClient = new();

public string TargetGroupName => _targetGroupName;
public string LoadBalancerName => _loadBalancerName;

/// <summary>
/// Constructor for the Elastic Load Balancer wrapper.
/// </summary>
/// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
client.</param>
/// <param name="configuration">The injected configuration.</param>
public ElasticLoadBalancerWrapper(
    IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
    IConfiguration configuration)
{
    _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
    var prefix = configuration["resourcePrefix"];
    _targetGroupName = prefix + "-tg";
    _loadBalancerName = prefix + "-lb";
}

/// <summary>
/// Get the HTTP Endpoint of a load balancer by its name.
/// </summary>
/// <param name="loadBalancerName">The name of the load balancer.</param>
/// <returns>The HTTP endpoint.</returns>
public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
{
    if (_endpoint == null)
    {
        var endpointResponse =
            await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { loadBalancerName }
                });
        _endpoint = endpointResponse.LoadBalancers[0].DNSName;
    }
}
```

```
        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
    CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
```

```
        Console.WriteLine($"Target group {groupName} not found.");
    }
    return result;
}

/// <summary>
/// Create an Elastic Load Balancing target group. The target group specifies
how the load balancer forwards
/// requests to instances in the group and how instance health is checked.
///
/// To speed up this demo, the health check is configured with shortened
times and lower thresholds. In production,
/// you might want to decrease the sensitivity of your health checks to avoid
unwanted failures.
/// </summary>
/// <param name="groupName">The name for the group.</param>
/// <param name="protocol">The protocol, such as HTTP.</param>
/// <param name="port">The port to use to forward requests, such as 80.</
param>
/// <param name="vpcId">The Id of the Vpc in which the load balancer
exists.</param>
/// <returns>The new TargetGroup object.</returns>
public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
ProtocolEnum protocol, int port, string vpcId)
{
    var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
    new CreateTargetGroupRequest()
    {
        Name = groupName,
        Protocol = protocol,
        Port = port,
        HealthCheckPath = "/healthcheck",
        HealthCheckIntervalSeconds = 10,
        HealthCheckTimeoutSeconds = 5,
        HealthyThresholdCount = 2,
        UnhealthyThresholdCount = 2,
        VpcId = vpcId
    });
    var targetGroup = createResponse.TargetGroups[0];
    return targetGroup;
}

/// <summary>
```

```
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
        _amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
                    _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
                    describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
                    LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
        }
    }
}
```

```
        Thread.Sleep(10000);
    }
    // Create the listener.
    await _amazonElasticLoadBalancingV2.CreateListenerAsync(
        new CreateListenerRequest()
        {
            LoadBalancerArn = loadBalancerArn,
            Protocol = targetGroup.Protocol,
            Port = targetGroup.Port,
            DefaultActions = new List<Action>()
            {
                new Action()
                {
                    Type = ActionTypeEnum.Forward,
                    TargetGroupArn = targetGroup.TargetGroupArn
                }
            }
        });
    return createLbResponse.LoadBalancers[0];
}

/// <summary>
/// Verify this computer can successfully send a GET request to the
/// load balancer endpoint.
/// </summary>
/// <param name="endpoint">The endpoint to check.</param>
/// <returns>True if successful.</returns>
public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
{
    var success = false;
    var retries = 3;
    while (!success && retries > 0)
    {
        try
        {
            var endpointResponse = await _httpClient.GetAsync($"http://{
{endpoint}");
            Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

            if (endpointResponse.IsSuccessStatusCode)
            {
                success = true;
            }
            else

```



```
        {
            retries = 0;
        }
    }
    catch (HttpRequestException)
    {
        Console.WriteLine("Connection error, retrying...");
        retries--;
        Thread.Sleep(10000);
    }
}

return success;
}

/// <summary>
/// Delete a load balancer by its specified name.
/// </summary>
/// <param name="name">The name of the load balancer to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteLoadBalancerByName(string name)
{
    try
    {
        var describeLoadBalancerResponse =
            await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { name }
                });
        var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
        await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
            new DeleteLoadBalancerRequest()
            {
                LoadBalancerArn = lbArn
            }
        );
    }
    catch (LoadBalancerNotFoundException)
    {
        Console.WriteLine($"Load balancer {name} not found.");
    }
}
}
```

```
/// <summary>
/// Delete a TargetGroup by its specified name.
/// </summary>
/// <param name="groupName">Name of the group to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTargetGroupByName(string groupName)
{
    var done = false;
    while (!done)
    {
        try
        {
            var groupResponse =
                await
                _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });

            var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
            await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
            Console.WriteLine($"Deleted load balancing target group
{groupName}.");
            done = true;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine(
                $"Target group {groupName} not found, could not delete.");
            done = true;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine("Target group not yet released, waiting...");
            Thread.Sleep(10000);
        }
    }
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
/// movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.WriteLine($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
AttributeDefinitions = new List<AttributeDefinition>()
{
    new AttributeDefinition()
    {
        AttributeName = "MediaType",
        AttributeType = ScalarAttributeType.S
    },
    new AttributeDefinition()
    {
        AttributeName = "ItemId",
        AttributeType = ScalarAttributeType.N
    }
},
KeySchema = new List<KeySchemaElement>()
{
    new KeySchemaElement()
    {
        AttributeName = "MediaType",
        KeyType = KeyType.HASH
    },
    new KeySchemaElement()
    {
        AttributeName = "ItemId",
        KeyType = KeyType.RANGE
    }
},
ProvisionedThroughput = new ProvisionedThroughput()
{
    ReadCapacityUnits = 5,
    WriteCapacityUnits = 5
}
};
await _amazonDynamoDb.CreateTableAsync(createRequest);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("\nWaiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = tableName
};

TableStatus status;
do
```

```
        {
            Thread.Sleep(2000);

            var describeTableResponse = await
                _amazonDynamoDb.DescribeTableAsync(request);
            status = describeTableResponse.Table.TableStatus;

            Console.WriteLine(".");
        }
        while (status != "ACTIVE");

        return status == TableStatus.ACTIVE;
    }
    catch (ResourceInUseException)
    {
        Console.WriteLine($"Table {tableName} already exists.");
        return false;
    }
}

/// <summary>
/// Populate the database table with data from a specified path.
/// </summary>
/// <param name="databaseTableName">The name of the table.</param>
/// <param name="recommendationsPath">The path of the recommendations data.</
param>
/// <returns>Async task.</returns>
public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
{
    var recommendationsText = await
        File.ReadAllTextAsync(recommendationsPath);
    var records =

        JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
    var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

    foreach (var record in records!)
    {
        batchWrite.AddPutItem(record);
    }

    await batchWrite.ExecuteAsync();
}
```

```
/// <summary>
/// Delete the recommendation table by name.
/// </summary>
/// <param name="tableName">The name of the recommendation table.</param>
/// <returns>Async task.</returns>
public async Task DestroyDatabaseByName(string tableName)
{
    try
    {
        await _amazonDynamoDb.DeleteTableAsync(
            new DeleteTableRequest() { TableName = tableName });
        Console.WriteLine($"Table {tableName} was deleted.");
    }
    catch (ResourceNotFoundException)
    {
        Console.WriteLine($"Table {tableName} not found");
    }
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
/// parameters
/// to drive the demonstration of resilient architecture, such as failure of a
/// dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
        _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";
}
```

```
public string TableParameter => _tableParameter;
public string TableName => _tableName;
public string HealthCheckParameter => _healthCheckParameter;
public string FailureResponseParameter => _failureResponseParameter;

/// <summary>
/// Constructor for the SmParameterWrapper.
/// </summary>
/// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
Management client.</param>
/// <param name="configuration">The injected configuration.</param>
public SmParameterWrapper(IAmazonSimpleSystemsManagement
amazonSimpleSystemsManagement, IConfiguration configuration)
{
    _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
    _tableName = configuration["databaseName"]!;
}

/// <summary>
/// Reset the Systems Manager parameters to starting values for the demo.
/// </summary>
/// <returns>Async task.</returns>
public async Task Reset()
{
    await this.PutParameterByName(_tableParameter, _tableName);
    await this.PutParameterByName(_failureResponseParameter, "none");
    await this.PutParameterByName(_healthCheckParameter, "shallow");
}

/// <summary>
/// Set the value of a named Systems Manager parameter.
/// </summary>
/// <param name="name">The name of the parameter.</param>
/// <param name="value">The value to set.</param>
/// <returns>Async task.</returns>
public async Task PutParameterByName(string name, string value)
{
    await _amazonSimpleSystemsManagement.PutParameterAsync(
        new PutParameterRequest() { Name = name, Value = value, Overwrite =
true });
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)
  - [DescribeIamInstanceProfileAssociations](#)
  - [DescribeInstances](#)
  - [DescribeLoadBalancers](#)
  - [DescribeSubnets](#)
  - [DescribeTargetGroups](#)
  - [DescribeTargetHealth](#)
  - [DescribeVpcs](#)
  - [RebootInstances](#)
  - [ReplaceIamInstanceProfileAssociation](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)



## Java

## SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;
```

```
public static final String DASHES = new String(new char[80]).replace("\0",
"-");

public static void main(String[] args) throws IOException,
InterruptedException {
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and
Manage a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.

        To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

    if (userInput.equals("y")) {
```

```
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("**** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """

            For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
            to set up a load-balanced web service endpoint and
explore some ways to make it resilient
            against various kinds of failures.

            Some of the resources create by this demo are:
```

```
        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to
`/` and to `/healthcheck`.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged
credentials.

        The template also defines an IAM policy that each instance uses
to assume a role that grants
        permissions to access the DynamoDB recommendation table and
Systems Manager parameters
        that control the flow of the demo.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
    """);

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the
demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load
balancer. The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
```

```
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group
for your default VPC must
                allow access from this computer. You can either add
it automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
```

```
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessul) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
    System.out.println("you can successfully make a GET request to the
load balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
```

This part of the demonstration shows how to toggle different parts of the system to create situations where the web service fails, and shows how using a resilient architecture can keep the web service running in spite of these failures.

At the start, the load balancer endpoint returns recommendations and reports that all targets are healthy.

```
        """);  
demoChoices(loadBalancer);
```

```
System.out.println(  
    ""
```

The web service running on the EC2 instances gets recommendations by querying a DynamoDB table.

The table name is contained in a Systems Manager parameter named `self.param_helper.table`.

To simulate a failure of the recommendation service, let's set this parameter to name a non-existent table.

```
        """);  
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
System.out.println(  
    ""
```

\nNow, sending a GET request to the load balancer endpoint returns a failure code. But, the service reports as healthy to the load balancer because shallow health checks don't check for failure of the recommendation service.

```
        """);  
demoChoices(loadBalancer);
```

```
System.out.println(  
    ""
```

Instead of failing when the recommendation service fails, the web service can return a static response.

While this is not a perfect solution, it presents the customer with a somewhat better experience than failure.

```
        """);  
paramHelper.put(paramHelper.failureResponse, "static");
```

```
System.out.println("""
```

Now, sending a GET request to the load balancer endpoint returns a static response.



```
        The service still reports as healthy because health checks are
still shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance
id value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " +
profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
        ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
        """);

demoChoices(loadBalancer);
```

```
System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on
for recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto
Scaling instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    """"
        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start
a new instance to replace it.
    """);
autoScaler.terminateInstance(badInstanceId);

System.out.println("""
```

Even while the instance is terminating and the new instance is starting, sending a GET request to the web service continues to get a successful recommendation response because the load balancer routes requests to the healthy instances. After the replacement instance starts and reports as healthy, it is included in the load balancing rotation.

Note that terminating and replacing an instance typically takes several minutes, during which time you can see the changing health check status until the new instance is running and healthy.

```
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
            all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
            one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
```

```
System.out.println("-".repeat(88));

switch (choice) {
    case 0 -> {
        System.out.println("Request:\n");
        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
        CloseableHttpClient httpClient =
HttpClientClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode =
response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
```

```
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
health check to update
                                Note that it can take a minute or two for the
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
}

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;
```

```
private static SsmClient ssmClient;

private IAMClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IAMClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private EC2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = EC2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
```

```
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile
is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
    // name.

    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.

    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
```

```
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
    while (!instReady) {
        if (tries % 6 == 0) {
            getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                .build());
            System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
        }
        tries++;
        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
```



```
        Collections.singletonList("cd / && sudo python3 server.py
80"))))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
    {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
```

```
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
        .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.policyArn())
        .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
```

```
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")
                .build();

            Filter filter1 = Filter.builder()
                .name("vpc-id")
                .values(VPC)
                .build();
```

```
DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);
String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
groupInfo.setGroupName(securityGroup);

for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
    System.out.println("Found security group: " +
secGroup.groupId());

    for (IpPermission ipPermission : secGroup.ipPermissions()) {
        if (ipPermission.fromPort() == port) {
            System.out.println("Found inbound rule: " +
ipPermission);

            for (IpRange ipRange : ipPermission.ipRanges()) {
                String cidrIp = ipRange.cidrIp();
                if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                    System.out.println(cidrIp + " is applicable");
                    portIsOpen = true;
                }
            }

            if (!ipPermission.prefixListIds().isEmpty()) {
                System.out.println("Prefix lList is applicable");
                portIsOpen = true;
            }

            if (!portIsOpen) {
                System.out
                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
            } else {
                break;
            }
        }
    }
}
```

```
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
```

```
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability
Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```

```
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```



```
ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
for (Policy policy : listPoliciesResponse.policies()) {
    if (policy.policyName().equals(policyName)) {
        // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
    || !listEntitiesResponse.policyRoles().isEmpty()) {
    // Detach the policy from any entities it is attached to.
DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy.arn())
    .roleName(roleName) // Specify the name of the IAM
role

    .build();

getIAMClient().detachRolePolicy(detachPolicyRequest);
System.out.println("Policy detached from entities.");
}

// Now, you can delete the policy.
DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

getIAMClient().deletePolicy(deletePolicyRequest);
System.out.println("Policy deleted successfully.");
break;
}
}
```

```
// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
```

```
        elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

.targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
```

```
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load
balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
    boolean success = false;
    int retries = 3;
```

```
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how
instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
```

```
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
```

```
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

            .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                .defaultActions(action)
                .port(port)
                .protocol(protocol)
                .defaultActions(action)
                .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
```



```
* hash key named 'MediaType' that defines the type of media recommended,
such
* as
* Book or Movie, and a range key named 'ItemId' that, combined with the
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException
{
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
    }
}
```

```
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws
IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
    }
}
```

```
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)
  - [DescribeIamInstanceProfileAssociations](#)
  - [DescribeInstances](#)
  - [DescribeLoadBalancers](#)
  - [DescribeSubnets](#)
  - [DescribeTargetGroups](#)
  - [DescribeTargetHealth](#)
  - [DescribeVpcs](#)
  - [RebootInstances](#)
  - [ReplaceIamInstanceProfileAssociation](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 * - deploy
 * - demo
 * - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
```

```
* that simplifies running a series of steps.
*/
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

すべてのリソースをデプロイするための手順を作成します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
```

```
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
];
```

```
),
new ScenarioOutput(
  "creatingTable",
  MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioAction("createTable", async () => {
  const client = new DynamoDBClient({});
  await client.send(
    new CreateTableCommand({
      TableName: NAMES.tableName,
      ProvisionedThroughput: {
        ReadCapacityUnits: 5,
        WriteCapacityUnits: 5,
      },
      AttributeDefinitions: [
        {
          AttributeName: "MediaType",
          AttributeType: "S",
        },
        {
          AttributeName: "ItemId",
          AttributeType: "N",
        },
      ],
      KeySchema: [
        {
          AttributeName: "MediaType",
          KeyType: "HASH",
        },
        {
          AttributeName: "ItemId",
          KeyType: "RANGE",
        },
      ],
    }),
  );
  await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
}),
new ScenarioOutput(
  "createdTable",
  MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioOutput(
  "populatingTable",
```



```
MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioAction("populateTable", () => {
  const client = new DynamoDBClient({});
  /**
   * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
  */
  const recommendations = JSON.parse(
    readFileSync(join(RESOURCES_PATH, "recommendations.json")),
  );

  return client.send(
    new BatchWriteItemCommand({
      RequestItems: {
        [NAMES.tableName]: recommendations.map((item) => ({
          PutRequest: { Item: item },
        })),
      },
    }),
  );
}),
new ScenarioOutput(
  "populatedTable",
  MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioOutput(
  "creatingKeyPair",
  MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
),
new ScenarioAction("createKeyPair", async () => {
  const client = new EC2Client({});
  const { KeyMaterial } = await client.send(
    new CreateKeyPairCommand({
      KeyName: NAMES.keyPairName,
    }),
  );

  writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
}),
new ScenarioOutput(
  "createdKeyPair",
  MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
),
```

```
new ScenarioOutput(
  "creatingInstancePolicy",
  MESSAGES.creatingInstancePolicy.replace(
    "${INSTANCE_POLICY_NAME}",
    NAMES.instancePolicyName,
  ),
),
new ScenarioAction("createInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const {
    Policy: { Arn },
  } = await client.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.instancePolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "instance_policy.json"),
      ),
    }),
  );
  state.instancePolicyArn = Arn;
}),
new ScenarioOutput("createdInstancePolicy", (state) =>
  MESSAGES.createdInstancePolicy
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
),
new ScenarioOutput(
  "creatingInstanceRole",
  MESSAGES.creatingInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioAction("createInstanceRole", () => {
  const client = new IAMClient({});
  return client.send(
    new CreateRoleCommand({
      RoleName: NAMES.instanceRoleName,
      AssumeRolePolicyDocument: readFileSync(
        join(ROOT, "assume-role-policy.json"),
      ),
    }),
  );
}),
```

```
new ScenarioOutput(
  "createdInstanceRole",
  MESSAGES.createdInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioOutput(
  "attachingPolicyToRole",
  MESSAGES.attachingPolicyToRole
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
),
new ScenarioAction("attachPolicyToRole", async (state) => {
  const client = new IAMClient({});
  await client.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.instanceRoleName,
      PolicyArn: state.instancePolicyArn,
    }),
  );
}),
new ScenarioOutput(
  "attachedPolicyToRole",
  MESSAGES.attachedPolicyToRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioOutput(
  "creatingInstanceProfile",
  MESSAGES.creatingInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  ),
),
new ScenarioAction("createInstanceProfile", async (state) => {
  const client = new IAMClient({});
  const {
    InstanceProfile: { Arn },
  } = await client.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.instanceProfileName,
    }),
  );
});
```

```
state.instanceProfileArn = Arn;

await waitUntilInstanceProfileExists(
  { client },
  { InstanceProfileName: NAMES.instanceProfileName },
);
}),
new ScenarioOutput("createdInstanceProfile", (state) =>
  MESSAGES.createdInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
),
new ScenarioOutput(
  "addingRoleToInstanceProfile",
  MESSAGES.addingRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioAction("addRoleToInstanceProfile", () => {
  const client = new IAMClient({});
  return client.send(
    new AddRoleToInstanceProfileCommand({
      RoleName: NAMES.instanceRoleName,
      InstanceProfileName: NAMES.instanceProfileName,
    })),
  );
}),
new ScenarioOutput(
  "addedRoleToInstanceProfile",
  MESSAGES.addedRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
  const ssmClient = new SSMClient({});
  const { Parameter } = await ssmClient.send(
    new GetParameterCommand({
      Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    })),
  );
  const ec2Client = new EC2Client({});
```

```
await ec2Client.send(
  new CreateLaunchTemplateCommand({
    LaunchTemplateName: NAMES.launchTemplateName,
    LaunchTemplateData: {
      InstanceType: "t3.micro",
      ImageId: Parameter.Value,
      IamInstanceProfile: { Name: NAMES.instanceProfileName },
      UserData: readFileSync(
        join(RESOURCES_PATH, "server_startup_script.sh"),
      ).toString("base64"),
      KeyName: NAMES.keyPairName,
    },
  }),
  // snippet-end:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
);
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),
new ScenarioAction("createAutoScalingGroup", async (state) => {
  const ec2Client = new EC2Client({});
  const { AvailabilityZones } = await ec2Client.send(
    new DescribeAvailabilityZonesCommand({}),
  );
  state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
  const autoScalingClient = new AutoScalingClient({});
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    autoScalingClient.send(
      new CreateAutoScalingGroupCommand({
        AvailabilityZones: state.availabilityZoneNames,
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
```

```

        Version: "$Default",
    },
    MinSize: 3,
    MaxSize: 3,
  )),
),
);
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeVpcs]
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }]},
    ),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeVpcs]
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeSubnets]
  const client = new EC2Client({});

```

```
const { Subnets } = await client.send(
  new DescribeSubnetsCommand({
    Filters: [
      { Name: "vpc-id", Values: [state.defaultVpc] },
      { Name: "availability-zone", Values: state.availabilityZoneNames },
      { Name: "default-for-az", Values: ["true"] },
    ],
  }),
);
// snippet-end:[javascript.v3.wkflw.resilient.DescribeSubnets]
state.subnets = Subnets.map((subnet) => subnet.SubnetId);
}),
new ScenarioOutput(
  "gotSubnets",
  /**
   * @param {{ subnets: string[] }} state
   */
  (state) =>
    MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
),
new ScenarioOutput(
  "creatingLoadBalancerTargetGroup",
  MESSAGES.creatingLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateTargetGroup]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new CreateTargetGroupCommand({
      Name: NAMES.loadBalancerTargetGroupName,
      Protocol: "HTTP",
      Port: 80,
      HealthCheckPath: "/healthcheck",
      HealthCheckIntervalSeconds: 10,
      HealthCheckTimeoutSeconds: 5,
      HealthyThresholdCount: 2,
      UnhealthyThresholdCount: 2,
      VpcId: state.defaultVpc,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateTargetGroup]
```

```
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }},
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
  }},
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
```



```
new ScenarioAction("createListener", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateListener]
  const client = new ElasticLoadBalancingV2Client({});
  const { Listeners } = await client.send(
    new CreateListenerCommand({
      LoadBalancerArn: state.loadBalancerArn,
      Protocol: state.targetGroupProtocol,
      Port: state.targetGroupPort,
      DefaultActions: [
        { Type: "forward", TargetGroupArn: state.targetGroupArn },
      ],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateListener]
  const listener = Listeners[0];
  state.loadBalancerListenerArn = listener.ListenerArn;
}),
new ScenarioOutput("createdListener", (state) =>
  MESSAGES.createdLoadBalancerListener.replace(
    "${LB_LISTENER_ARN}",
    state.loadBalancerListenerArn,
  ),
),
new ScenarioOutput(
  "attachingLoadBalancerTargetGroup",
  MESSAGES.attachingLoadBalancerTargetGroup
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
    .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
),
new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.AttachTargetGroup]
  const client = new AutoScalingClient({});
  await client.send(
    new AttachLoadBalancerTargetGroupsCommand({
      AutoScalingGroupName: NAMES.autoScalingGroupName,
      TargetGroupARNs: [state.targetGroupArn],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.AttachTargetGroup]
}),
new ScenarioOutput(
  "attachedLoadBalancerTargetGroup",
  MESSAGES.attachedLoadBalancerTargetGroup,
),
```

```
new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
new ScenarioAction(
  "verifyInboundPort",
  /**
   *
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
   */
  async (state) => {
    const client = new EC2Client({});
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({
        Filters: [{ Name: "group-name", Values: ["default"] }],
      }),
    );
    if (!SecurityGroups) {
      state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
    }
    state.defaultSecurityGroup = SecurityGroups[0];

    /**
     * @type {string}
     */
    const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
    state.myIp = ipResponse.trim();
    const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
      ({ IpRanges }) =>
        IpRanges.some(
          ({ CidrIp }) =>
            CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
        ),
    )
      .filter(({ IpProtocol }) => IpProtocol === "tcp")
      .filter(({ FromPort }) => FromPort === 80);

    state.myIpRules = myIpRules;
  },
),
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
```

```
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    } else {
      return MESSAGES.noIpRules;
    }
  },
),
new ScenarioInput(
  "shouldAddInboundRule",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return false;
    } else {
      return MESSAGES.noIpRules;
    }
  },
  { type: "confirm" },
),
new ScenarioAction(
  "addInboundRule",
  /**
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup }} state
   */
  async (state) => {
    if (!state.shouldAddInboundRule) {
      return;
    }

    const client = new EC2Client({});
    await client.send(
      new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.defaultSecurityGroup.GroupId,
        CidrIp: `${state.myIp}/32`,
        FromPort: 80,
        ToPort: 80,
        IpProtocol: "tcp",
      })),

```

```
    );
  },
),
new ScenarioOutput("addedInboundRule", (state) => {
  if (state.shouldAddInboundRule) {
    return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
  } else {
    return false;
  }
}),
new ScenarioOutput("verifyingEndpoint", (state) =>
  MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioAction("verifyEndpoint", async (state) => {
  try {
    const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
      axios.get(`http://${state.loadBalancerDns}`),
    );
    state.endpointResponse = JSON.stringify(response.data, null, 2);
  } catch (e) {
    state.verifyEndpointError = e;
  }
}),
new ScenarioOutput("verifiedEndpoint", (state) => {
  if (state.verifyEndpointError) {
    console.error(state.verifyEndpointError);
  } else {
    return MESSAGES.verifiedEndpoint.replace(
      "${ENDPOINT_RESPONSE}",
      state.endpointResponse,
    );
  }
}),
];
```

デモを実行するための手順を作成します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { readFileSync } from "node:fs";
import { join } from "node:path";
```

```
import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "../constants.js";
import { findLoadBalancer } from "../shared.js";
```

```
const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetGroups]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetGroups]

  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  state.targetHealthDescriptions = TargetHealthDescriptions;
});
```

```
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
  balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
    },
  },
);
```

```
        output: getHealthCheckResult,
      },
    ],
  );

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
];
```



```
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    ),
  ),
```

```
);
}),
new ScenarioAction(
  "badCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
   */
  async (state) => {
    await createSsmOnlyInstanceProfile();
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    // snippet-start:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    // snippet-end:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    // snippet-start:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );
    // snippet-end:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
```

```
    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
      }),
    );
  },
),
new ScenarioOutput(
  "testBadCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}} state
   */
  (state) =>
    MESSAGES.demoTestBadCredentials.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
),
loadBalancerLoop,
new ScenarioInput(
```

```
"deepHealthCheckConfirmation",
MESSAGES.demoDeepHealthCheckConfirmation,
{ type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
  if (!state.deepHealthCheckConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("deepHealthCheck", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmHealthCheckKey,
      Value: "deep",
      Overwrite: true,
      Type: "String",
    }),
  );
}),
new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "killInstanceConfirmation",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
   ssm').InstanceInformation }} state
   */
  (state) =>
    MESSAGES.demoKillInstanceConfirmation.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
  { type: "confirm" },
),
new ScenarioAction("killInstanceExit", (state) => {
  if (!state.killInstanceConfirmation) {
    process.exit();
  }
}),
new ScenarioAction(
  "killInstance",
  /**
```

```
    * @param {{ targetInstance: import('@aws-sdk/client-
    ssm').InstanceInformation }} state
    */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
```

```
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  })),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
}
```

```
);
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: Policy.Arn,
  }),
);
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
  }),
);
// snippet-start:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
// snippet-end:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
await iamClient.send(
  new AddRoleToInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    RoleName: NAMES.ssmOnlyRoleName,
  }),
);

return InstanceProfile;
}
```

すべてのリソースを破棄するための手順を作成します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
```

```
    EC2Client,
    DeleteKeyPairCommand,
    DeleteLaunchTemplateCommand,
} from "@aws-sdk/client-ec2";
import {
    IAMClient,
    DeleteInstanceProfileCommand,
    RemoveRoleFromInstanceProfileCommand,
    DeletePolicyCommand,
    DeleteRoleCommand,
    DetachRolePolicyCommand,
    paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
    AutoScalingClient,
    DeleteAutoScalingGroupCommand,
    TerminateInstanceInAutoScalingGroupCommand,
    UpdateAutoScalingGroupCommand,
    paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
    DeleteLoadBalancerCommand,
    DeleteTargetGroupCommand,
    DescribeTargetGroupsCommand,
    ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
    ScenarioOutput,
    ScenarioInput,
    ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
    new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
    new ScenarioAction(
        "abort",
```



```
(state) => state.destroy === false && process.exit(),
),
new ScenarioAction("deleteTable", async (c) => {
  try {
    const client = new DynamoDBClient({});
    await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
  } catch (e) {
    c.deleteTableError = e;
  }
}),
new ScenarioOutput("deleteTableResult", (state) => {
  if (state.deleteTableError) {
    console.error(state.deleteTableError);
    return MESSAGES.deleteTableError.replace(
      "${TABLE_NAME}",
      NAMES.tableName,
    );
  } else {
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }
}),
new ScenarioAction("deleteKeyPair", async (state) => {
  try {
    const client = new EC2Client({});
    await client.send(
      new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
    );
    unlinkSync(`${NAMES.keyPairName}.pem`);
  } catch (e) {
    state.deleteKeyPairError = e;
  }
}),
new ScenarioOutput("deleteKeyPairResult", (state) => {
  if (state.deleteKeyPairError) {
    console.error(state.deleteKeyPairError);
    return MESSAGES.deleteKeyPairError.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  } else {
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }
});
```

```
    }
  })),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          })
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  })),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.detachedPolicyFromRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  })),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`
      );
    } else {
```

```
    return client.send(
      new DeletePolicyCommand({
        PolicyArn: policy.Arn,
      }),
    );
  }
}),
new ScenarioOutput("deletePolicyResult", (state) => {
  if (state.deletePolicyError) {
    console.error(state.deletePolicyError);
    return MESSAGES.deletePolicyError.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  } else {
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }
}),
new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new RemoveRoleFromInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  } catch (e) {
    state.removeRoleFromInstanceProfileError = e;
  }
}),
new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
  if (state.removeRoleFromInstanceProfile) {
    console.error(state.removeRoleFromInstanceProfileError);
    return MESSAGES.removeRoleFromInstanceProfileError
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  } else {
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
});
```

```
    }
  )),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  )),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    } else {
      return MESSAGES.deletedInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
  )),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  )),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
```

```
    console.error(state.deleteInstanceProfileError);
    return MESSAGES.deleteInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  } else {
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
  try {
    await terminateGroupInstances(NAMES.autoScalingGroupName);
    await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
      await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
    });
  } catch (e) {
    state.deleteAutoScalingGroupError = e;
  }
}),
new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
  if (state.deleteAutoScalingGroupError) {
    console.error(state.deleteAutoScalingGroupError);
    return MESSAGES.deleteAutoScalingGroupError.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  } else {
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }
}),
new ScenarioAction("deleteLaunchTemplate", async (state) => {
  const client = new EC2Client({});
  try {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
    await client.send(
      new DeleteLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
      }),
    ),
```

```
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
  } catch (e) {
    state.deleteLaunchTemplateError = e;
  }
}),
new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
  if (state.deleteLaunchTemplateError) {
    console.error(state.deleteLaunchTemplateError);
    return MESSAGES.deleteLaunchTemplateError.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  } else {
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }
}),
new ScenarioAction("deleteLoadBalancer", async (state) => {
  try {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
    const client = new ElasticLoadBalancingV2Client({});
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
  } catch (e) {
    state.deleteLoadBalancerError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {
  if (state.deleteLoadBalancerError) {
    console.error(state.deleteLoadBalancerError);
  }
});
```

```
    return MESSAGES.deleteLoadBalancerError.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  } else {
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }
}),
new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
  const client = new ElasticLoadBalancingV2Client({});
  try {
    const { TargetGroups } = await client.send(
      new DescribeTargetGroupsCommand({
        Names: [NAMES.loadBalancerTargetGroupName],
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      client.send(
        new DeleteTargetGroupCommand({
          TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
      ),
    );
  } catch (e) {
    state.deleteLoadBalancerTargetGroupError = e;
  }
  // snippet-end:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
}),
new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
  if (state.deleteLoadBalancerTargetGroupError) {
    console.error(state.deleteLoadBalancerTargetGroupError);
    return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  } else {
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }
});
```

```
    );
  }
 )),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    } else {
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```



```
    console.error(state.detachSsmOnlyCustomRolePolicyError);
    return MESSAGES.detachSsmOnlyCustomRolePolicyError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  } else {
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }
}),
new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DetachRolePolicyCommand({
        RoleName: NAMES.ssmOnlyRoleName,
        PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
      }),
    );
  } catch (e) {
    state.detachSsmOnlyAWSRolePolicyError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
  if (state.detachSsmOnlyAWSRolePolicyError) {
    console.error(state.detachSsmOnlyAWSRolePolicyError);
    return MESSAGES.detachSsmOnlyAWSRolePolicyError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  } else {
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }
}),
new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DeleteInstanceProfileCommand({
        InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      }),
    );
  } catch (e) {
```

```
    state.deleteSsmOnlyInstanceProfileError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
  if (state.deleteSsmOnlyInstanceProfileError) {
    console.error(state.deleteSsmOnlyInstanceProfileError);
    return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  } else {
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }
}),
new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
  try {
    const iamClient = new IAMClient({});
    const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
    await iamClient.send(
      new DeletePolicyCommand({
        PolicyArn: ssmOnlyPolicy.Arn,
      }),
    );
  } catch (e) {
    state.deleteSsmOnlyPolicyError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
  if (state.deleteSsmOnlyPolicyError) {
    console.error(state.deleteSsmOnlyPolicyError);
    return MESSAGES.deleteSsmOnlyPolicyError.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  } else {
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }
}),
}),
```

```
new ScenarioAction("deleteSsmOnlyRole", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DeleteRoleCommand({
        RoleName: NAMES.ssmOnlyRoleName,
      }),
    );
  } catch (e) {
    state.deleteSsmOnlyRoleError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
  if (state.deleteSsmOnlyRoleError) {
    console.error(state.deleteSsmOnlyRoleError);
    return MESSAGES.deleteSsmOnlyRoleError.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  } else {
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }
}),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
```

```
*/
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    } else {
      console.log(err.name);
      throw err;
    }
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
```

```
const client = new AutoScalingClient({});
const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
for await (const page of paginatedGroups) {
  const group = page.AutoScalingGroups.find(
    (g) => g.AutoScalingGroupName === groupName,
  );
  if (group) {
    return group;
  }
}
throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)
  - [DescribeIamInstanceProfileAssociations](#)
  - [DescribeInstances](#)
  - [DescribeLoadBalancers](#)
  - [DescribeSubnets](#)
  - [DescribeTargetGroups](#)

- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
class Runner:
    def __init__(
        self, resource_path, recommendation, autoscaler, loadbalancer,
        param_helper
    ):
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

    def deploy(self):
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        print(
```

```
        "\nFor this demo, we'll use the AWS SDK for Python (Boto3) to create
several AWS resources\n"
        "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n"
        "against various kinds of failures.\n\n"
        "Some of the resources create by this demo are:\n"
    )
    print(
        "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations."
    )
    print(
        "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server."
    )
    print(
        "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones."
    )
    print(
        "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests."
    )
    print("-" * 88)
    q.ask("Press Enter when you're ready to start deploying resources.")

    print(
        f"Creating and populating a DynamoDB table named
'{self.recommendation.table_name}'."
    )
    self.recommendation.create()
    self.recommendation.populate(recommendations_path)
    print("-" * 88)

    print(
        f"Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.\n"
        f"This script starts a Python web server defined in the `server.py`
script. The web server\n"
        f"listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
        f"For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
```

```
        f"run a web server, such as Apache, with least-privileged
credentials.\n"
    )
    print(
        f"The template also defines an IAM policy that each instance uses to
assume a role that grants\n"
        f"permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
        f"that control the flow of the demo.\n"
    )
    self.autoscaler.create_template(startup_script, instance_policy)
    print("-" * 88)

    print(
        f"Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
        f"Availability Zone."
    )
    zones = self.autoscaler.create_group(3)
    print("-" * 88)
    print(
        "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
        "HTTP requests. You can see these instances in the console or
continue with the demo."
    )
    print("-" * 88)
    q.ask("Press Enter when you're ready to continue.")

    print(f"Creating variables that control the flow of the demo.\n")
    self.param_helper.reset()

    print(
        "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
        "defines how the load balancer connects to instances. The load
balancer provides a\n"
        "single endpoint where clients connect and dispatches requests to
instances in the group.\n"
    )
    vpc = self.autoscaler.get_default_vpc()
    subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
    target_group = self.loadbalancer.create_target_group(
        self.protocol, self.port, vpc["VpcId"]
```



```
)
self.loadbalancer.create_load_balancer(
    [subnet["SubnetId"] for subnet in subnets], target_group
)
self.autoscaler.attach_load_balancer_target_group(target_group)
print(f"Verifying access to the load balancer endpoint...")
lb_success = self.loadbalancer.verify_load_balancer_endpoint()
if not lb_success:
    print(
        "Couldn't connect to the load balancer, verifying that the port
is open..."
    )
    current_ip_address = requests.get(
        "http://checkip.amazonaws.com"
    ).text.strip()
    sec_group, port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.port, current_ip_address
    )
    sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.ssh_port, current_ip_address
    )
    if not port_is_open:
        print(
            "For this example to work, the default security group for
your default VPC must\n"
            "allows access from this computer. You can either add it
automatically from this\n"
            "example or add it yourself using the AWS Management Console.
\n"
        )
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.port, current_ip_address
            )
    if not ssh_port_is_open:
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
```

```
        f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
        q.is_yesno,
    ):
        self.autoscaler.open_inbound_port(
            sec_group["GroupId"], self.ssh_port, current_ip_address
        )
        lb_success = self.loadbalancer.verify_load_balancer_endpoint()
    if lb_success:
        print("Your load balancer is ready. You can access it by browsing to:
\n")
        print(f"\thttp://{self.loadbalancer.endpoint()}\n")
    else:
        print(
            "Couldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
            "manually verifying that your VPC and security group are
configured correctly and that\n"
            "you can successfully make a GET request to the load balancer
endpoint:\n"
        )
        print(f"\thttp://{self.loadbalancer.endpoint()}\n")
    print("-" * 88)
    q.ask("Press Enter when you're ready to continue with the demo.")

def demo_choices(self):
    actions = [
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo.",
    ]
    choice = 0
    while choice != 2:
        print("-" * 88)
        print(
            "\nSee the current state of the service by selecting one of the
following choices:\n"
        )
        choice = q.choose("\nWhich action would you like to take? ", actions)
        print("-" * 88)
        if choice == 0:
            print("Request:\n")
            print(f"GET http://{self.loadbalancer.endpoint()}")
            response = requests.get(f"http://{self.loadbalancer.endpoint()}")
```

```
        print("\nResponse:\n")
        print(f"{response.status_code}")
        if response.headers.get("content-type") == "application/json":
            pp(response.json())
    elif choice == 1:
        print("\nChecking the health of load balancer targets:\n")
        health = self.loadbalancer.check_target_health()
        for target in health:
            state = target["TargetHealth"]["State"]
            print(
                f"\tTarget {target['Target']['Id']} on port
{target['Target']['Port']} is {state}"
            )
            if state != "healthy":
                print(
                    f"\t\t{target['TargetHealth']['Reason']}:
{target['TargetHealth']['Description']}\n"
                )
            print(
                f"\nNote that it can take a minute or two for the health
check to update\n"
                f"after changes are made.\n"
            )
    elif choice == 2:
        print("\nOkay, let's move on.")
        print("-" * 88)

def demo(self):
    ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

    print("\nResetting parameters to starting values for demo.\n")
    self.param_helper.reset()

    print(
        "\nThis part of the demonstration shows how to toggle different parts
of the system\n"
        "to create situations where the web service fails, and shows how
using a resilient\n"
        "architecture can keep the web service running in spite of these
failures."
    )
    print("-" * 88)

    print(
```

```
        "At the start, the load balancer endpoint returns recommendations and
reports that all targets are healthy."
    )
    self.demo_choices()

    print(
        f"The web service running on the EC2 instances gets recommendations
by querying a DynamoDB table.\n"
        f"The table name is contained in a Systems Manager parameter named
'{self.param_helper.table}'.\n"
        f"To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.\n"
    )
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    print(
        "\nNow, sending a GET request to the load balancer endpoint returns a
failure code. But, the service reports as\n"
        "healthy to the load balancer because shallow health checks don't
check for failure of the recommendation service."
    )
    self.demo_choices()

    print(
        f"Instead of failing when the recommendation service fails, the web
service can return a static response.\n"
        f"While this is not a perfect solution, it presents the customer with
a somewhat better experience than failure.\n"
    )
    self.param_helper.put(self.param_helper.failure_response, "static")
    print(
        f"\nNow, sending a GET request to the load balancer endpoint returns
a static response.\n"
        f"The service still reports as healthy because health checks are
still shallow.\n"
    )
    self.demo_choices()

    print("Let's reinstate the recommendation service.\n")
    self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)
    print(
        "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n"
        "access the DynamoDB recommendation table.\n"
    )
```

```
)
self.autoscaler.create_instance_profile(
    ssm_only_policy,
    self.autoscaler.bad_creds_policy_name,
    self.autoscaler.bad_creds_role_name,
    self.autoscaler.bad_creds_profile_name,
    ["AmazonSSMManagedInstanceCore"],
)
instances = self.autoscaler.get_instances()
bad_instance_id = instances[0]
instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
print(
    f"\nReplacing the profile for instance {bad_instance_id} with a
profile that contains\n"
    f"bad credentials...\n"
)
self.autoscaler.replace_instance_profile(
    bad_instance_id,
    self.autoscaler.bad_creds_profile_name,
    instance_profile["AssociationId"],
)
print(
    "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n"
    "depending on which instance is selected by the load balancer.\n"
)
self.demo_choices()

print(
    "\nLet's implement a deep health check. For this demo, a deep health
check tests whether\n"
    "the web service can access the DynamoDB table that it depends on for
recommendations. Note that\n"
    "the deep health check is only for ELB routing and not for Auto
Scaling instance health.\n"
    "This kind of deep health check is not recommended for Auto Scaling
instance health, because it\n"
    "risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.\n"
)
print(
    "By implementing deep health checks, the load balancer can detect
when one of the instances is failing\n"
    "and take that instance out of rotation.\n"
)
```

```
)
self.param_helper.put(self.param_helper.health_check, "deep")
print(
    f"\nNow, checking target health indicates that the instance with bad
credentials ({bad_instance_id})\n"
    f"is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy \n"
    f"instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because\n"
    "the load balancer takes unhealthy instances out of its rotation.\n"
)
self.demo_choices()

print(
    "\nBecause the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy\n"
    "instance is to terminate it and let the auto scaler start a new
instance to replace it.\n"
)
self.autoscaler.terminate_instance(bad_instance_id)
print(
    "\nEven while the instance is terminating and the new instance is
starting, sending a GET\n"
    "request to the web service continues to get a successful
recommendation response because\n"
    "the load balancer routes requests to the healthy instances. After
the replacement instance\n"
    "starts and reports as healthy, it is included in the load balancing
rotation.\n"
    "\nNote that terminating and replacing an instance typically takes
several minutes, during which time you\n"
    "can see the changing health check status until the new instance is
running and healthy.\n"
)
self.demo_choices()

print(
    "\nIf the recommendation service fails now, deep health checks mean
all instances report as unhealthy.\n"
)
self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
print(
    "\nWhen all instances are unhealthy, the load balancer continues to
route requests even to\n"
```

```
        "unhealthy instances, allowing them to fail open and return a static
response rather than fail\n"
        "closed and report failure to the customer."
    )
    self.demo_choices()
    self.param_helper.reset()

def destroy(self):
    print(
        "This concludes the demo of how to build and manage a resilient
service.\n"
        "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n"
        "that were created for this demo."
    )
    if q.ask("Do you want to clean up all demo resources? (y/n) ",
q.is_yesno):
        self.loadbalancer.delete_load_balancer()
        self.loadbalancer.delete_target_group()
        self.autoscaler.delete_group()
        self.autoscaler.delete_key_pair()
        self.autoscaler.delete_template()
        self.autoscaler.delete_instance_profile(
            self.autoscaler.bad_creds_profile_name,
            self.autoscaler.bad_creds_role_name,
        )
        self.recommendation.destroy()
    else:
        print(
            "Okay, we'll leave the resources intact.\n"
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        )

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
```

```
)
parser.add_argument(
    "--resource_path",
    default="../../../../workflows/resilient_service/resources",
    help="The path to resource files used by this example, such as IAM
policies and\n"
    "instance scripts.",
)
args = parser.parse_args()

print("-" * 88)
print(
    "Welcome to the demonstration of How to Build and Manage a Resilient
Service!"
)
print("-" * 88)

prefix = "doc-example-resilience"
recommendation = RecommendationService.from_client(
    "doc-example-recommendation-service"
)
autoscaler = AutoScaler.from_client(prefix)
loadbalancer = LoadBalancer.from_client(prefix)
param_helper = ParameterHelper.from_client(recommendation.table_name)
runner = Runner(
    args.resource_path, recommendation, autoscaler, loadbalancer,
param_helper
)
actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
for action in actions:
    if action == "deploy":
        runner.deploy()
    elif action == "demo":
        runner.demo()
    elif action == "destroy":
        runner.destroy()

print("-" * 88)
print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
```



```
logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
main()
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
            created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
```

```
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

@classmethod
def from_client(cls, resource_prefix):
    """
    Creates this class from Boto3 clients.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    """
    as_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")
    return cls(
        resource_prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        as_client,
        ec2_client,
        ssm_client,
        iam_client,
    )

def create_instance_profile(
    self, policy_file, policy_name, role_name, profile_name,
    aws_managed_policies=()
):
    """
    Creates a policy, role, and profile that is associated with instances
    created by
    this class. An instance's associated profile defines a role that is
    assumed by the
    instance. The role has attached policies that specify the AWS permissions
    granted to
    clients that run on the instance.

    :param policy_file: The name of a JSON file that contains the policy
    definition to
```

```

        create and attach to the role.
:param policy_name: The name to give the created policy.
:param role_name: The name to give the created role.
:param profile_name: The name to the created profile.
:param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                the role, such as
AmazonSSMManagedInstanceCore to grant
                                use of Systems Manager to send commands to
the instance.
:return: The ARN of the profile that is created.
"""
assume_role_doc = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Service": "ec2.amazonaws.com"},
            "Action": "sts:AssumeRole",
        }
    ],
}
with open(policy_file) as file:
    instance_policy_doc = file.read()

policy_arn = None
try:
    pol_response = self.iam_client.create_policy(
        PolicyName=policy_name, PolicyDocument=instance_policy_doc
    )
    policy_arn = pol_response["Policy"]["Arn"]
    log.info("Created policy with ARN %s.", policy_arn)
except ClientError as err:
    if err.response["Error"]["Code"] == "EntityAlreadyExists":
        log.info("Policy %s already exists, nothing to do.", policy_name)
        list_pol_response = self.iam_client.list_policies(Scope="Local")
        for pol in list_pol_response["Policies"]:
            if pol["PolicyName"] == policy_name:
                policy_arn = pol["Arn"]
                break
    if policy_arn is None:
        raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

```

```
    try:
        self.iam_client.create_role(
            RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
        )
        self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
        for aws_policy in aws_managed_policies:
            self.iam_client.attach_role_policy(
                RoleName=role_name,
                PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
            )
        log.info("Created role %s and attached policy %s.", role_name,
policy_arn)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            log.info("Role %s already exists, nothing to do.", role_name)
        else:
            raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

    try:
        profile_response = self.iam_client.create_instance_profile(
            InstanceProfileName=profile_name
        )
        waiter = self.iam_client.get_waiter("instance_profile_exists")
        waiter.wait(InstanceProfileName=profile_name)
        time.sleep(10) # wait a little longer
        profile_arn = profile_response["InstanceProfile"]["Arn"]
        self.iam_client.add_role_to_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )
        log.info("Created profile %s and added role %s.", profile_name,
role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            prof_response = self.iam_client.get_instance_profile(
                InstanceProfileName=profile_name
            )
            profile_arn = prof_response["InstanceProfile"]["Arn"]
            log.info(
                "Instance profile %s already exists, nothing to do.",
profile_name
            )
        else:
```

```
        raise AutoScalerError(
            f"Couldn't create profile {profile_name} and attach it to
role\n"
            f"{role_name}: {err}"
        )
    return profile_arn

def get_instance_profile(self, instance_id):
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't get instance profile association for instance
{instance_id}: {err}"
        )
    else:
        return response["IamInstanceProfileAssociations"][0]

def replace_instance_profile(
    self, instance_id, new_instance_profile_name, profile_association_id
):
    """
    Replaces the profile associated with a running instance. After the
profile is
    replaced, the instance is rebooted to ensure that it uses the new
profile. When
    the instance is ready, Systems Manager is used to restart the Python web
server.

    :param instance_id: The ID of the instance to update.
    :param new_instance_profile_name: The name of the new profile to
associate with
                                the specified instance.
```

```
        :param profile_association_id: The ID of the existing profile association
for the
                                instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)
        inst_ready = False
        tries = 0
        while not inst_ready:
            if tries % 6 == 0:
                self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                log.info(
                    "Rebooting instance %s and waiting for it to be
ready.",
                                instance_id,
                )
            tries += 1
            time.sleep(10)
            response = self.ssm_client.describe_instance_information()
            for info in response["InstanceInformationList"]:
                if info["InstanceId"] == instance_id:
                    inst_ready = True
        self.ssm_client.send_command(
            InstanceIds=[instance_id],
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
        )
        log.info("Restarted the Python web server on instance %s.",
instance_id)
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
        )
```

```
def delete_instance_profile(self, profile_name, role_name):
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
        log.info("Deleted instance profile %s.", profile_name)
        attached_policies = self.iam_client.list_attached_role_policies(
            RoleName=role_name
        )
        for pol in attached_policies["AttachedPolicies"]:
            self.iam_client.detach_role_policy(
                RoleName=role_name, PolicyArn=pol["PolicyArn"]
            )
            if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
        self.iam_client.delete_role(RoleName=role_name)
        log.info("Deleted role %s.", role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
profile_name
            )
        else:
            raise AutoScalerError(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )

def create_key_pair(self, key_pair_name):
    """
```

```
Creates a new key pair.

:param key_pair_name: The name of the key pair to create.
:return: The newly created key pair.
"""
try:
    response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
    with open(f"{key_pair_name}.pem", "w") as file:
        file.write(response["KeyMaterial"])
        chmod(f"{key_pair_name}.pem", 0o600)
    log.info("Created key pair %s.", key_pair_name)
except ClientError as err:
    raise AutoScalerError(f"Couldn't create key pair {key_pair_name}:
{err}")

def delete_key_pair(self):
    """
    Deletes a key pair.

    :param key_pair_name: The name of the key pair to delete.
    """
    try:
        self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
        remove(f"{self.key_pair_name}.pem")
        log.info("Deleted key pair %s.", self.key_pair_name)
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't delete key pair {self.key_pair_name}: {err}"
        )
    except FileNotFoundError:
        log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
    except PermissionError:
        log.info(
            "Inadequate permissions to delete key pair %s.",
self.key_pair_name
        )
    except Exception as err:
        raise AutoScalerError(
            f"Couldn't delete key pair {self.key_pair_name}: {err}"
        )
```



```
def create_template(self, server_startup_script_file, instance_policy_file):
    """
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling. The
    launch template specifies a Bash script in its user data field that runs
    after
    the instance is started. This script installs Python packages and starts
    a
    Python web server on the instance.

    :param server_startup_script_file: The path to a Bash script file that is
    run
                                     when an instance starts.
    :param instance_policy_file: The path to a file that defines a
    permissions policy
                                to create and attach to the instance
    profile.
    :return: Information about the newly created template.
    """
    template = {}
    try:
        self.create_key_pair(self.key_pair_name)
        self.create_instance_profile(
            instance_policy_file,
            self.instance_policy_name,
            self.instance_role_name,
            self.instance_profile_name,
        )
        with open(server_startup_script_file) as file:
            start_server_script = file.read()
        ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
        ami_id = ami_latest["Parameter"]["Value"]
        lt_response = self.ec2_client.create_launch_template(
            LaunchTemplateName=self.launch_template_name,
            LaunchTemplateData={
                "InstanceType": self.inst_type,
                "ImageId": ami_id,
                "IamInstanceProfile": {"Name": self.instance_profile_name},
                "UserData": base64.b64encode(
                    start_server_script.encode(encoding="utf-8")
                ).decode(encoding="utf-8"),
                "KeyName": self.key_pair_name,
            },
        )
```

```
        template = lt_response["LaunchTemplate"]
        log.info(
            "Created launch template %s for AMI %s on %s.",
            self.launch_template_name,
            ami_id,
            self.inst_type,
        )
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.AlreadyExistsException"
        ):
            log.info(
                "Launch template %s already exists, nothing to do.",
                self.launch_template_name,
            )
        else:
            raise AutoScalerError(
                f"Couldn't create launch template
{self.launch_template_name}: {err}."
            )
        return template

def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )
        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
```

```
        )
    else:
        raise AutoScalerError(
            f"Couldn't delete launch template
{self.launch_template_name}: {err}."
        )

def get_availability_zones(self):
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get availability zones: {err}.")
    else:
        return zones

def create_group(self, group_size):
    """
    Creates an EC2 Auto Scaling group with the specified size.

    :param group_size: The number of instances to set for the minimum and
    maximum in
        the group.
    :return: The list of Availability Zones specified for the group.
    """
    zones = []
    try:
        zones = self.get_availability_zones()
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=self.group_name,
            AvailabilityZones=zones,
            LaunchTemplate={
                "LaunchTemplateName": self.launch_template_name,
                "Version": "$Default",
            },
            MinSize=group_size,
```

```
        MaxSize=group_size,
    )
    log.info(
        "Created EC2 Auto Scaling group %s with availability zones %s.",
        self.launch_template_name,
        zones,
    )
except ClientError as err:
    if err.response["Error"]["Code"] == "AlreadyExists":
        log.info(
            "EC2 Auto Scaling group %s already exists, nothing to do.",
            self.group_name,
        )
    else:
        raise AutoScalerError(
            f"Couldn't create EC2 Auto Scaling group {self.group_name}:
{err}"
        )
    return zones

def get_instances(self):
    """
    Gets data about the instances in the EC2 Auto Scaling group.

    :return: Data about the instances.
    """
    try:
        as_response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        instance_ids = [
            i["InstanceId"]
            for i in as_response["AutoScalingGroups"][0]["Instances"]
        ]
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't get instances for Auto Scaling group
{self.group_name}: {err}"
        )
    else:
        return instance_ids
```

```
def terminate_instance(self, instance_id):
    """
    Terminates and instances in an EC2 Auto Scaling group. After an instance
    is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id, ShouldDecrementDesiredCapacity=False
        )
        log.info("Terminated instance %s.", instance_id)
    except ClientError as err:
        raise AutoScalerError(f"Couldn't terminate instance {instance_id}:
{err}")

def attach_load_balancer_target_group(self, lb_target_group):
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
    Scaling group.
    The target group specifies how the load balancer forward requests to the
    instances
    in the group.

    :param lb_target_group: Data about the ELB target group to attach.
    """
    try:
        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't attach load balancer target group
{lb_target_group['TargetGroupName']}\n"
            f"to auto scaling group {self.group_name}"
        )
```

```
def _try_terminate_instance(self, inst_id):
    stopping = False
    log.info(f"Stopping {inst_id}.")
    while not stopping:
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=inst_id, ShouldDecrementDesiredCapacity=True
            )
            stopping = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ScalingActivityInProgress":
                log.info("Scaling activity in progress for %s. Waiting...",
inst_id)
                time.sleep(10)
            else:
                raise AutoScalerError(f"Couldn't stop instance {inst_id}:
{err}.")

    def _try_delete_group(self):
        """
        Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
        the function waits and retries until the group is successfully deleted.
        """
        stopped = False
        while not stopped:
            try:
                self.autoscaling_client.delete_auto_scaling_group(
                    AutoScalingGroupName=self.group_name
                )
                stopped = True
                log.info("Deleted EC2 Auto Scaling group %s.", self.group_name)
            except ClientError as err:
                if (
                    err.response["Error"]["Code"] == "ResourceInUse"
                    or err.response["Error"]["Code"] ==
"ScalingActivityInProgress"
                ):
                    log.info(
                        "Some instances are still running. Waiting for them to
stop..."
                    )
```

```
        time.sleep(10)
    else:
        raise AutoScalerError(
            f"Couldn't delete group {self.group_name}: {err}."
        )

def delete_group(self):
    """
    Terminates all instances in the group, deletes the EC2 Auto Scaling
    group.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=self.group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self._try_terminate_instance(inst_id)
                self._try_delete_group()
        else:
            log.info("No groups found named %s, nothing to do.",
self.group_name)
    except ClientError as err:
        raise AutoScalerError(f"Couldn't delete group {self.group_name}:
{err}.")

def get_default_vpc(self):
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
```

```
        raise AutoScalerError(f"Couldn't get default VPC: {err}")
    else:
        return response["Vpcs"][0]

def verify_inbound_port(self, vpc, port, ip_address):
    """
    Verify the default security group of the specified VPC allows ingress
    from this
    computer. This can be done by allowing ingress from this computer's IP
    address. In some situations, such as connecting from a corporate network,
    you
    must instead specify a prefix list ID. You can also temporarily open the
    port to
    any IP address while running this example. If you do, be sure to remove
    public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
    :param ip_address: This computer's IP address.
    :return: The default security group of the specific VPC, and a value that
    indicates
        whether the specified port is open.
    """
    try:
        response = self.ec2_client.describe_security_groups(
            Filters=[
                {"Name": "group-name", "Values": ["default"]},
                {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
            ]
        )
        sec_group = response["SecurityGroups"][0]
        port_is_open = False
        log.info("Found default security group %s.", sec_group["GroupId"])
        for ip_perm in sec_group["IpPermissions"]:
            if ip_perm.get("FromPort", 0) == port:
                log.info("Found inbound rule: %s", ip_perm)
                for ip_range in ip_perm["IpRanges"]:
                    cidr = ip_range.get("CidrIp", "")
                    if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                        port_is_open = True
                if ip_perm["PrefixListIds"]:
                    port_is_open = True
```



```
        if not port_is_open:
            log.info(
                "The inbound rule does not appear to be open to
either this computer's IP\n"
                "address of %s, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.",
                ip_address,
            )
        else:
            break
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't verify inbound rule for port {port} for VPC
{vpc['VpcId']}: {err}")
    )
else:
    return sec_group, port_is_open

def open_inbound_port(self, sec_group_id, port, ip_address):
    """
    Add an ingress rule to the specified security group that allows access on
the
    specified port from the specified IP address.

    :param sec_group_id: The ID of the security group to modify.
    :param port: The port to open.
    :param ip_address: The IP address that is granted access.
    """
    try:
        self.ec2_client.authorize_security_group_ingress(
            GroupId=sec_group_id,
            CidrIp=f"{ip_address}/32",
            FromPort=port,
            ToPort=port,
            IpProtocol="tcp",
        )
        log.info(
            "Authorized ingress to %s on port %s from %s.",
            sec_group_id,
            port,
            ip_address,
        )
    except ClientError as err:
```

```
        raise AutoScalerError(
            f"Couldn't authorize ingress to {sec_group_id} on port {port}
from {ip_address}: {err}"
        )

def get_subnets(self, vpc_id, zones):
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    try:
        response = self.ec2_client.describe_subnets(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )
        subnets = response["Subnets"]
        log.info("Found %s subnets for the specified zones.", len(subnets))
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get subnets: {err}")
    else:
        return subnets
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
class LoadBalancer:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, target_group_name, load_balancer_name, elb_client):
        """
        :param target_group_name: The name of the target group associated with
        the load balancer.
```

```
    :param load_balancer_name: The name of the load balancer.
    :param elb_client: A Boto3 Elastic Load Balancing client.
    """
    self.target_group_name = target_group_name
    self.load_balancer_name = load_balancer_name
    self.elb_client = elb_client
    self._endpoint = None

    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from a Boto3 client.

        :param resource_prefix: The prefix to give to AWS resources created by
        this class.
        """
        elb_client = boto3.client("elbv2")
        return cls(f"{resource_prefix}-tg", f"{resource_prefix}-lb", elb_client)

    def endpoint(self):
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        if self._endpoint is None:
            try:
                response = self.elb_client.describe_load_balancers(
                    Names=[self.load_balancer_name]
                )
                self._endpoint = response["LoadBalancers"][0]["DNSName"]
            except ClientError as err:
                raise LoadBalancerError(
                    f"Couldn't get the endpoint for load balancer
                    {self.load_balancer_name}: {err}")
            return self._endpoint

    def create_target_group(self, protocol, port, vpc_id):
        """
        Creates an Elastic Load Balancing target group. The target group
        specifies how
```

the load balancer forward requests to instances in the group and how instance health is checked.

To speed up this demo, the health check is configured with shortened times and lower thresholds. In production, you might want to decrease the sensitivity of your health checks to avoid unwanted failures.

```
:param protocol: The protocol to use to forward requests, such as 'HTTP'.
:param port: The port to use to forward requests, such as 80.
:param vpc_id: The ID of the VPC in which the load balancer exists.
:return: Data about the newly created target group.
"""
try:
    response = self.elb_client.create_target_group(
        Name=self.target_group_name,
        Protocol=protocol,
        Port=port,
        HealthCheckPath="/healthcheck",
        HealthCheckIntervalSeconds=10,
        HealthCheckTimeoutSeconds=5,
        HealthyThresholdCount=2,
        UnhealthyThresholdCount=2,
        VpcId=vpc_id,
    )
    target_group = response["TargetGroups"][0]
    log.info("Created load balancing target group %s.",
self.target_group_name)
except ClientError as err:
    raise LoadBalancerError(
        f"Couldn't create load balancing target group
{self.target_group_name}: {err}")
)
else:
    return target_group

def delete_target_group(self):
    """
    Deletes the target group.
    """
    done = False
```

```
while not done:
    try:
        response = self.elb_client.describe_target_groups(
            Names=[self.target_group_name]
        )
        tg_arn = response["TargetGroups"][0]["TargetGroupArn"]
        self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
        log.info(
            "Deleted load balancing target group %s.",
            self.target_group_name
        )
        done = True
    except ClientError as err:
        if err.response["Error"]["Code"] == "TargetGroupNotFound":
            log.info(
                "Load balancer target group %s not found, nothing to
do.",
                self.target_group_name,
            )
            done = True
        elif err.response["Error"]["Code"] == "ResourceInUse":
            log.info(
                "Target group not yet released from load balancer,
waiting..."
            )
            time.sleep(10)
        else:
            raise LoadBalancerError(
                f"Couldn't delete load balancing target group
{self.target_group_name}: {err}"
            )

def create_load_balancer(self, subnet_ids, target_group):
    """
    Creates an Elastic Load Balancing load balancer that uses the specified
subnets
and forwards requests to the specified target group.

:param subnet_ids: A list of subnets to associate with the load balancer.
:param target_group: An existing target group that is added as a listener
to the
                    load balancer.
:return: Data about the newly created load balancer.
```

```
"""
try:
    response = self.elb_client.create_load_balancer(
        Name=self.load_balancer_name, Subnets=subnet_ids
    )
    load_balancer = response["LoadBalancers"][0]
    log.info("Created load balancer %s.", self.load_balancer_name)
    waiter = self.elb_client.get_waiter("load_balancer_available")
    log.info("Waiting for load balancer to be available...")
    waiter.wait(Names=[self.load_balancer_name])
    log.info("Load balancer is available!")
    self.elb_client.create_listener(
        LoadBalancerArn=load_balancer["LoadBalancerArn"],
        Protocol=target_group["Protocol"],
        Port=target_group["Port"],
        DefaultActions=[
            {
                "Type": "forward",
                "TargetGroupArn": target_group["TargetGroupArn"],
            }
        ],
    )
    log.info(
        "Created listener to forward traffic from load balancer %s to
target group %s.",
        self.load_balancer_name,
        target_group["TargetGroupName"],
    )
except ClientError as err:
    raise LoadBalancerError(
        f"Failed to create load balancer {self.load_balancer_name}"
        f"and add a listener for target group
{target_group['TargetGroupName']}: {err}"
    )
else:
    self._endpoint = load_balancer["DNSName"]
    return load_balancer

def delete_load_balancer(self):
    """
    Deletes a load balancer.
    """
    try:
```

```
        response = self.elb_client.describe_load_balancers(
            Names=[self.load_balancer_name]
        )
        lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
        self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
        log.info("Deleted load balancer %s.", self.load_balancer_name)
        waiter = self.elb_client.get_waiter("load_balancers_deleted")
        log.info("Waiting for load balancer to be deleted...")
        waiter.wait(Names=[self.load_balancer_name])
    except ClientError as err:
        if err.response["Error"]["Code"] == "LoadBalancerNotFound":
            log.info(
                "Load balancer %s does not exist, nothing to do.",
                self.load_balancer_name,
            )
        else:
            raise LoadBalancerError(
                f"Couldn't delete load balancer {self.load_balancer_name}:"
                {err}"
            )

    def verify_load_balancer_endpoint(self):
        """
        Verify this computer can successfully send a GET request to the load
        balancer endpoint.
        """
        success = False
        retries = 3
        while not success and retries > 0:
            try:
                lb_response = requests.get(f"http://{self.endpoint()}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    success = True
            else:
                retries = 0
        except requests.exceptions.ConnectionError:
            log.info(
                "Got connection error from load balancer endpoint,
                retrying..."
            )
```

```
        )
        retries -= 1
        time.sleep(10)
    return success

def check_target_health(self):
    """
    Checks the health of the instances in the target group.

    :return: The health status of the target group.
    """
    try:
        tg_response = self.elb_client.describe_target_groups(
            Names=[self.target_group_name]
        )
        health_response = self.elb_client.describe_target_health(
            TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
        )
    except ClientError as err:
        raise LoadBalancerError(
            f"Couldn't check health of {self.target_group_name} targets:
{err}"
        )
    else:
        return health_response["TargetHealthDescriptions"]
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
    movies,
    and songs.
    """

    def __init__(self, table_name, dynamodb_client):
        """
        :param table_name: The name of the DynamoDB recommendations table.
```



```
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    @classmethod
    def from_client(cls, table_name):
        """
        Creates this class from a Boto3 client.

        :param table_name: The name of the DynamoDB recommendations table.
        """
        ddb_client = boto3.client("dynamodb")
        return cls(table_name, ddb_client)

    def create(self):
        """
        Creates a DynamoDB table to use a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended,
        such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
        MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
```

```
except ClientError as err:
    if err.response["Error"]["Code"] == "ResourceInUseException":
        log.info("Table %s exists, nothing to be do.", self.table_name)
    else:
        raise RecommendationServiceError(
            self.table_name, f"ClientError when creating table: {err}."
        )
else:
    return response

def populate(self, data_file):
    """
    Populates the recommendations table from a JSON file.

    :param data_file: The path to the data file.
    """
    try:
        with open(data_file) as data:
            items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
    except ClientError as err:
        raise RecommendationServiceError(
            self.table_name, f"Couldn't populate table from {data_file}:
{err}"
        )

def destroy(self):
    """
    Deletes the recommendations table.
    """
    try:
        self.dynamodb_client.delete_table(TableName=self.table_name)
        log.info("Deleting table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_not_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s deleted.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
        log.info("Table %s does not exist, nothing to do.",
self.table_name)
    else:
        raise RecommendationServiceError(
            self.table_name, f"ClientError when deleting table: {err}."
        )
```

Systems Manager のアクションをラップするクラスを作成します。

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
    to drive
    the demonstration of resilient architecture, such as failure of a dependency
    or
    how the service responds to a health check.
    """

    table = "doc-example-resilient-architecture-table"
    failure_response = "doc-example-resilient-architecture-failure-response"
    health_check = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name, ssm_client):
        """
        :param table_name: The name of the DynamoDB table that is used as a
        recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    @classmethod
    def from_client(cls, table_name):
        ssm_client = boto3.client("ssm")
        return cls(table_name, ssm_client)

    def reset(self):
        """
        Resets the Systems Manager parameters to starting values for the demo.
```

```
a
    These are the name of the DynamoDB recommendation table, no response when
    dependency fails, and shallow health checks.
    """
    self.put(self.table, self.table_name)
    self.put(self.failure_response, "none")
    self.put(self.health_check, "shallow")

def put(self, name, value):
    """
    Sets the value of a named Systems Manager parameter.

    :param name: The name of the parameter.
    :param value: The new value of the parameter.
    """
    try:
        self.ssm_client.put_parameter(
            Name=name, Value=value, Overwrite=True, Type="String"
        )
        log.info("Setting demo parameter %s to '%s'.", name, value)
    except ClientError as err:
        raise ParameterHelperError(
            f"Couldn't set parameter {name} to {value}: {err}"
        )
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)

- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して IAM グループを作成し、ユーザーをグループに追加します。

次のコードサンプルは、以下の操作方法を示しています。

- グループを作成し、そのグループに Amazon S3 のフルアクセス許可を付与します。
- Amazon S3 にアクセス許可のない新しいユーザーを作成します。
- ユーザーをグループに追加し、そのユーザーが Amazon S3 のアクセス許可を持っていることを確認してから、リソースをクリーンアップします。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;

namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
}
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
```

```
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
    {
        UserName = userName,
    });

    return response.AccessKey;

}

/// <summary>
/// Create an IAM group.
/// </summary>
/// <param name="groupName">The name to give the IAM group.</param>
/// <returns>The IAM group that was created.</returns>
public async Task<Group> CreateGroupAsync(string groupName)
{
    var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
{ GroupName = groupName });
    return response.Group;
}

/// <summary>
/// Create an IAM policy.
/// </summary>
/// <param name="policyName">The name to give the new IAM policy.</param>
/// <param name="policyDocument">The policy document for the new policy.</
param>
/// <returns>The new IAM policy object.</returns>
public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
{
    var response = await _IAMService.CreatePolicyAsync(new
CreatePolicyRequest
    {
        PolicyDocument = policyDocument,
        PolicyName = policyName,
    });

    return response.Policy;
}
```



```
/// <summary>
/// Create a new IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="rolePolicyDocument">The name of the IAM policy document
/// for the new role.</param>
/// <returns>The Amazon Resource Name (ARN) of the role.</returns>
public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePolicyDocument,
    };

    var response = await _IAMService.CreateRoleAsync(request);
    return response.Role.Arn;
}

/// <summary>
/// Create an IAM service-linked role.
/// </summary>
/// <param name="serviceName">The name of the AWS Service.</param>
/// <param name="description">A description of the IAM service-linked role.</
param>
/// <returns>The IAM role that was created.</returns>
public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
string description)
{
    var request = new CreateServiceLinkedRoleRequest
    {
        AWSServiceName = serviceName,
        Description = description
    };

    var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
    return response.Role;
}

/// <summary>
```

```
/// Create an IAM user.
/// </summary>
/// <param name="userName">The username for the new IAM user.</param>
/// <returns>The IAM user that was created.</returns>
public async Task<User> CreateUserAsync(string userName)
{
    var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
    return response.User;
}

/// <summary>
/// Delete an IAM user's access key.
/// </summary>
/// <param name="accessKeyId">The Id for the IAM access key.</param>
/// <param name="userName">The username of the user that owns the IAM
/// access key.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
{
    var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
    {
        AccessKeyId = accessKeyId,
        UserName = userName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteGroupAsync(string groupName)
{
    var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

```
/// <summary>
/// Delete an IAM policy associated with an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group associated with the
/// policy.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
{
    var request = new DeleteGroupPolicyRequest()
    {
        GroupName = groupName,
        PolicyName = policyName,
    };

    var response = await _IAMService.DeleteGroupPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM policy.
/// </summary>
/// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
/// delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeletePolicyAsync(string policyArn)
{
    var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRoleAsync(string roleName)
{
```

```
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
```

```
/// <param name="policyName">The name of the IAM policy to delete.</param>
/// <param name="userName">The username of the IAM user.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
{
    var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Detach an IAM policy from an IAM role.
/// </summary>
/// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
/// <param name="roleName">The name of the IAM role.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Gets the IAM password policy for an AWS account.
/// </summary>
/// <returns>The PasswordPolicy for the AWS account.</returns>
public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
{
    var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
    return response.PasswordPolicy;
}
```

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {
        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }

    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });
        return response.Role;
    }

    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }
}
```

```
}

/// <summary>
/// List the IAM role policies that are attached to an IAM role.
/// </summary>
/// <param name="roleName">The IAM role to list IAM policies for.</param>
/// <returns>A list of the IAM policies attached to the IAM role.</returns>
public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
{
    var attachedPolicies = new List<AttachedPolicyType>();
    var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

    await foreach (var response in attachedRolePoliciesPaginator.Responses)
    {
        attachedPolicies.AddRange(response.AttachedPolicies);
    }

    return attachedPolicies;
}

/// <summary>
/// List IAM groups.
/// </summary>
/// <returns>A list of IAM groups.</returns>
public async Task<List<Group>> ListGroupsAsync()
{
    var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
    var groups = new List<Group>();

    await foreach (var response in groupsPaginator.Responses)
    {
        groups.AddRange(response.Groups);
    }

    return groups;
}
```

```
/// <summary>
/// List IAM policies.
/// </summary>
/// <returns>A list of the IAM policies.</returns>
public async Task<List<ManagedPolicy>> ListPoliciesAsync()
{
    var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
    var policies = new List<ManagedPolicy>();

    await foreach (var response in listPoliciesPaginator.Responses)
    {
        policies.AddRange(response.Policies);
    }

    return policies;
}

/// <summary>
/// List IAM role policies.
/// </summary>
/// <param name="roleName">The IAM role for which to list IAM policies.</
param>
/// <returns>A list of IAM policy names.</returns>
public async Task<List<string>> ListRolePoliciesAsync(string roleName)
{
    var listRolePoliciesPaginator =
_IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
roleName });
    var policyNames = new List<string>();

    await foreach (var response in listRolePoliciesPaginator.Responses)
    {
        policyNames.AddRange(response.PolicyNames);
    }

    return policyNames;
}

/// <summary>
/// List IAM roles.
/// </summary>
```



```
/// <returns>A list of IAM roles.</returns>
public async Task<List<Role>> ListRolesAsync()
{
    var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
    var roles = new List<Role>();

    await foreach (var response in listRolesPaginator.Responses)
    {
        roles.AddRange(response.Roles);
    }

    return roles;
}

/// <summary>
/// List SAML authentication providers.
/// </summary>
/// <returns>A list of SAML providers.</returns>
public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
{
    var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
    return response.SAMLProviderList;
}

/// <summary>
/// List IAM users.
/// </summary>
/// <returns>A list of IAM users.</returns>
public async Task<List<User>> ListUsersAsync()
{
    var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
    var users = new List<User>();

    await foreach (var response in listUsersPaginator.Responses)
    {
        users.AddRange(response.Users);
    }

    return users;
}
```

```
}

/// <summary>
/// Remove a user from an IAM group.
/// </summary>
/// <param name="userName">The username of the user to remove.</param>
/// <param name="groupName">The name of the IAM group to remove the user
from.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
{
    // Remove the user from the group.
    var removeUserRequest = new RemoveUserFromGroupRequest()
    {
        UserName = userName,
        GroupName = groupName,
    };

    var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Add or update an inline policy document that is embedded in an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
{
    var request = new PutGroupPolicyRequest
    {
        GroupName = groupName,
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };
};
```

```
        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        }
    }
}
```

```
};

var response = await _IAMService.PutUserPolicyAsync(request);
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Wait for a new access key to be ready to use.
/// </summary>
/// <param name="accessKeyId">The Id of the access key.</param>
/// <returns>A boolean value indicating the success of the action.</returns>
public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
{
    var keyReady = false;

    do
    {
        try
        {
            var response = await _IAMService.GetAccessKeyLastUsedAsync(
                new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
            if (response.UserName is not null)
            {
                keyReady = true;
            }
        }
        catch (NoSuchEntityException)
        {
            keyReady = false;
        }
    } while (!keyReady);

    return keyReady;
}
}

using Microsoft.Extensions.Configuration;

namespace IAMGroups;

public class IAMGroups
{
```

```
private static ILogger logger = null!;

// Represents JSON code for AWS full access policy for Amazon Simple
// Storage Service (Amazon S3).
private const string S3FullAccessPolicyDocument = "{" +
    " \"Statement\" : [{" +
        " \"Action\" : [\"s3:*\"],\" +
        " \"Effect\" : \"Allow\",\" +
        " \"Resource\" : \"*\"]" +
    "}]";

static async Task Main(string[] args)
{
    // Set up dependency injection for the AWS service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
                    LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
                    LogLevel.Trace))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
                .AddTransient<IAMWrapper>()
                .AddTransient<UIWrapper>()
            )
        .Build();

    logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<IAMGroups>();

    IConfiguration configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load test settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();

    var groupUserName = configuration["GroupUserName"];
    var groupName = configuration["GroupName"];
    var groupPolicyName = configuration["GroupPolicyName"];
    var groupBucketName = configuration["GroupBucketName"];
```

```
var wrapper = host.Services.GetRequiredService<IAMWrapper>();
var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

uiWrapper.DisplayGroupsOverview();
uiWrapper.PressEnter();

// Create an IAM group.
uiWrapper.DisplayTitle("Create IAM group");
Console.WriteLine("Let's begin by creating a new IAM group.");
var group = await wrapper.CreateGroupAsync(groupName);

// Add an inline IAM policy to the group.
uiWrapper.DisplayTitle("Add policy to group");
Console.WriteLine("Add an inline policy to the group that allows members
to have full access to");
Console.WriteLine("Amazon Simple Storage Service (Amazon S3) buckets.");

await wrapper.PutGroupPolicyAsync(group.GroupName, groupPolicyName,
S3FullAccessPolicyDocument);

uiWrapper.PressEnter();

// Now create a new user.
uiWrapper.DisplayTitle("Create an IAM user");
Console.WriteLine("Now let's create a new IAM user.");
var groupUser = await wrapper.CreateUserAsync(groupUserName);

// Add the new user to the group.
uiWrapper.DisplayTitle("Add the user to the group");
Console.WriteLine("Adding the user to the group, which will give the user
the same permissions as the group.");
await wrapper.AddUserToGroupAsync(groupUser.UserName, group.GroupName);

Console.WriteLine($"User, {groupUser.UserName}, has been added to the
group, {group.GroupName}.");
uiWrapper.PressEnter();

Console.WriteLine("Now that we have created a user, and added the user to
the group, let's create an IAM access key.");

// Create access and secret keys for the user.
var accessKey = await wrapper.CreateAccessKeyAsync(groupUserName);
Console.WriteLine("Key created.");
```

```
        uiWrapper.WaitABit(15, "Waiting for the access key to be ready for
use.");

        uiWrapper.DisplayTitle("List buckets");
        Console.WriteLine("To prove that the user has access to Amazon S3, list
the S3 buckets for the account.");

        var s3Client = new AmazonS3Client(accessKey.AccessKeyId,
accessKey.SecretAccessKey);
        var stsClient = new
AmazonSecurityTokenServiceClient(accessKey.AccessKeyId,
accessKey.SecretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client, stsClient);

        var buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
{bucket.CreationDate}");
            });
        }

        // Show that the user also has write access to Amazon S3 by creating
// a new bucket.
        uiWrapper.DisplayTitle("Create a bucket");
        Console.WriteLine("Since group members have full access to Amazon S3,
let's create a bucket.");
        var success = await s3Wrapper.PutBucketAsync(groupBucketName);

        if (success)
        {
            Console.WriteLine($"Successfully created the bucket:
{groupBucketName}.");
        }

        uiWrapper.PressEnter();

        Console.WriteLine("Let's list the user's S3 buckets again to show the new
bucket.");
```

```
        buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("First delete the bucket we created.");
        await s3Wrapper.DeleteBucketAsync(groupBucketName);

        Console.WriteLine($"Now remove the user, {groupUserName}, from the group,
{groupName}.");
        await wrapper.RemoveUserFromGroupAsync(groupUserName, groupName);

        Console.WriteLine("Delete the user's access key.");
        await wrapper.DeleteAccessKeyAsync(accessKey.AccessKeyId, groupUserName);

        // Now we can safely delete the user.
        Console.WriteLine("Now we can delete the user.");
        await wrapper.DeleteUserAsync(groupUserName);

        uiWrapper.PressEnter();

        Console.WriteLine("Now we will delete the IAM policy attached to the
group.");
        await wrapper.DeleteGroupPolicyAsync(groupName, groupPolicyName);

        Console.WriteLine("Now we delete the IAM group.");
        await wrapper.DeleteGroupAsync(groupName);

        uiWrapper.PressEnter();

        Console.WriteLine("The IAM groups demo has completed.");

        uiWrapper.PressEnter();
    }
}
```



```
namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };
    }
}
```

```
        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }

    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
    { BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
```

```
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}

namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
    }
}
```

```
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
```

```
/// </summary>
/// <param name="strToCenter">The string to be centered.</param>
/// <returns>The padded string.</returns>
public string CenterString(string strToCenter)
{
    var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
    var leftPad = new string(' ', padAmount);
    return $"{leftPad}{strToCenter}";
}

/// <summary>
/// Display a line of hyphens, the centered text of the title, and another
/// line of hyphens.
/// </summary>
/// <param name="strTitle">The string to be displayed.</param>
public void DisplayTitle(string strTitle)
{
    Console.WriteLine(SepBar);
    Console.WriteLine(CenterString(strTitle));
    Console.WriteLine(SepBar);
}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }


    PressEnter();
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [AddUserToGroup](#)
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreateGroup](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeleteGroup](#)
  - [DeleteGroupPolicy](#)
  - [DeleteUser](#)
  - [PutGroupPolicy](#)
  - [RemoveUserFromGroup](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して IAM ユーザーを作成し、AWS STS を持つロールを引き受ける

次のコード例は、ユーザーを作成しロールを割り当てる方法を示しています。

 Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#)などの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- 指定したアカウントに Amazon S3 バケットへのアクセス権限を付与するロールを作成します。
- ユーザーにロールを引き受けさせるポリシーを追加します。

- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示しリソースをクリーンアップします。

## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;

namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
```

```
/// Add an existing IAM user to an existing IAM group.
/// </summary>
/// <param name="userName">The username of the user to add.</param>
/// <param name="groupName">The name of the group to add the user to.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
{
    var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
    {
        GroupName = groupName,
        UserName = userName,
    });

    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Attach an IAM policy to a role.
/// </summary>
/// <param name="policyArn">The policy to attach.</param>
/// <param name="roleName">The role that the policy will be attached to.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AttachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Create an IAM access key for a user.
/// </summary>
/// <param name="userName">The username for which to create the IAM access
```



```
/// key.</param>
/// <returns>The AccessKey.</returns>
public async Task<AccessKey> CreateAccessKeyAsync(string userName)
{
    var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
    {
        UserName = userName,
    });

    return response.AccessKey;
}

/// <summary>
/// Create an IAM group.
/// </summary>
/// <param name="groupName">The name to give the IAM group.</param>
/// <returns>The IAM group that was created.</returns>
public async Task<Group> CreateGroupAsync(string groupName)
{
    var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
{ GroupName = groupName });
    return response.Group;
}

/// <summary>
/// Create an IAM policy.
/// </summary>
/// <param name="policyName">The name to give the new IAM policy.</param>
/// <param name="policyDocument">The policy document for the new policy.</
param>
/// <returns>The new IAM policy object.</returns>
public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
{
    var response = await _IAMService.CreatePolicyAsync(new
CreatePolicyRequest
    {
        PolicyDocument = policyDocument,
        PolicyName = policyName,
    });
}
```

```
        return response.Policy;
    }

    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }

    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
}
```

```
}

/// <summary>
/// Create an IAM user.
/// </summary>
/// <param name="userName">The username for the new IAM user.</param>
/// <returns>The IAM user that was created.</returns>
public async Task<User> CreateUserAsync(string userName)
{
    var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
    return response.User;
}

/// <summary>
/// Delete an IAM user's access key.
/// </summary>
/// <param name="accessKeyId">The Id for the IAM access key.</param>
/// <param name="userName">The username of the user that owns the IAM
/// access key.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
{
    var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
    {
        AccessKeyId = accessKeyId,
        UserName = userName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteGroupAsync(string groupName)
{
```

```
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an IAM role.
    /// </summary>
```

```
/// <param name="roleName">The name of the IAM role to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRoleAsync(string roleName)
{
    var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM role policy.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="policyName">The name of the IAM role policy to delete.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
{
    var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Delete an IAM user.
/// </summary>
/// <param name="userName">The username of the IAM user to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserAsync(string userName)
{
    var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

```
/// <summary>
/// Delete an IAM user policy.
/// </summary>
/// <param name="policyName">The name of the IAM policy to delete.</param>
/// <param name="userName">The username of the IAM user.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
{
    var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Detach an IAM policy from an IAM role.
/// </summary>
/// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
/// <param name="roleName">The name of the IAM role.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
{
    var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Gets the IAM password policy for an AWS account.
/// </summary>
/// <returns>The PasswordPolicy for the AWS account.</returns>
public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
{
```

```
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }

    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {
        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }

    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }

    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
```

```
{
    var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
    return response.User;
}

/// <summary>
/// List the IAM role policies that are attached to an IAM role.
/// </summary>
/// <param name="roleName">The IAM role to list IAM policies for.</param>
/// <returns>A list of the IAM policies attached to the IAM role.</returns>
public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
{
    var attachedPolicies = new List<AttachedPolicyType>();
    var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

    await foreach (var response in attachedRolePoliciesPaginator.Responses)
    {
        attachedPolicies.AddRange(response.AttachedPolicies);
    }

    return attachedPolicies;
}

/// <summary>
/// List IAM groups.
/// </summary>
/// <returns>A list of IAM groups.</returns>
public async Task<List<Group>> ListGroupsAsync()
{
    var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
    var groups = new List<Group>();

    await foreach (var response in groupsPaginator.Responses)
    {
        groups.AddRange(response.Groups);
    }
}
```



```
        return groups;
    }

    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }

    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
_IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
}
```

```
/// <summary>
/// List IAM roles.
/// </summary>
/// <returns>A list of IAM roles.</returns>
public async Task<List<Role>> ListRolesAsync()
{
    var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
    var roles = new List<Role>();

    await foreach (var response in listRolesPaginator.Responses)
    {
        roles.AddRange(response.Roles);
    }

    return roles;
}

/// <summary>
/// List SAML authentication providers.
/// </summary>
/// <returns>A list of SAML providers.</returns>
public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
{
    var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
    return response.SAMLProviderList;
}

/// <summary>
/// List IAM users.
/// </summary>
/// <returns>A list of IAM users.</returns>
public async Task<List<User>> ListUsersAsync()
{
    var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
    var users = new List<User>();

    await foreach (var response in listUsersPaginator.Responses)
    {
```

```
        users.AddRange(response.Users);
    }

    return users;
}

/// <summary>
/// Remove a user from an IAM group.
/// </summary>
/// <param name="userName">The username of the user to remove.</param>
/// <param name="groupName">The name of the IAM group to remove the user
from.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
{
    // Remove the user from the group.
    var removeUserRequest = new RemoveUserFromGroupRequest()
    {
        UserName = userName,
        GroupName = groupName,
    };

    var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Add or update an inline policy document that is embedded in an IAM group.
/// </summary>
/// <param name="groupName">The name of the IAM group.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
{
    var request = new PutGroupPolicyRequest
    {
        GroupName = groupName,
```

```
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutGroupPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Update the inline policy document embedded in a role.
/// </summary>
/// <param name="policyName">The name of the policy to embed.</param>
/// <param name="roleName">The name of the role to update.</param>
/// <param name="policyDocument">The policy document that defines the role.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutRolePolicyAsync(string policyName, string
roleName, string policyDocument)
{
    var request = new PutRolePolicyRequest
    {
        PolicyName = policyName,
        RoleName = roleName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutRolePolicyAsync(request);
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Add or update an inline policy document that is embedded in an IAM user.
/// </summary>
/// <param name="userName">The name of the IAM user.</param>
/// <param name="policyName">The name of the IAM policy.</param>
/// <param name="policyDocument">The policy document defining the IAM
policy.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
{
    var request = new PutUserPolicyRequest
```

```
    {
        UserName = userName,
        PolicyName = policyName,
        PolicyDocument = policyDocument
    };

    var response = await _IAMService.PutUserPolicyAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Wait for a new access key to be ready to use.
/// </summary>
/// <param name="accessKeyId">The Id of the access key.</param>
/// <returns>A boolean value indicating the success of the action.</returns>
public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
{
    var keyReady = false;

    do
    {
        try
        {
            var response = await _IAMService.GetAccessKeyLastUsedAsync(
                new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
            if (response.UserName is not null)
            {
                keyReady = true;
            }
        }
        catch (NoSuchEntityException)
        {
            keyReady = false;
        }
    } while (!keyReady);

    return keyReady;
}
}
```

```
using Microsoft.Extensions.Configuration;
```

```
namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<IAMWrapper>()
                    .AddTransient<UIWrapper>()
                )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();

        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;

        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();
    }
}
```

```
uiWrapper.DisplayBasicsOverview();
uiWrapper.PressEnter();

// First create a user. By default, the new user has
// no permissions.
uiWrapper.DisplayTitle("Create User");
Console.WriteLine($"Creating a new user with user name: {userName}.");
var user = await iamWrapper.CreateUserAsync(userName);
var userArn = user.Arn;

Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

// Define a role policy document that allows the new user
// to assume the role.
string assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\", " +
    "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": { " +
        "$ \"AWS\": \"{userArn}\" " +
        "}, " +
        "\"Action\": \"sts:AssumeRole\" " +
    "}] " +
    "}";

// Permissions to list all buckets.
string policyDocument = "{" +
    "\"Version\": \"2012-10-17\", " +
    "\"Statement\" : [{" +
        "\"Action\" : [\"s3:ListAllMyBuckets\"], " +
        "\"Effect\" : \"Allow\", " +
        "\"Resource\" : \"*\\" " +
    "}] " +
    "}";

// Create an AccessKey for the user.
uiWrapper.DisplayTitle("Create access key");
Console.WriteLine("Now let's create an access key for the new user.");
var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

var accessKeyId = accessKey.AccessKeyId;
```

```
var secretAccessKey = accessKey.SecretAccessKey;

Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

// Now try listing the Amazon Simple Storage Service (Amazon S3)
// buckets. This should fail at this point because the user doesn't
// have permissions to perform this task.
uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
var buckets = await s3Wrapper.ListMyBucketsAsync();

Console.WriteLine(buckets is null
    ? "As expected, the call to list the buckets has returned a null
list."
    : "Something went wrong. This shouldn't have worked.");

uiWrapper.PressEnter();

uiWrapper.DisplayTitle("Create IAM role");
Console.WriteLine($"Creating the role: {roleName}");

// Creating an IAM role to allow listing the S3 buckets. A role name
// is not case sensitive and must be unique to the account for which it
// is created.
var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

uiWrapper.PressEnter();

// Create a policy with permissions to list S3 buckets.
uiWrapper.DisplayTitle("Create IAM policy");
Console.WriteLine($"Creating the policy: {s3PolicyName}");
```



```
    Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
    var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

    // Wait 15 seconds for the IAM policy to be available.
    uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

    // Attach the policy to the role you created earlier.
    uiWrapper.DisplayTitle("Attach new IAM policy");
    Console.WriteLine("Now let's attach the policy to the role.");
    await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

    // Wait 15 seconds for the role to be updated.
    Console.WriteLine();
    uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

    // Use the AWS Security Token Service (AWS STS) to have the user
    // assume the role we created.
    var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

    // Wait for the new credentials to become valid.
    uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

    var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

    // Try again to list the buckets using the client created with
    // the new user's credentials. This time, it should work.
    var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

    s3Wrapper.UpdateClients(s3Client2, stsClient2);

    buckets = await s3Wrapper.ListMyBucketsAsync();

    uiWrapper.DisplayTitle("List Amazon S3 buckets");
    Console.WriteLine("This time we should have buckets to list.");
    if (buckets is not null)
    {
        buckets.ForEach(bucket =>
        {
            Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
```

```
        });
    }

    uiWrapper.PressEnter();

    // Now clean up all the resources used in the example.
    uiWrapper.DisplayTitle("Clean up resources");
    Console.WriteLine("Thank you for watching. The IAM Basics demo is
complete.");
    Console.WriteLine("Please wait while we clean up the resources we
created.");

    await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

    await iamWrapper.DeletePolicyAsync(policy.Arn);

    await iamWrapper.DeleteRoleAsync(roleName);

    await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

    await iamWrapper.DeleteUserAsync(userName);

    uiWrapper.PressEnter();

    Console.WriteLine("All done cleaning up our resources. Thank you for your
patience.");
    }
}

namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
```

```
/// Constructor for the S3Wrapper class.
/// </summary>
/// <param name="s3Service">An Amazon S3 client object.</param>
/// <param name="stsService">An AWS Security Token Service (AWS STS)
/// client object.</param>
public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
{
    _s3Service = s3Service;
    _stsService = stsService;
}

/// <summary>
/// Assumes an AWS Identity and Access Management (IAM) role that allows
/// Amazon S3 access for the current session.
/// </summary>
/// <param name="roleSession">A string representing the current session.</
param>
/// <param name="roleToAssume">The name of the IAM role to assume.</param>
/// <returns>Credentials for the newly assumed IAM role.</returns>
public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
roleToAssume)
{
    // Create the request to use with the AssumeRoleAsync call.
    var request = new AssumeRoleRequest()
    {
        RoleSessionName = roleSession,
        RoleArn = roleToAssume,
    };

    var response = await _stsService.AssumeRoleAsync(request);

    return response.Credentials;
}

/// <summary>
/// Delete an S3 bucket.
/// </summary>
/// <param name="bucketName">Name of the S3 bucket to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
}
```

```
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
        { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
```

```
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}

namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();
    }
}
```

```
        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
```

```
{
    Console.WriteLine(SepBar);
    Console.WriteLine(CenterString(strTitle));
    Console.WriteLine(SepBar);
}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }

    PressEnter();
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)

- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
# user.
#
# "IAM access" permissions are needed to run this code.
# "STS assume role" permissions are needed to run this code. (Note: It might
# be necessary to
# create a custom policy).
#
# Returns:
# 0 - If successful.
# 1 - If an error occurred.
#####
function iam_create_user_assume_role() {
    {
        if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

            source ./iam_operations.sh
        fi
    }

    echo_repeat "*" 88
    echo "Welcome to the IAM create user and assume role demo."
    echo
    echo "This demo will create an IAM user, create an IAM role, and apply the role
to the user."
```



```
echo_repeat "*" 88
echo

echo -n "Enter a name for a new IAM user: "
get_input
user_name=${get_input_result}

local user_arn
user_arn=$(iam_create_user -u "$user_name")

# shellcheck disable=SC2181
if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
fi

local access_key_response
access_key_response=$(iam_create_user_access_key -u "$user_name")
# shellcheck disable=SC2181
if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
fi

IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
local key_name=${access_key_values[0]}
local key_secret=${access_key_values[1]}

echo "Created access key named $key_name"

echo "Wait 10 seconds for the user to be ready."
sleep 10
echo_repeat "*" 88
echo

local iam_role_name
iam_role_name=$(generate_random_name "test-role")
echo "Creating a role named $iam_role_name with user $user_name as the
principal."
```

```
local assume_role_policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Principal\": {\"AWS\": \"${user_arn}\"},
    \"Action\": \"sts:AssumeRole\"
  }]
}"

local role_arn
role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

# shellcheck disable=SC2181
if [ $? == 0 ]; then
  echo "Created IAM role named $iam_role_name"
else
  errecho "The role failed to create. This demo will exit."
  clean_up "$user_name" "$key_name"
  return 1
fi

local policy_name
policy_name=$(generate_random_name "test-policy")
local policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Action\": \"s3:ListAllMyBuckets\",
    \"Resource\": \"arn:aws:s3:::*\"}]}"

local policy_arn
policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
# shellcheck disable=SC2181
if [[ $? == 0 ]]; then
  echo "Created IAM policy named $policy_name"
else
  errecho "The policy failed to create."
  clean_up "$user_name" "$key_name" "$iam_role_name"
  return 1
fi

if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
  echo "Attached policy $policy_arn to role $iam_role_name"
```

```
else
  errecho "The policy failed to attach."
  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
  return 1
fi

local assume_role_policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Action\": \"sts:AssumeRole\",
    \"Resource\": \"${role_arn}\"}]}"

local assume_role_policy_name
assume_role_policy_name=$(generate_random_name "test-assume-role-")

# shellcheck disable=SC2181
local assume_role_policy_arn
assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
# shellcheck disable=SC2181
if [ $? == 0 ]; then
  echo "Created IAM policy named $assume_role_policy_name for sts assume role"
else
  errecho "The policy failed to create."
  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
  return 1
fi

echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
sleep 10
echo_repeat "*" 88
echo

echo "Try to list buckets without the new user assuming the role."
echo_repeat "*" 88
echo

# Set the environment variables for the created user.
# bashsupport disable=BP2001
export AWS_ACCESS_KEY_ID=$key_name
# bashsupport disable=BP2001
```

```
export AWS_SECRET_ACCESS_KEY=$key_secret

local buckets
buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
else
    errecho "Because the role with permissions has not been assumed, listing
buckets failed."
fi

echo
echo_repeat "*" 88
echo "Now assume the role $iam_role_name and list the buckets."
echo_repeat "*" 88
echo

local credentials

credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
else
    errecho "Failed to assume role."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
    return 1
fi

IFS=$'\t ' read -r -a credentials <<<"$credentials"

export AWS_ACCESS_KEY_ID=${credentials[0]}
export AWS_SECRET_ACCESS_KEY=${credentials[1]}
# bashsupport disable=BP2001
export AWS_SESSION_TOKEN=${credentials[2]}
```

```
buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
succeeded because of "
    echo "the assumed role."
else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
    return 1
fi

local result=0
export AWS_ACCESS_KEY_ID=""
export AWS_SECRET_ACCESS_KEY=""

echo
echo_repeat "*" 88
echo "The created resources will now be deleted."
echo_repeat "*" 88
echo

clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    result=1
fi

return $result
}
```

このシナリオで使用される IAM 関数。

```
#####  
# function iam_user_exists  
#  
# This function checks to see if the specified AWS Identity and Access Management  
# (IAM) user already exists.  
#  
# Parameters:  
#     $1 - The name of the IAM user to check.  
#  
# Returns:  
#     0 - If the user already exists.  
#     1 - If the user doesn't exist.  
#####  
function iam_user_exists() {  
    local user_name  
    user_name=$1  
  
    # Check whether the IAM user already exists.  
    # We suppress all output - we're interested only in the return code.  
  
    local errors  
    errors=$(aws iam get-user \  
        --user-name "$user_name" 2>&1 >/dev/null)  
  
    local error_code=${?}  
  
    if [[ $error_code -eq 0 ]]; then  
        return 0 # 0 in Bash script means true.  
    else  
        if [[ $errors != *"error"*(NoSuchEntity)* ]]; then  
            aws_cli_error_log $error_code  
            errecho "Error calling iam get-user $errors"  
        fi  
  
        return 1 # 1 in Bash script means false.  
    fi  
}  
  
#####  
# function iam_create_user  
#  
# This function creates the specified IAM user, unless  
# it already exists.
```

```
#
# Parameters:
#     -u user_name  -- The name of the user to create.
#
# Returns:
#     The ARN of the user.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user() {
    local user_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user"
        echo "Creates an WS Identity and Access Management (IAM) user. You must
supply a username:"
        echo "  -u user_name    The name of the user. It must be unique within the
account."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$user_name" ]]; then
        errecho "ERROR: You must provide a username with the -u parameter."
        usage
    fi
}
```

```
    return 1
fi

iecho "Parameters:\n"
iecho "    User name:  $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#     -u user_name -- The name of the IAM user.
#     [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#     [access_key_id access_key_secret]
#
# And:
#     0 - If successful.
#     1 - If it fails.
```



```
#####
function iam_create_user_access_key() {
    local user_name file_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) key pair."
        echo "  -u user_name    The name of the IAM user."
        echo "  [-f file_name]  Optional file name for the access key output."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:f:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            f) file_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$user_name" ]]; then
        errecho "ERROR: You must provide a username with the -u parameter."
        usage
        return 1
    fi

    response=$(aws iam create-access-key \
        --user-name "$user_name" \
        --output text)

    local error_code=${?}
}
```

```

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
fi

if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
fi

local key_id key_secret
# shellcheck disable=SC2086
key_id=$(echo $response | cut -f 2 -d ' ')
# shellcheck disable=SC2086
key_secret=$(echo $response | cut -f 4 -d ' ')

echo "$key_id $key_secret"

return 0
}

#####
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_json -- The assume role policy document.
#
# Returns:
#     The ARN of the role.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_role() {
    local role_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) role."
    }

```

```
    echo "  -n role_name    The name of the IAM role."
    echo "  -p policy_json  -- The assume role policy document."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) role_name="${OPTARG}" ;;
        p) policy_document="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
```

```

aws_cli_error_log $error_code
errecho "ERROR: AWS reports create-role operation failed.\n$response"
return 1
fi

echo "$response"

return 0
}

#####
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#     -n policy_name -- The name of the IAM policy.
#     -p policy_json -- The policy document.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_policy() {
    local policy_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_policy"
        echo "Creates an AWS Identity and Access Management (IAM) policy."
        echo "  -n policy_name  The name of the IAM policy."
        echo "  -p policy_json -- The policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) policy_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}

```

```
;;
\?)
    echo "Invalid parameter"
    usage
    return 1
;;
esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
fi

echo "$response"
}

#####
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a role.
#
```

```

# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_attach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_attach_role_policy"
        echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name    The name of the IAM role."
        echo "  -p policy_ARN -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$role_name" ]]; then
        errecho "ERROR: You must provide a role name with the -n parameter."
        usage
        return 1
    fi
}

```

```

fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_detach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {

```

```
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name    The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) role_name="${OPTARG}" ;;
        p) policy_arn="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}
```



```
if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#     -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_policy() {
    local policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_policy"
        echo "Deletes an WS Identity and Access Management (IAM) policy"
        echo "  -n policy_arn -- The name of the IAM policy arn."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)

```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Policy arn: $policy_arn"
iecho ""

response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
}

#####
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#
# Returns:
```

```

#      0 - If successful.
#      1 - If it fails.
#####
function iam_delete_role() {
    local role_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_role"
        echo "Deletes an WS Identity and Access Management (IAM) role"
        echo "  -n role_name -- The name of the IAM role."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    echo "role_name:$role_name"
    if [[ -z "$role_name" ]]; then
        errecho "ERROR: You must provide a role name with the -n parameter."
        usage
        return 1
    fi

    iecho "Parameters:\n"
    iecho "  Role name:  $role_name"
    iecho ""

    response=$(aws iam delete-role \

```

```
--role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
fi

iecho "delete-role response:$response"
iecho

return 0
}

#####
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user.
#     -k access_key -- The access key to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_access_key() {
    local user_name access_key response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_access_key"
        echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
        echo "  -u user_name    The name of the user."
        echo "  -k access_key   The access key to delete."
        echo ""
    }
}

# Retrieve the calling parameters.
```

```
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "  Username:  $user_name"
iecho "  Access key: $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi
```

```
fi

iecho "delete-access-key response:$response"
iecho

return 0
}

#####
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user to create.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_user() {
    local user_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_user"
        echo "Deletes an WS Identity and Access Management (IAM) user. You must
supply a username:"
        echo "  -u user_name    The name of the user."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage

```

```
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    User name:  $user_name"
iecho ""

# If the user does not exist, we don't want to try to delete it.
if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
fi

response=$(aws iam delete-user \
    --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
fi

iecho "delete-user response:$response"
iecho

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」で以下のトピックを参照してください。
- [AttachRolePolicy](#)

- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
         \sa DeleteCreatedEntities
         \param client: IAM client.
         \param role: IAM role.
         \param user: IAM user.
         \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy);
    }
}
```



```
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
    user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
    necessary to
//     create a custom policy).
/*!
    \sa iamCreateUserAssumeRoleScenario
    \param clientConfig: Aws client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
std::endl;
        }
    }
}
```

```
    user = outcome.GetResult().GetUser();
}

// 2. Create a role.
{
    // Get the IAM user for the current client in order to access its ARN.
    Aws::String iamUserArn;
    {
        Aws::IAM::Model::GetUserRequest request;
        Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error getting Iam user. " <<
                outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully retrieved Iam user "
                << outcome.GetResult().GetUser().GetUserName()
                << std::endl;
        }

        iamUserArn = outcome.GetResult().GetUser().GetArn();
    }

    Aws::IAM::Model::CreateRoleRequest request;

    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleName = "iam-demo-role-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleName(roleName);

    // Build policy document for role.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");

    Aws::Utils::Document jsonPrincipal;
    jsonPrincipal.WithString("AWS", iamUserArn);
    jsonStatement.WithObject("Principal", jsonPrincipal);
    jsonStatement.WithString("Action", "sts:AssumeRole");
    jsonStatement.WithObject("Condition", Aws::Utils::Document());
}
```

```
Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Setting policy for role\n  "
           << policyDocument.View().WriteCompact() << std::endl;

// Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
              outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a role with name " << roleName
              << std::endl;
}

role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3::*");
```

```
Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Creating a policy.\n  " <<
policyDocument.View().WriteCompact()
    << std::endl;

// Set IAM policy document as JSON string.
request.SetPolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating policy. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a policy with name, " <<
policyName <<
        "." << std::endl;
}

policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSCClient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtil::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);
```

```
Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

// Repeatedly call AssumeRole, because there is often a delay
// before the role is available to be assumed.
// Repeat at most 20 times when access is denied.
int count = 0;
while (true) {
    assumeRoleOutcome = stsClient.AssumeRole(request);
    if (!assumeRoleOutcome.IsSuccess()) {
        if (count > 20 ||
            assumeRoleOutcome.GetError().GetErrorType() !=
            Aws::STS::STSErrors::ACCESS_DENIED) {
            std::cerr << "Error assuming role after 20 tries. " <<
                assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    else {
        std::cout << "Successfully assumed the role after " << count
            << " seconds." << std::endl;
        break;
    }
    count++;
}

credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
```

```
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                    listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                    << "Access to list buckets denied because privileges have
not been applied."
                    << std::endl;
            }
        }
        else {
            std::cerr
                << "Successfully retrieved bucket lists when this should not
happen."
                << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
            request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                << policy.GetPolicyName() <<
                ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }
}
```

```
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
    role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
            Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                       credentials.GetSecretAccessKey(),
                                       credentials.GetSessionToken()),
            Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
            clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                listBucketsOutcome.GetError().GetMessage() <<
std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            std::cout << "Successfully retrieved bucket lists after " << count
                << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}
```

```
bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy) {

    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
            request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                    << policy.GetArn()
                    << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                    << policy.GetArn() << std::endl;
            }
        }
    }
}
```




```
    }  
  
    }  
  
    if (role.RoleIdHasBeenSet()) {  
        // Delete the role.  
        Aws::IAM::Model::DeleteRoleRequest request;  
        request.SetRoleName(role.GetRoleName());  
  
        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);  
        if (!outcome.IsSuccess()) {  
            std::cerr << "Error deleting role. " <<  
                outcome.GetError().GetMessage() << std::endl;  
            result = false;  
        }  
        else {  
            std::cout << "Successfully deleted the role with name "  
                << role.GetRoleName() << std::endl;  
        }  
    }  
    }  
  
    if (user.ArnHasBeenSet()) {  
        // Delete the user.  
        Aws::IAM::Model::DeleteUserRequest request;  
        request.WithUserName(user.GetUserName());  
  
        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);  
        if (!outcome.IsSuccess()) {  
            std::cerr << "Error deleting user. " <<  
                outcome.GetError().GetMessage() << std::endl;  
            result = false;  
        }  
        else {  
            std::cout << "Successfully deleted the user with name "  
                << user.GetUserName() << std::endl;  
        }  
    }  
    }  
  
    return result;  
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

Go

SDK for Go V2

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
// (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
```

```
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
    sdkConfig aws.Config
    accountWrapper actions.AccountWrapper
    policyWrapper actions.PolicyWrapper
    roleWrapper actions.RoleWrapper
    userWrapper actions.UserWrapper
    questioner demotools.IQuestioner
    helper IScenarioHelper
    isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
// configuration.
// It uses the specified config to get an IAM client and create wrappers for the
// actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner,
    helper IScenarioHelper) AssumeRoleScenario {
    iamClient := iam.NewFromConfig(sdkConfig)
    return AssumeRoleScenario{
        sdkConfig:    sdkConfig,
        accountWrapper: actions.AccountWrapper{IamClient: iamClient},
        policyWrapper: actions.PolicyWrapper{IamClient: iamClient},
        roleWrapper:   actions.RoleWrapper{IamClient: iamClient},
        userWrapper:  actions.UserWrapper{IamClient: iamClient},
        questioner:   questioner,
        helper:       helper,
    }
}

// addTestOptions appends the API options specified in the original configuration
// to
// another configuration. This is used to attach the middleware stubber to
// clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
    if scenario.isTestRun {
        scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
            scenario.sdkConfig.APIOptions...)
    }
}
```

```
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong with the demo.\n")
            log.Println(r)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role demo.")
    log.Println(strings.Repeat("-", 88))

    user := scenario.CreateUser()
    accessKey := scenario.CreateAccessKey(user)
    role := scenario.CreateRoleAndPolicies(user)
    noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
    scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
    scenario.Cleanup(user, role)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
    log.Println("Let's create an example user with no permissions.")
    userName := scenario.questioner.Ask("Enter a name for the example user:", demotools.NotEmpty{})
    user, err := scenario.userWrapper.GetUser(userName)
    if err != nil {
        panic(err)
    }
    if user == nil {
        user, err = scenario.userWrapper.CreateUser(userName)
        if err != nil {
            panic(err)
        }
        log.Printf("Created user %v.\n", *user.UserName)
    } else {
```

```
    log.Printf("User %v already exists.\n", *user.UserName)
}
log.Println(strings.Repeat("-", 88))
return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
    *types.AccessKey {
    accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
    if err != nil {
        panic(err)
    }
    log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
    log.Println("Waiting a few seconds for your user to be ready...")
    scenario.helper.Pause(10)
    log.Println(strings.Repeat("-", 88))
    return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
// buckets for
// the current account and attaches the policy to a newly created role. It also
// adds an
// inline policy to the specified user that grants the user permission to assume
// the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
    *types.Role {
    log.Println("Let's create a role and policy that grant permission to list S3
    buckets.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    listBucketsRole, err :=
    scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
    if err != nil {panic(err)}
    log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
    listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
        scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
    if err != nil {panic(err)}
    log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
    err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
        *listBucketsRole.RoleName)
    if err != nil {panic(err)}
    log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
        *listBucketsRole.RoleName)
```

```
err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
scenario.helper.GetName(),
[]string{"sts:AssumeRole"}, *listBucketsRole.Arn)
if err != nil {panic(err)}
log.Printf("Created an inline policy for user %v that lets the user assume the
role.\n",
*user.UserName)
log.Println("Let's give AWS a few seconds to propagate these new resources and
connections...")
scenario.helper.Pause(10)
log.Println(strings.Repeat("-", 88))
return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
access key
// credentials and tries to list buckets for the account. Because the user does
not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
*types.AccessKey) *aws.Config {
log.Println("Let's try to list buckets without permissions. This should return
an AccessDenied error.")
scenario.questioner.Ask("Press Enter when you're ready.")
noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
*accessKey.AccessKeyId, *accessKey.SecretAccessKey, "")),
))
if err != nil {panic(err)}

// Add test options if this is a test run. This is needed only for testing
purposes.
scenario.addTestOptions(&noPermsConfig)

s3Client := s3.NewFromConfig(noPermsConfig)
_, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
if err != nil {
// The SDK for Go does not model the AccessDenied error, so check ErrorCode
directly.
var ae smithy.APIError
if errors.As(err, &ae) {
switch ae.ErrorCode() {
case "AccessDenied":
```

```
    log.Println("Got AccessDenied error, which is the expected result because\n"
+
    "the ListBuckets call was made without permissions.")
    default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
    }
    }
    } else {
    log.Println("Expected AccessDenied error when calling ListBuckets without
permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
    }
    log.Println(strings.Repeat("-", 88))
    return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
//    created from
//    the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
//    list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
//    generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
*aws.Config, role *types.Role) {
    log.Println("Let's assume the role that grants permission to list buckets and
try again.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    stsClient := sts.NewFromConfig(*noPermsConfig)
    tempCredentials, err := stsClient.AssumeRole(context.TODO(),
&sts.AssumeRoleInput{
        RoleArn:          role.Arn,
        RoleSessionName:  aws.String("AssumeRoleExampleSession"),
        DurationSeconds:  aws.Int32(900),
    })
    if err != nil {
        log.Printf("Couldn't assume role %v.\n", *role.RoleName)
        panic(err)
    }
}
```

```
}
log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
    config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
        *tempCredentials.Credentials.AccessKeyId,
        *tempCredentials.Credentials.SecretAccessKey,
        *tempCredentials.Credentials.SessionToken),
    ),
)
if err != nil {panic(err)}

// Add test options if this is a test run. This is needed only for testing
// purposes.
scenario.addTestOptions(&assumeRoleConfig)

s3Client := s3.NewFromConfig(assumeRoleConfig)
result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
if err != nil {
    log.Println("Couldn't list buckets with assumed role credentials.")
    panic(err)
}
log.Println("Successfully called ListBuckets with assumed role credentials, \n"
+
    "here are some of them:")
for i := 0; i < len(result.Buckets) && i < 5; i++ {
    log.Printf("\t%v\n", *result.Buckets[i].Name)
}
log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
    if scenario.questioner.AskBool(
        "Do you want to delete the resources created for this example? (y/n)", "y",
    ) {
        policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
        if err != nil {panic(err)}
        for _, policy := range policies {
            err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
                *policy.PolicyArn)
            if err != nil {panic(err)}
            err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
            if err != nil {panic(err)}
            log.Printf("Detached policy %v from role %v and deleted the policy.\n",
```



```
    *policy.PolicyName, *role.RoleName)
}
err = scenario.roleWrapper.DeleteRole(*role.RoleName)
if err != nil {panic(err)}
log.Printf("Deleted role %v.\n", *role.RoleName)

userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
if err != nil {panic(err)}
for _, userPol := range userPols {
    err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
    if err != nil {panic(err)}
    log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
}
keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
if err != nil {panic(err)}
for _, key := range keys {
    err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
    if err != nil {panic(err)}
    log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
*user.UserName)
}
err = scenario.userWrapper.DeleteUser(*user.UserName)
if err != nil {panic(err)}
log.Printf("Deleted user %v.\n", *user.UserName)
log.Println(strings.Repeat("-", 88))
}
}
```

アカウントアクションをラップする構造体を定義します。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
    iamClient *iam.Client
}
```

```
// GetAccountPasswordPolicy gets the account password policy for the current
// account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
error) {
    var pwPolicy *types.PasswordPolicy
    result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
        &iam.GetAccountPasswordPolicyInput{})
    if err != nil {
        log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
    } else {
        pwPolicy = result.PasswordPolicy
    }
    return pwPolicy, err
}

// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
error) {
    var providers []types.SAMLProviderListEntry
    result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
        &iam.ListSAMLProvidersInput{})
    if err != nil {
        log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
    } else {
        providers = result.SAMLProviderList
    }
    return providers, err
}
```

ポリシーアクションをラップする構造体を定義します。

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version string
    Statement []PolicyStatement
}
```

```
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect string
    Action []string
    Principal map[string]string `json:",omitempty"`
    Resource *string `json:",omitempty"`
}

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    IamClient *iam.Client
}

// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
error) {
    var policies []types.Policy
    result, err := wrapper.IamClient.ListPolicies(context.TODO(),
&iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPolicies),
    })
    if err != nil {
        log.Printf("Couldn't list policies. Here's why: %v\n", err)
    } else {
        policies = result.Policies
    }
    return policies, err
}

// CreatePolicy creates a policy that grants a list of actions to the specified
resource.
// PolicyDocument shows how to work with a policy document as a data structure
and
```

```
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
    resourceArn string) (*types.Policy, error) {
    var policy *types.Policy
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Action: actions,
            Resource: aws.String(resourceArn),
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
            resourceArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
        &iam.CreatePolicyInput{
            PolicyDocument: aws.String(string(policyBytes)),
            PolicyName: aws.String(policyName),
        })
    if err != nil {
        log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}

// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
    var policy *types.Policy
    result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
    } else {
        policy = result.Policy
    }
}
```

```
    return policy, err
}

// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
    _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
    }
    return err
}
```

ロールアクションをラップする構造体を定義します。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    IamClient *iam.Client
}

// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
    var roles []types.Role
    result, err := wrapper.IamClient.ListRoles(context.TODO(),
        &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
    )
    if err != nil {
        log.Printf("Couldn't list roles. Here's why: %v\n", err)
    } else {
        roles = result.Roles
    }
    return roles, err
}
```

```
// CreateRole creates a role that trusts a specified user. The trusted user can
// assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
(*types.Role, error) {
    var role *types.Role
    trustPolicy := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Principal: map[string]string{"AWS": trustedUserArn},
            Action: []string{"sts:AssumeRole"},
        }},
    }
    policyBytes, err := json.Marshal(trustPolicy)
    if err != nil {
        log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
            trustedUserArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreateRole(context.TODO(),
        &iam.CreateRoleInput{
            AssumeRolePolicyDocument: aws.String(string(policyBytes)),
            RoleName:                  aws.String(roleName),
        })
    if err != nil {
        log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
    var role *types.Role
```

```
result, err := wrapper.IamClient.GetRole(context.TODO(),
    &iam.GetRoleInput{RoleName: aws.String(roleName)})
if err != nil {
    log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
} else {
    role = result.Role
}
return role, err
}

// CreateServiceLinkedRole creates a service-linked role that is owned by the
// specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
    description string) (*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
    &iam.CreateServiceLinkedRoleInput{
        AWSServiceName: aws.String(serviceName),
        Description:     aws.String(description),
    })
    if err != nil {
        log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
            serviceName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
    &iam.DeleteServiceLinkedRoleInput{
        RoleName: aws.String(roleName)},
    )
    if err != nil {
        log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
            roleName, err)
    }
    return err
}
```

```
}

// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
error {
    _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
&iam.AttachRolePolicyInput{
    PolicyArn: aws.String(policyArn),
    RoleName:  aws.String(roleName),
})
    if err != nil {
        log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
roleName, err)
    }
    return err
}

// ListAttachedRolePolicies lists the policies that are attached to the specified
role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
([]types.AttachedPolicy, error) {
    var policies []types.AttachedPolicy
    result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
&iam.ListAttachedRolePoliciesInput{
    RoleName: aws.String(roleName),
})
    if err != nil {
        log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
roleName, err)
    } else {
        policies = result.AttachedPolicies
    }
    return policies, err
}

// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
error {
```



```
_, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
&iam.DetachRolePolicyInput{
    PolicyArn: aws.String(policyArn),
    RoleName:  aws.String(roleName),
})
if err != nil {
    log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
err)
}
return err
}

// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
&iam.ListRolePoliciesInput{
    RoleName: aws.String(roleName),
})
if err != nil {
    log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
err)
} else {
    policies = result.PolicyNames
}
return policies, err
}

// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
    _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
    RoleName: aws.String(roleName),
})
if err != nil {
    log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
}
return err
}
```

ユーザーアクションをラップする構造体を定義します。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
    var users []types.User
    result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
        MaxItems: aws.Int32(maxUsers),
    })
    if err != nil {
        log.Printf("Couldn't list users. Here's why: %v\n", err)
    } else {
        users = result.Users
    }
    return users, err
}

// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NoSuchEntityException:
                log.Printf("User %v does not exist.\n", userName)
                err = nil
            }
        }
    }
}
```

```
    default:
        log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
    }
} else {
    user = result.User
}
return user, err
}

// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.CreateUser(context.TODO(),
        &iam.CreateUserInput{
            UserName: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    } else {
        user = result.User
    }
    return user, err
}

// CreateUserPolicy adds an inline policy to a user. This example creates a
// policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
// and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
    actions []string,
    roleArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Action: actions,
            Resource: aws.String(roleArn),
```

```
    }},
  }
  policyBytes, err := json.Marshal(policyDoc)
  if err != nil {
    log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
err)
    return err
  }
  _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
&iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
})
  if err != nil {
    log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
err)
  }
  return err
}

// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
  var policies []string
  result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
&iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
})
  if err != nil {
    log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
err)
  } else {
    policies = result.PolicyNames
  }
  return policies, err
}

// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
error {
```

```
_, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
&iam.DeleteUserPolicyInput{
    PolicyName: aws.String(policyName),
    UserName:   aws.String(userName),
})
if err != nil {
    log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
err)
}
return err
}

// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
_, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
    UserName: aws.String(userName),
})
if err != nil {
    log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
}
return err
}

// CreateAccessKeyPair creates an access key for a user. The returned access key
contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
(*types.AccessKey, error) {
var key *types.AccessKey
result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
&iam.CreateAccessKeyInput{
    UserName: aws.String(userName)})
if err != nil {
    log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
userName, err)
} else {
    key = result.AccessKey
}
return key, err
}
```

```
// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
    _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
        &iam.DeleteAccessKeyInput{
            AccessKeyId: aws.String(keyId),
            Username:   aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
    }
    return err
}

// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
([]types.AccessKeyMetadata, error) {
    var keys []types.AccessKeyMetadata
    result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
        &iam.ListAccessKeysInput{
            Username: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
            err)
    } else {
        keys = result.AccessKeyMetadata
    }
    return keys, err
}
```

- APIの詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)

- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
/*
To run this Java V2 code example, set up your development environment,
including your credentials.

For information, see this documentation topic:

https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

This example performs these operations:

1. Creates a user that has no permissions.
2. Creates a role and policy that grants Amazon S3 permissions.
3. Creates a role.
4. Grants the user permissions.
```

5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service client object with the temporary credentials.

6. Deletes the resources.

```
*/
```

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\",\" +
        "  \"Statement\": [\" +
        "    {\" +
        "      \"Effect\": \"Allow\",\" +
        "      \"Action\": [\" +
        "        \"s3:*\"\" +
        "      ],\" +
        "      \"Resource\": \"*\"\" +
        "    }\" +
        "  ]\" +
        "};

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = ""

            Usage:
                <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

            Where:
                username - The name of the IAM user to create.\s
                policyName - The name of the policy to create.\s
                roleName - The name of the role to create.\s
                roleSessionName - The name of the session required for the
assumeRole operation.\s
                bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
}

String userName = args[0];
String policyName = args[1];
String roleName = args[2];
String roleSessionName = args[3];
String bucketName = args[4];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IAM example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
User createUser = createIAMUser(iam, userName);

System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"\" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
```

```
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();
```

```
        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static User createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String
json) {

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
```

```
        .description("Created using the AWS SDK for Java")
        .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " +
response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
```

```
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
        String keySecret) {

    // Use the creds of the new IAM user that was created in this code
example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
```

```
        .region(Region.US_EAST_1)

    .credentialsProvider(StaticCredentialsProvider.create(credentials))
      .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
                StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
                secToken)))
            .region(region)
            .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in " + bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }

    } catch (StsException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}

public static void deleteRole(IamClient iam, String roleName, String polArn)
{
    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
        DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String
accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
```

```
        .userName(username)
        .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)



- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
```

```
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
  }
};
```

```
export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
  (AWS STS).
  // It's not best practice to use access keys. For more information, see
  https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });
}
```

```
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
  error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
```

```
        Action: ["s3:ListAllMyBuckets"],
        Resource: "*",
    },
],
}),
PolicyName: policyName,
}),
);

if (!listBucketPolicy) {
    throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
    new AttachRolePolicyCommand({
        PolicyArn: listBucketPolicy.Arn,
        RoleName: Role.RoleName,
    }),
);

// Assume the role.
const stsClient = new STSClient({
    credentials: {
        accessKeyId: AccessKeyId,
        secretAccessKey: SecretAccessKey,
    },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
    { intervalInMs: 2000, maxRetries: 60 },
    () =>
        stsClient.send(
            new AssumeRoleCommand({
                RoleArn: Role.Arn,
                RoleSessionName: `iamBasicScenarioSession-${Math.floor(
                    Math.random() * 1000000,
                )}`,
                DurationSeconds: 900,
            }),
        ),
);
```

```
if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
  }),
);

await iamClient.send(
  new DeleteRoleCommand({
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeleteAccessKeyCommand({
    UserName: userName,
    AccessKeyId,
  }),
);
```

```
);

await iamClient.send(
  new DeleteUserCommand({
    UserName: userName,
  }),
);
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)

- [PutUserPolicy](#)

## Kotlin

### SDK for Kotlin

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
<bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
operation.
        fileLocation - The file location to the JSON required to create the role
(seen Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
```



```

    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "      \"Effect\": \"Allow\"," +
            "      \"Action\": [" +
            "        \"s3:*\"" +
            "      ]," +

```

```
        "        \"Resource\": \"*\")\" +
        \"    }\" +
        \"  ]\" +
        \"}\"

val request =
    CreatePolicyRequest {
        policyName = policyNameVal
        policyDocument = policyDocumentValue
    }

IamClient { region = \"AWS_GLOBAL\" }.use { iamClient ->
    val response = iamClient.createPolicy(request)
    return response.policy?.arn.toString()
}
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = \"Created using the AWS SDK for Kotlin\"
        }

    IamClient { region = \"AWS_GLOBAL\" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }
}
```

```
IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    val response = iamClient.listAttachedRolePolicies(request)
    val attachedPolicies = response.attachedPolicies

    // Ensure that the policy is not attached to this role.
    val checkStatus: Int
    if (attachedPolicies != null) {
        checkStatus = checkMyList(attachedPolicies, policyArnVal)
        if (checkStatus == -1) {
            return
        }
    }

    val policyRequest =
        AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
    iamClient.attachRolePolicy(policyRequest)
    println("Successfully attached policy $policyArnVal to role
    $roleNameVal")
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String
```

```
) {  
    val stsClient =  
        StsClient {  
            region = "us-east-1"  
        }  
  
    val roleRequest =  
        AssumeRoleRequest {  
            roleArn = roleArnVal  
            roleSessionName = roleSessionNameVal  
        }  
  
    val roleResponse = stsClient.assumeRole(roleRequest)  
    val myCreds = roleResponse.credentials  
    val key = myCreds?.accessKeyId  
    val secKey = myCreds?.secretAccessKey  
    val secToken = myCreds?.sessionToken  
  
    val staticCredentials =  
        StaticCredentialsProvider {  
            accessKeyId = key  
            secretAccessKey = secKey  
            sessionToken = secToken  
        }  
  
    // List all objects in an Amazon S3 bucket using the temp creds.  
    val s3 =  
        S3Client {  
            credentialsProvider = staticCredentials  
            region = "us-east-1"  
        }  
  
    println("Created a S3Client using temp credentials.")  
    println("Listing objects in $bucketName")  
  
    val listObjects =  
        ListObjectsRequest {  
            bucket = bucketName  
        }  
  
    val response = s3.listObjects(listObjects)  
    response.contents?.forEach { myObject ->  
        println("The name of the key is ${myObject.key}")  
        println("The owner is ${myObject.owner}")  
    }
```

```
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }
}
```

```
iam.deleteUser(request)
println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("-----\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("-----\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_{$uuid}");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
        \"Action\": \"sts:AssumeRole\"
    }]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_{$uuid}",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_{$uuid}",
    $listAllBucketsPolicyDocument);
```

```
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"sts:AssumeRole\",
        \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_{$uuid}",
    $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_{$uuid}",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
```



```
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError

def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()

def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.
    """
```

```
    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                            that has permissions to create users, roles, and
policies
                            in the account.
:return: The newly created user, user key, and role.
"""
try:
    user = iam_resource.create_user(UserName=f"demo-user-{{uuid4()}}")
    print(f"Created user {user.name}.")
except ClientError as error:
    print(
        f"Couldn't create a user for the demo. Here's why: "
        f"{{error.response['Error']['Message']}}")
    )
    raise

try:
    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")
except ClientError as error:
    print(
        f"Couldn't create access keys for user {user.name}. Here's why: "
        f"{{error.response['Error']['Message']}}")
    )
    raise

print(f"Wait for user to be ready.", end="")
progress_bar(10)

try:
    role = iam_resource.create_role(
        RoleName=f"demo-role-{{uuid4()}}",
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        )
    )
```

```
        ),
    )
    print(f"Created role {role.name}.")
except ClientError as error:
    print(
        f"Couldn't create a role for the demo. Here's why: "
        f"{error.response['Error']['Message']}"
    )
    raise

try:
    policy = iam_resource.create_policy(
        PolicyName=f"demo-policy-{uuid4()}",
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*"
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the
role.")
except ClientError as error:
    print(
        f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
        f"{error.response['Error']['Message']}"
    )
    raise

try:
    user.create_policy(
        PolicyName=f"demo-user-policy-{uuid4()}",
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": role.arn,
        }
    ],
)
)
print(
    f"Created an inline policy for {user.name} that lets the user assume
"
    f"the role."
)
except ClientError as error:
    print(
        f"Couldn't create an inline policy for user {user.name}. Here's why:
"
        f"{error.response['Error']['Message']}"
    )
    raise

print("Give AWS time to propagate these new resources and connections.",
end="")
progress_bar(10)

return user, user_key, role

def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
        have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
```

```
        raise RuntimeError("Expected to get AccessDenied error when listing
buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise

def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
    account.
    Uses the temporary credentials from the role to list the buckets that are
    owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
    role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
    grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
        aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
    credentials.
    s3_resource = boto3.resource(
        "s3",
```

```
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
```

```
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    user.delete()
    print(f"Deleted {user.name}.")
except ClientError as error:
    print(
        "Couldn't delete user policy or delete user. Here's why: "
        f"{error.response['Error']['Message']}"
    )

def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
"AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")

if __name__ == "__main__":
    usage_demo()
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)



- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
```

```
# @param duration [Integer] The number of seconds to wait.
def wait(duration)
  puts("Give AWS time to propagate resources...")
  sleep(duration)
end

# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_client.create_user(user_name: user_name).user
  @logger.info("Created demo user named #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Tried and failed to create demo user.")
  @logger.info("\t#{e.code}: #{e.message}")
  @logger.info("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
  @logger.info("Created accesskey pair for user #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create access keys for user #{user.user_name}.")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
```

```
def create_role(role_name, user)
  trust_policy = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Principal: {'AWS': user.arn},
      Action: "sts:AssumeRole"
    }]
  }.to_json
  role = @iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: trust_policy
  ).role
  @logger.info("Created role #{role.role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create a role for the demo. Here's why: ")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  role
end

# Creates a policy that grants permission to list S3 buckets in the account,
and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy_document = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Action: "s3:ListAllMyBuckets",
      Resource: "arn:aws:s3:::*"
    }]
  }.to_json
  policy = @iam_client.create_policy(
    policy_name: policy_name,
    policy_document: policy_document
  ).policy
  @iam_client.attach_role_policy(
    role_name: role.role_name,
```

```
    policy_arn: policy.arn
  )
  @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
  policy_document = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Action: "sts:AssumeRole",
      Resource: role.arn
    }]
  }.to_json
  @iam_client.put_user_policy(
    user_name: user.user_name,
    policy_name: policy_name,
    policy_document: policy_document
  )
  puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

# Creates an Amazon S3 resource with specified credentials. This is separated
into a
# factory function so that it can be mocked for unit testing.
#
```

```
# @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
def create_s3_resource(credentials)
  Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
end

# Lists the S3 buckets for the account, using the specified Amazon S3 resource.
# Because the resource uses credentials with limited access, it may not be able
to
# list the S3 buckets.
#
# @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
def list_buckets(s3_resource)
  count = 10
  s3_resource.buckets.each do |bucket|
    @logger.info "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
rescue Aws::Errors::ServiceError => e
  if e.code == "AccessDenied"
    puts("Attempt to list buckets with no permissions: AccessDenied.")
  else
    @logger.info("Couldn't list buckets for the account. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end
end
end

# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
```

```
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
# are used.
# @param sts_client [AWS::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end

# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
    @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
    @iam_client.delete_policy(policy_arn: policy.policy_arn)
    @logger.info("Detached and deleted policy #{policy.policy_name}.")
  end
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
```

```
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
end
```

```
puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
$PROGRAM_NAME == __FILE__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。



```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
  as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
    =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [{
            \"Effect\": \"Allow\",
            \"Action\": \"s3:ListAllMyBuckets\",
            \"Resource\": \"arn:aws:s3::*\"}]
    }
```

```

    }"
    .to_string();
    let inline_policy_document = "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [{
            \"Effect\": \"Allow\",
            \"Action\": \"sts:AssumeRole\",
            \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}",
"iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [{
            \"Effect\": \"Allow\",
            \"Principal\": {\"AWS\": \"{}\"},
            \"Action\": \"sts:AssumeRole\"
        }]
    }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}", "iam_demo_role_", uuid),
    )
}

```

```
        &assume_role_policy_document,
    )
    .await?;
println!("Created the role with the ARN: {}", assume_role_role.arn());

let list_all_buckets_policy = iam_service::create_policy(
    &client,
    &format!("{}", "iam_demo_policy_", uuid),
    &list_all_buckets_policy_document,
)
.await?;
println!(
    "Created policy: {}",
    list_all_buckets_policy.policy_name.as_ref().unwrap()
);

let attach_role_policy_result =
    iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
        .await?;
println!(
    "Attached the policy to the role: {:?}",
    attach_role_policy_result
);

let inline_policy_name = format!("{}", "iam_demo_inline_policy_", uuid);
let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
    .await?;
println!("Created inline policy.");

//First, fail to list the buckets with the user.
let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
let fail_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
println!("Fail config: {:?}", fail_config);
let fail_client: s3Client = s3Client::new(&fail_config);
match fail_client.list_buckets().send().await {
    Ok(e) => {
```

```
        println!("This should not run. {:?}", e);
    }
    Err(e) => {
        println!("Successfully failed with error: {:?}", e)
    }
}

let sts_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
let sts_client: stsClient = stsClient::new(&sts_config);
sleep(Duration::from_secs(10)).await;
let assumed_role = sts_client
    .assume_role()
    .role_arn(assume_role_role.arn())
    .role_session_name(&format!("{}", "iam_demo_assumerole_session_",
uuid))
    .send()
    .await;
println!("Assumed role: {:?}", assumed_role);
sleep(Duration::from_secs(10)).await;

let assumed_credentials = iamCredentials::from_keys(
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .access_key_id(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .secret_access_key(),
    Some(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
```

```
        .unwrap()
        .session_token
        .clone(),
    ),
);

let succeed_config = aws_config::from_env()
    .credentials_provider(assumed_credentials)
    .load()
    .await;
println!("succeed config: {:?}", succeed_config);
let succeed_client: s3Client = s3Client::new(&succeed_config);
sleep(Duration::from_secs(10)).await;
match succeed_client.list_buckets().send().await {
    Ok(_) => {
        println!("This should now run successfully.")
    }
    Err(e) => {
        println!("This should not run. {:?}", e);
        panic!()
    }
}

//Clean up.
iam_service::detach_role_policy(
    &client,
    assume_role_role.role_name(),
    list_all_buckets_policy.arn().unwrap_or_default(),
)
.await?;
iam_service::delete_policy(&client, list_all_buckets_policy).await?;
iam_service::delete_role(&client, &assume_role_role).await?;
println!("Deleted role {}", assume_role_role.role_name());
iam_service::delete_access_key(&client, &user, &key).await?;
println!("Deleted key for {}", key.user_name());
iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
println!("Deleted inline user policy: {}", inline_policy_name);
iam_service::delete_user(&client, &user).await?;
println!("Deleted user {}", user.user_name());

Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して読み取り専用 IAM ユーザーおよび読み取り/書き込みできる IAM ユーザーを作成する

次のコード例は、ユーザーを作成し、そのユーザーにポリシーをアタッチする方法を示します。

### Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#) などの ID プロバイダーとのフェデレーションを使用してください。

- IAM ユーザーを 2 つ作成します。
- Amazon S3 バケット内のオブジェクトを取得して格納するポリシーを 1 人のユーザーに割り当てます。

- このバケットからオブジェクトを取得することを許可するポリシーを、セカンドユーザーにアタッチします。
- ユーザーの認証情報に基づいて、バケットについてのさまざまなアクセス許可を取得します。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(Username=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
```

```
    else:
        return user

def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user

def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users

def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
```



```
such as access keys and policies, must be deleted or detached.

:param user_name: The name of the user.
"""
try:
    iam.User(user_name).delete()
    logger.info("Deleted user %s.", user_name)
except ClientError:
    logger.exception("Couldn't delete user %s.", user_name)
    raise

def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
            user_name)
        raise

def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
```

```
raise
```

IAM ポリシーアクションをラップする関数を作成します。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
        form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
        policy
        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all
        objects
        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
```

```
        Description=description,
        PolicyDocument=json.dumps(policy_doc),
    )
    logger.info("Created policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
    raise
else:
    return policy

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

IAM アクセスキーのアクションをラップする関数を作成します。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.
```

```
:param user_name: The name of the user.
:return: The created access key.
"""
try:
    key_pair = iam.User(user_name).create_access_key_pair()
    logger.info(
        "Created access key pair for %s. Key ID is %s.",
        key_pair.user_name,
        key_pair.id,
    )
except ClientError:
    logger.exception("Couldn't create access key pair for %s.", user_name)
    raise
else:
    return key_pair

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """
    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

ラッパー関数を使用して、異なるポリシーを持つユーザーを作成し、その認証情報を使用して Amazon S3 バケットにアクセスします。

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
```

This demonstration creates two users: one user who can put and get objects in an Amazon S3 bucket, and another user who can only get objects from the bucket. The demo then shows how the users can perform only the actions they are permitted

to perform.

```
"""
```

```
logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
print("-" * 88)
print("Welcome to the AWS Identity and Account Management user demo.")
print("-" * 88)
print(
    "Users can have policies and roles attached to grant them specific "
    "permissions."
)
s3 = boto3.resource("s3")
bucket = s3.create_bucket(
    Bucket=f"demo-iam-bucket-{time.time_ns()}",
    CreateBucketConfiguration={
        "LocationConstraint": s3.meta.client.meta.region_name
    },
)
print(f"Created an Amazon S3 bucket named {bucket.name}.")
user_read_writer = create_user("demo-iam-read-writer")
user_reader = create_user("demo-iam-reader")
print(f"Created two IAM users: {user_read_writer.name} and
{user_reader.name}")
update_user(user_read_writer.name, "demo-iam-creator")
update_user(user_reader.name, "demo-iam-getter")
users = list_users()
user_read_writer = next(
    user for user in users if user.user_id == user_read_writer.user_id
)
user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
print(
    f"Changed the names of the users to {user_read_writer.name} "
    f"and {user_reader.name}."
)

read_write_policy = policy_wrapper.create_policy(
    "demo-iam-read-write-policy",
    "Grants rights to create and get an object in the demo bucket.",
    ["s3:PutObject", "s3:GetObject"],
```

```
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{{time.time_ns()}}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
```

```
        raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
```

```
        "allowed to delete objects."
    )
    print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [AttachUserPolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteUser](#)
  - [DetachUserPolicy](#)
  - [ListUsers](#)



- [UpdateUser](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して IAM アクセスキーを管理する

次のコード例は、アクセスキーの管理方法を示しています。

### Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#) などの ID プロバイダーとのフェデレーションを使用してください。

- アクセスキーを作成して一覧表示します。
- アクセスキーが最後に使用された時刻とその方法を検索します。
- アクセスキーの更新や削除を行います。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM アクセスキーのアクションをラップする関数を作成します。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair

def get_last_use(key_id):
```

```
"""
Gets information about when and how a key was last used.

:param key_id: The ID of the key to look up.
:return: Information about the key's last use.
"""
try:
    response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
    last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
    last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
    logger.info(
        "Key %s was last used by %s on %s to access %s.",
        key_id,
        response["UserName"],
        last_used_date,
        last_service,
    )
except ClientError:
    logger.exception("Couldn't get last use of key %s.", key_id)
    raise
else:
    return response

def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """
    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
            key_id)
    except ClientError:
```

```
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
            key_id
        )
        raise

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

ラッパー関数を使用して、現在のユーザーのアクセスキーアクションを実行します。

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
```

```
print(
    f"This demo creates an access key for the current user "
    f"({current_user_name}), manipulates the key in a few ways, and then "
    f"deletes it."
)
all_keys = list_keys(current_user_name)
if len(all_keys) == 2:
    print(
        "The current user already has the maximum of 2 access keys. To run "
        "this demo, either delete one of the access keys or use a user "
        "that has only 1 access key."
    )
else:
    new_key = create_key(current_user_name)
    print(f"Created a new key with id {new_key.id} and secret "
          {new_key.secret}.")
    print_keys()
    existing_key = next(key for key in all_keys if key != new_key)
    last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
    print(
        f"Key {all_keys[0].id} was last used to access "
        {last_use['ServiceName']} "
        f"on {last_use['LastUsedDate']}"
    )
    update_key(current_user_name, new_key.id, False)
    print(f"Key {new_key.id} is now deactivated.")
    print_keys()
    delete_key(current_user_name, new_key.id)
    print_keys()
    print("Thanks for watching!")
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CreateAccessKey](#)
  - [DeleteAccessKey](#)
  - [GetAccessKeyLastUsed](#)
  - [ListAccessKeys](#)
  - [UpdateAccessKey](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して IAM ポリシーを管理する

次のコードサンプルは、以下の操作方法を示しています。

- ポリシーを作成して一覧表示します。
- ポリシーバージョンを作成して取得します。
- ポリシーを以前のバージョンにロールバックします。
- ポリシーを削除します。

### Python

#### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ポリシーアクションをラップする関数を作成します。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.
```

```
:param name: The name of the policy to create.
:param description: The description of the policy.
:param actions: The actions allowed by the policy. These typically take the
                 form of service:action, such as s3:PutObject.
:param resource_arn: The Amazon Resource Name (ARN) of the resource this
policy
                    applies to. This ARN can contain wildcards, such as
                    'arn:aws:s3::my-bucket/*' to allow actions on all
objects
                    in the bucket named 'my-bucket'.
:return: The newly created policy.
"""
policy_doc = {
    "Version": "2012-10-17",
    "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
}
try:
    policy = iam.create_policy(
        PolicyName=name,
        Description=description,
        PolicyDocument=json.dumps(policy_doc),
    )
    logger.info("Created policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
    raise
else:
    return policy

def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
```

```
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies

def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
        version for the policy. Otherwise, the default
        is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```



```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement

def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
```

```
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
            rollback_version = policy_versions.pop()
            rollback_version.set_as_default()
            logger.info("Set %s as the default version.",
rollback_version.version_id)
            default_version.delete()
            logger.info("Deleted original default version %s.",
default_version.version_id)
        except IndexError:
            if default_version is None:
                logger.warning("No default version found for %s.", policy_arn)
            elif rollback_version is None:
                logger.warning(
so "                "Default version %s found for %s, but no previous version exists,
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
        except ClientError:
            logger.exception("Couldn't roll back version for %s.", policy_arn)
            raise
        else:
            return rollback_version

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
```

```
except ClientError:
    logger.exception("Couldn't delete policy %s.", policy_arn)
    raise
```

ラッパー関数を使用して、ポリシーを作成し、バージョンを更新し、それらに関する情報を取得します。

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::made-up-bucket-name"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
```

```
print(
    f"Rolled back to version {rollback_version.version_id} for "
    f"{policy.policy_name}."
)
default_statement = get_default_policy_statement(policy.arn)
print(f"The default policy statement for {policy.policy_name} is now:")
pprint.pprint(default_statement)
delete_policy(policy.arn)
print(f"Deleted policy {policy.policy_name}.")
print("Thanks for watching!")
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CreatePolicy](#)
  - [CreatePolicyVersion](#)
  - [DeletePolicy](#)
  - [DeletePolicyVersion](#)
  - [GetPolicyVersion](#)
  - [ListPolicies](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して IAM ロールを管理する

次のコードサンプルは、以下の操作方法を示しています。

- IAM ロールを作成します。
- ロールにポリシーをアタッチおよびデタッチします
- ロールを削除します。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ロールアクションをラップする関数を作成します。

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
```

```
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role

def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. Note this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
            role_name)
        raise

def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. Note this is the name, not the
    ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
```

```
    )
    raise

def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

ラッパー関数を使用してロールを作成し、ポリシーをアタッチおよびデタッチします。

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
"batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
```

```
policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
attach_policy(role.name, policy_arn)
print(f"Attached policy {policy_arn} to {role.name}.")
print(f"Policies attached to role {role.name} are:")
list_attached_policies(role.name)
detach_policy(role.name, policy_arn)
print(f"Detached policy {policy_arn} from {role.name}.")
delete_role(role.name)
print(f"Deleted {role.name}.")
print("Thanks for watching!")
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateRole](#)
  - [DeleteRole](#)
  - [DetachRolePolicy](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して IAM アカウントを管理する

次のコードサンプルは、以下の操作方法を示しています。

- アカウントエイリアスを取得して更新します。
- ユーザーと認証情報に関するレポートを生成します。
- アカウントの使用状況の概要を取得します。
- アカウント内のユーザー、グループ、ロール、ポリシーについて、相互の関連付けなどの詳細を取得します。



## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM アカウントのアクションをラップする関数を作成します。

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
    the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise

def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise

def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
    hours
    after the last one was generated.
    """
    try:
```

```
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
%s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
account.")
        raise
    else:
        return response

def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]

def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
```

```
        raise
    else:
        return summary.summary_map

def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
    such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

ラッパー関数を呼び出して、アカウントのエイリアスを変更し、アカウントに関するレポートを取得します。

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
```

```
old_aliases = list_aliases()
if len(old_aliases) > 0:
    print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
else:
    print("Your account currently has no alias.")
for index in range(1, 3):
    new_alias = f"alias-{index}-{time.time_ns()}"
    print(f"Setting your account alias to {new_alias}")
    create_alias(new_alias)
current_aliases = list_aliases()
print(f"Your account alias is now {current_aliases}.")
delete_alias(current_aliases[0])
print(f"Your account now has no alias.")
if len(old_aliases) > 0:
    print(f"Restoring your original alias back to {old_aliases[0]}...")
    create_alias(old_aliases[0])

print("-" * 88)
print("You can get various reports about your account.")
print("Let's generate a credentials report...")
report_state = None
while report_state != "COMPLETE":
    cred_report_response = generate_credential_report()
    old_report_state = report_state
    report_state = cred_report_response["State"]
    if report_state != old_report_state:
        print(report_state, sep="")
    else:
        print(".", sep="")
    sys.stdout.flush()
    time.sleep(1)
print()
cred_report = get_credential_report()
col_count = 3
print(f"Got credentials report. Showing only the first {col_count} columns.")
cred_lines = [
    line.split(",")[:col_count] for line in
cred_report.decode("utf-8").split("\n")
]
col_width = max([len(item) for line in cred_lines for item in line]) + 2
for line in cred_report.decode("utf-8").split("\n"):
    print(
        "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
```

```
)

print("-" * 88)
print("Let's get an account summary.")
summary = get_summary()
print("Here's your summary:")
pprint.pprint(summary)

print("-" * 88)
print("Let's get authorization details!")
details = get_authorization_details([])
see_details = input("These are pretty long, do you want to see them (y/n)? ")
if see_details.lower() == "y":
    pprint.pprint(details)

print("-" * 88)
pw_policy_created = None
see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
if see_pw_policy.lower() == "y":
    while True:
        if print_password_policy():
            break
        else:
            answer = input(
                "Do you want to create a default password policy (y/n)? "
            )
            if answer.lower() == "y":
                pw_policy_created = iam.create_account_password_policy()
            else:
                break
if pw_policy_created is not None:
    answer = input("Do you want to delete the password policy (y/n)? ")
    if answer.lower() == "y":
        pw_policy_created.delete()
        print("Password policy deleted.")

print("The SAML providers for your account are:")
list_saml_providers(10)

print("-" * 88)
print("Thanks for watching.")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CreateAccountAlias](#)
  - [DeleteAccountAlias](#)
  - [GenerateCredentialReport](#)
  - [GetAccountAuthorizationDetails](#)
  - [GetAccountSummary](#)
  - [GetCredentialReport](#)
  - [ListAccountAliases](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して IAM ポリシーのバージョンをロールバックする

次のコードサンプルは、以下の操作方法を示しています。

- ポリシーのバージョンの日付順リストを取得します。
- デフォルトのポリシーバージョンを検索します。
- デフォルト値を以前のポリシーバージョンにします。
- 古いデフォルトバージョンを削除します。

### Python

#### SDK for Python (Boto3)

##### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
def rollback_policy_version(policy_arn):
```

```
"""
Rolls back to the previous default policy, if it exists.

1. Gets the list of policy versions in order by date.
2. Finds the default.
3. Makes the previous policy the default.
4. Deletes the old default version.

:param policy_arn: The ARN of the policy to roll back.
:return: The default version of the policy after the rollback.
"""
try:
    policy_versions = sorted(
        iam.Policy(policy_arn).versions.all(),
        key=operator.attrgetter("create_date"),
    )
    logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
except ClientError:
    logger.exception("Couldn't get versions for %s.", policy_arn)
    raise

default_version = None
rollback_version = None
try:
    while default_version is None:
        ver = policy_versions.pop()
        if ver.is_default_version:
            default_version = ver
    rollback_version = policy_versions.pop()
    rollback_version.set_as_default()
    logger.info("Set %s as the default version.",
rollback_version.version_id)
    default_version.delete()
    logger.info("Deleted original default version %s.",
default_version.version_id)
except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists,
so "
            "nothing to roll back to.",
            default_version.version_id,
```



```
        policy_arn,
    )
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [DeletePolicyVersion](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK での IAM Policy Builder API の使用

次のコードサンプルは、以下の操作方法を示しています。

- オブジェクト指向の API を使用して IAM ポリシーを作成します。
- IAM サービスで IAM Policy Builder API を使用します。

### Java

#### SDK for Java 2.x

##### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

例では、次の入力を使用します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

時間ベースのポリシーを作成します。

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_GREATER_THAN)

        .key("aws:CurrentTime")

        .value("2020-04-01T00:00:00Z"))
        .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_LESS_THAN)

        .key("aws:CurrentTime")
```

```

        .value("2020-06-30T23:59:59Z")))
            .build();

        // Use an IamPolicyWriter to write out the JSON string to a more
readable
        // format.
        return policy.toJson(IamPolicyWriter.builder()
            .prettyPrint(true)
            .build());
    }

```

複数の条件を含むポリシーを作成します。

```

public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")

.addAction("dynamodb:BatchGetItem")

            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb>DeleteItem")

.addAction("dynamodb:BatchWriteItem")

.addAction("arn:aws:dynamodb:*:*:table/table-name")

.addConditions(IamConditionOperator.STRING_EQUALS

.addPrefix("ForAllValues:"),

"dynamodb:Attributes",

List.of("column-
name1", "column-name2", "column-name3"))

.addCondition(b1 -> b1

.operator(IamConditionOperator.STRING_EQUALS

.addSuffix("IfExists"))

```

```
.key("dynamodb:Select")

.value("SPECIFIC_ATTRIBUTES"))
        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

ポリシーにプリンシパルを使用します。

```
public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)

        .addResource("arn:aws:s3:::BUCKETNAME/*")

        .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.ARN_NOT_EQUALS)

        .key("aws:PrincipalArn")

        .value("arn:aws:iam::444455556666:user/user-name"))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

クロスアカウントのアクセスを許可します。

```
public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
```

```

    .addPrincipal(IamPrincipalType.AWS, "111122223333")
        .addAction("s3:PutObject")
        .addResource("arn:aws:s3::DOC-
EXAMPLE-BUCKET/*")
        .addCondition(b1 -> b1

    .operator(IamConditionOperator.STRING_EQUALS)
        .key("s3:x-amz-
acl")
        .value("bucket-
owner-full-control"))))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

IamPolicy を作成してアップロードします。

```

    public String createAndUploadPolicyExample(IamClient iam, String
accountID, String policyName) {
        // Build the policy.
        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
"2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")

            .addResource("arn:aws:dynamodb:us-east-1:" + accountID
                + ":table/
exampleTableName")
                .build())
            .build();
        // Upload the policy.
        iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
        return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }

```

IamPolicy をダウンロードして使用します。

```
public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
            String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion =
getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse = iam
        .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned
from IAM.
    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
        StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
    * All IamPolicy components are immutable, so use the copy method
that creates a
    * new instance that
    * can be altered in the same method call.
    *
    * Add the ability to get an item from DynamoDB as an additional
action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));
}
```

```
        return
        newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }
```

- 詳細については、「[AWS SDK for Java 2.x デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreatePolicy](#)
  - [GetPolicy](#)
  - [GetPolicyVersion](#)

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した AWS STS のコード例

以下は、AWS STS Software Development Kit (SDK) で AWS を使用する方法を説明するコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### コードの例

- [AWS SDK を使用した AWS STS 向けアクション](#)
  - [AWS SDK または CLI で AssumeRole を使用する](#)
  - [AWS SDK または CLI で AssumeRoleWithWebIdentity を使用する](#)
  - [AWS SDK または CLI で DecodeAuthorizationMessage を使用する](#)

- [AWS SDK または CLI で GetFederationToken を使用する](#)
- [AWS SDK または CLI で GetSessionToken を使用する](#)
- [AWS SDK を使用する AWS STS のシナリオ](#)
  - [AWS SDK を使用して AWS STS で MFA トークンを必要とする IAM ロールを割り当てる](#)
  - [AWS SDK を使用して、フェデレーションユーザー向けに AWS STS で URL を作成する](#)
  - [AWS SDK を使用して、AWS STS で MFA トークンを必要とするセッショントークンを取得する](#)

## AWS SDK を使用した AWS STS 向けアクション

以下は、AWS SDK を使用して個々の AWS STS アクションを実行する方法を説明するコード例です。これらは AWS STS API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「[AWS Security Token Service \(AWS STS\) API リファレンス](#)」を参照してください。

### 例

- [AWS SDK または CLI で AssumeRole を使用する](#)
- [AWS SDK または CLI で AssumeRoleWithWebIdentity を使用する](#)
- [AWS SDK または CLI で DecodeAuthorizationMessage を使用する](#)
- [AWS SDK または CLI で GetFederationToken を使用する](#)
- [AWS SDK または CLI で GetSessionToken を使用する](#)

## AWS SDK または CLI で **AssumeRole** を使用する

以下のコード例は、AssumeRole の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [MFA トークンを必要とする IAM ロールを割り当てる](#)
- [フェデレーションユーザー向け URL の作成](#)



## .NET

### AWS SDK for .NET

#### Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
        id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
            the
            // default user.
```

```
        var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

        var client = new
Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

        // Get and display the information about the identity of the default
user.
        var callerIdRequest = new GetCallerIdentityRequest();
        var caller = await client.GetCallerIdentityAsync(callerIdRequest);
        Console.WriteLine($"Original Caller: {caller.Arn}");

        // Create the request to use with the AssumeRoleAsync call.
        var assumeRoleReq = new AssumeRoleRequest()
        {
            DurationSeconds = 1600,
            RoleSessionName = "Session1",
            RoleArn = roleArnToAssume
        };

        var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

        // Now create a new client based on the credentials of the caller
assuming the role.
        var client2 = new AmazonSecurityTokenServiceClient(credentials:
assumeRoleRes.Credentials);

        // Get and display information about the caller that has assumed the
defined role.
        var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
        Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
    }
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[AssumeRole](#)」を参照してください。

## Bash

### Bash スクリプトを使用した AWS CLI

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
# credentials.
#
# Parameters:
#     -n role_session_name -- The name of the session.
#     -r role_arn -- The ARN of the role to assume.
#
# Returns:
```

```

#     [access_key_id, secret_access_key, session_token]
#     And:
#     0 - If successful.
#     1 - If an error occurred.
#####
function sts_assume_role() {
    local role_session_name role_arn response
    local option OPTARG # Required to use getopts command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function sts_assume_role"
        echo "Assumes a role in the AWS account and returns the temporary
credentials:"
        echo "  -n role_session_name -- The name of the session."
        echo "  -r role_arn -- The ARN of the role to assume."
        echo ""
    }

    while getopts n:r:h option; do
        case "${option}" in
            n) role_session_name=${OPTARG} ;;
            r) role_arn=${OPTARG} ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    response=$(aws sts assume-role \
        --role-session-name "$role_session_name" \
        --role-arn "$role_arn" \
        --output text \
        --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then

```

```
aws_cli_error_log $error_code
errecho "ERROR: AWS reports create-role operation failed.\n$response"
return 1
fi

echo "$response"

return 0
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[AssumeRole](#)」を参照してください。

## C++

### SDK for C++

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[AssumeRole](#)」を参照してください。

## CLI

### AWS CLI

ロールを引き受けるには

次の `assume-role` コマンドは、IAM ロール `s3-access-example` のために短期間有効な認証情報のセットを取得します。

```
aws sts assume-role \
  --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
  --role-session-name s3-access-example
```

出力:

```
{
  "AssumedRoleUser": {
    "AssumedRoleId": "AR0A3XFRBF535PLBIFPI4:s3-access-example",
```

```
    "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
  },
  "Credentials": {
    "SecretAccessKey": "9drTJvcXLB89EXAMPLEL8923FB892xMFI",
    "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtY1FVgAUj
+7Indz3LU0aTWk1WKIjHmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo50ytwU=",
    "Expiration": "2016-03-15T00:05:07Z",
    "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
  }
}
```

コマンドの出力には、AWS に対する認証に使用できるアクセスキー、シークレットキー、およびセッショントークンが含まれています。

AWS CLI を使用する場合は、ロールに関連付けられた名前付きプロファイルを設定できます。プロファイルを使用すると、AWS CLI は `assign-role` を呼び出し、ユーザーのために認証情報を管理します。詳細については、「AWS CLI ユーザーガイド」の「[AWS CLI で IAM ロールを使用する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AssumeRole](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
```

```
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleArn> <roleSessionName>\s
```



```
        Where:
            roleArn - The Amazon Resource Name (ARN) of the role to
assume (for example, rn:aws:iam::000008047983:role/s3role).\s
            roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleArn = args[0];
    String roleSessionName = args[1];
    Region region = Region.US_EAST_1;
    StsClient stsClient = StsClient.builder()
        .region(region)
        .build();

    assumeGivenRole(stsClient, roleArn, roleSessionName);
    stsClient.close();
}

public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();

        // Display the time when the temp creds expire.
        Instant exTime = myCreds.expiration();
        String tokenInfo = myCreds.sessionToken();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());
```

```
        formatter.format(exTime);
        System.out.println("The token " + tokenInfo + " expires on " +
exTime);

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AssumeRole](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

クライアントを作成します。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

IAM ロールを割り当てます。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";
```

```
export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[AssumeRole](#)」を参照してください。

SDK for JavaScript (v2)

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
```

```
    DurationSeconds: 900,
  };
  var roleCreds;

  // Create the STS service object
  var sts = new AWS.STS({ apiVersion: "2011-06-15" });

  //Assume Role
  sts.assumeRole(roleToAssume, function (err, data) {
    if (err) console.log(err, err.stack);
    else {
      roleCreds = {
        accessKeyId: data.Credentials.AccessKeyId,
        secretAccessKey: data.Credentials.SecretAccessKey,
        sessionToken: data.Credentials.SessionToken,
      };
      stsGetCallerIdentity(roleCreds);
    }
  });

  //Get Arn of current identity
  function stsGetCallerIdentity(creds) {
    var stsParams = { credentials: creds };
    // Create STS service object
    var sts = new AWS.STS(stsParams);

    sts.getCallerIdentity({}, function (err, data) {
      if (err) {
        console.log(err, err.stack);
      } else {
        console.log(data.Arn);
      }
    });
  }
}
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[AssumeRole](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 一時的な認証情報一式 (アクセスキー、シークレットキー、セッショントークン) を返します。この認証情報は、リクエストしたユーザーが通常はアクセスできない AWS リソースに 1 時間アクセスするために使用できます。返される認証情報には、引き受けているロールのアクセスポリシーと提供されたポリシーで許可されている権限があります (提供されたポリシーを使用して、引き受けているロールのアクセスポリシーで定義されている権限を超える権限を付与することはできません)。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-Policy "...JSON policy..." -DurationInSeconds 3600
```

例 2: 引き受けているロールのアクセスポリシーで定義されているのと同じ権限を持つ、1 時間有効の一時的な認証情報を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600
```

例 3: コマンドレットの実行に使用されるユーザー認証情報に関連付けられている MFA からシリアル番号と生成されたトークンを提供する一時的な認証情報一式を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

例 4: 顧客アカウントで定義されているロールを引き受けた一時的な認証情報一式を返します。第三者が引き受けることができるロールごとに、顧客アカウントは、ロールを引き受けるたび、-ExternalId パラメータで渡す必要がある識別子を使用してロールを作成する必要があります。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
-DurationInSeconds 3600 -ExternalId "ABC123"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AssumeRole](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

MFA トークンを必要とする IAM ロールを想定し、一時的な認証情報を使用してアカウントの Amazon S3 バケットを一覧表示します。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
        grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
    MFA
        device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
    role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")
```

```
s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AssumeRole](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

GitHubには、その他のリソースもあります。用例一覧を検索し、[AWSコード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
```

```
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
# are used.
# @param sts_client [AWS::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AssumeRole](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
Option<String>) {
  let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
    .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
    .configure(config)
    .build()
    .await;

  let local_config = aws_config::from_env()
    .credentials_provider(provider)
```



```
.load()
.await;
let client = Client::new(&local_config);
let req = client.get_caller_identity();
let resp = req.send().await;
match resp {
    Ok(e) => {
        println!("UserID :           {}",
e.user_id().unwrap_or_default());
        println!("Account:           {}",
e.account().unwrap_or_default());
        println!("Arn      :           {}", e.arn().unwrap_or_default());
    }
    Err(e) => println!("{:?}", e),
}
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[AssumeRole](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
    async throws -> STSClientTypes.Credentials {
    let input = AssumeRoleInput(
```

```
        roleArn: role.arn,
        roleSessionName: sessionName
    )
    do {
        let output = try await stsClient.assumeRole(input: input)

        guard let credentials = output.credentials else {
            throw ServiceHandlerError.authError
        }

        return credentials
    } catch {
        throw error
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AssumeRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **AssumeRoleWithWebIdentity** を使用する

以下のコード例は、AssumeRoleWithWebIdentity の使用方法を示しています。

### CLI

#### AWS CLI

ウェブ ID (OAuth 2.0) で認証されたロールの短期認証情報を取得するには

次の `assume-role-with-web-identity` コマンドは、IAM ロール `app1` のために短期間有効な認証情報のセットを取得します。リクエストは、指定されたウェブ ID プロバイダーから提供されたウェブ ID トークンを使用して認証されます。ユーザーが実行できることをさらに制限するために、2 つの追加ポリシーがセッションに適用されます。返される認証情報は、生成されてから 1 時間後に失効します。

```
aws sts assume-role-with-web-identity \
```

```

--duration-seconds 3600 \
--role-session-name "app1" \
--provider-id "www.amazon.com" \
--policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
--role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
--web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1w17WTI7jn-Pcb6M-
buCgHhF0zTQxod27L9Cqn0Lio7N3gZAGpsp6n1-
AJB0CJckcyXe2c6uD0sr0JeZlKUm2eTDVMf8IEhDVI0r1Q0nTV6KzzAI30Y87Vd_cVMQ"

```

出力:

```

{
  "SubjectFromWebIdentityToken": "amzn1.account.AF6RH07KZU5XRvQJGXXK6HB56KR2A"
  "Audience": "client.5498841531868486423.1548@apps.example.com",
  "AssumedRoleUser": {
    "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
    "AssumedRoleId": "AROACLKWSQRAOEXAMPLE:app1"
  }
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY",
    "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4B1CFFxWNE10PTgk5TthT
+FvwqnKwRc0IfrrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgrmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgkBN9bkUDNCJiBeb/
AX1zBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wn0E49mR1/+0tkIKG07fAE",
    "Expiration": "2020-05-19T18:06:10+00:00"
  },
  "Provider": "www.amazon.com"
}

```

詳細については、「AWS IAM ユーザーガイド」の「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AssumeRoleWithWebIdentity](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: Login with Amazon ID プロバイダーで認証されたユーザーの一時的な認証情報一式 (1 時間有効) を返します。認証情報は、ロール ARN によって識別されるロールに関連付けられたアクセスポリシーを引き受けます。オプションで、JSON ポリシーを `-Policy` パラメータに渡して、アクセス権限をさらに絞り込むことができます (ロールに関連付けられている権限で使用可能な権限よりも多くの権限を付与することはできません)。 `-WebIdentityToken` に渡される値は、ID プロバイダーから返された一意のユーザー識別子です。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
  -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
  FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[AssumeRoleWithWebIdentity](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で `DecodeAuthorizationMessage` を使用する

以下のコード例は、`DecodeAuthorizationMessage` の使用方法を示しています。

#### CLI

##### AWS CLI

リクエストへの応答として返されるエンコードされた認証メッセージをデコードするには

次の `decode-authorization-message` の例では、Amazon Web Services リクエストへの応答として返されるエンコードされたメッセージから、リクエストの承認ステータスに関する追加情報をデコードします。

```
aws sts decode-authorization-message \
  --encoded-message EXAMPLEwodyRNrt1QARDip-
  eTA6i6Dr1UhHhPQrLWB_1Ab15pAKx19mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqe07e2NMk5j_hmzAiChJN-8oy3
  0jau7BMj0TWw0tHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
  p0u3FZjwYStfvTb3GHs3-6rLribG09jZ0ktkfE6vqx1FzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioggo
```

```
JWP5pwe_mAyyqh0NLw-r1S56YC_90onj9A80sNrH1I-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYgTpp5NIeSpuf3f0HanFuBZgENxZQZ2dlH3xJGMttYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067P0FrZECLjj0tNPBOZCcuEKE
```

出力:

```
{
  "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
  \"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\", \"effect
  \":\"DENY\", \"principals\":{\"items\":[{\"value\":\"ARO0A123456789EXAMPLE
  \"}]}}, \"principalGroups\":{\"items\":[]}, \"actions\":{\"items\":[{\"value
  \":\"ec2:RunInstances\"}]}, \"resources\":{\"items\":[{\"value\":\"*
  \"}]}}, \"conditions\":{\"items\":[]}}}, \"failures\":{\"items\":[]},
  \"context\":{\"principal\":{\"id\":\"ARO0A123456789EXAMPLE:Ana\", \"arn
  \":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"}, \"action\":
  \"RunInstances\", \"resource\": \"arn:aws:ec2:us-east-1:111122223333:instance/*
  \", \"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
  \"values\":{\"items\":[{\"value\":\"2\"}]}}], {\"key\":\"ec2:InstanceMarketType
  \", \"values\":{\"items\":[{\"value\":\"on-demand\"}]}}], {\"key\":\"aws:Resource
  \", \"values\":{\"items\":[{\"value\":\"instance/*\"}]}}], {\"key\":\"aws:Account
  \", \"values\":{\"items\":[{\"value\":\"111122223333\"}]}}], {\"key\":
  \"ec2:AvailabilityZone\", \"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}}],
  {\"key\":\"ec2:ebsoptimized\", \"values\":{\"items\":[{\"value\":\"false\"}]}}],
  {\"key\":\"ec2:IsLaunchTemplateResource\", \"values\":{\"items\":[{\"value\":
  \"false\"}]}}], {\"key\":\"ec2:InstanceType\", \"values\":{\"items\":{
  [\"value\":\"t2.micro\"}]}}], {\"key\":\"ec2:RootDeviceType\", \"values\":{\"items\":
  [\"value\":\"ebs\"}]}}], {\"key\":\"aws:Region\", \"values\":{\"items\":{
  [\"value\":\"us-east-1\"}]}}], {\"key\":\"ec2:MetadataHttpEndpoint\", \"values\":{\"items
  \":[{\"value\":\"enabled\"}]}}], {\"key\":\"aws:Service\", \"values\":{\"items
  \":[{\"value\":\"ec2\"}]}}], {\"key\":\"ec2:InstanceID\", \"values\":{\"items\":
  [\"value\":\"*\"}]}}], {\"key\":\"ec2:MetadataHttpTokens\", \"values\":{\"items
  \":[{\"value\":\"required\"}]}}], {\"key\":\"aws:Type\", \"values\":{\"items
  \":[{\"value\":\"instance\"}]}}], {\"key\":\"ec2:Tenancy\", \"values\":{\"items
  \":[{\"value\":\"default\"}]}}], {\"key\":\"ec2:Region\", \"values\":{\"items
  \":[{\"value\":\"us-east-1\"}]}}], {\"key\":\"aws:ARN\", \"values\":{\"items\":
  [\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}"
```

詳細については、「AWS IAM ユーザーガイド」の「[ポリシーの評価論理](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DecodeAuthorizationMessage](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: リクエストへの応答として返された、指定されたエンコード済みメッセージコンテンツに含まれる追加情報をデコードします。承認ステータスの詳細は、アクションをリクエストしたユーザーが見てはならない特権情報である可能性があるため、追加情報はエンコードされません。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[DecodeAuthorizationMessage](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **GetFederationToken** を使用する

以下のコード例は、GetFederationToken の使用方法を示しています。

#### CLI

##### AWS CLI

IAM ユーザーアクセスキー認証情報を使用して一時的なセキュリティ認証情報一式を返すには

次の `get-federation-token` の例では、ユーザーの一時的なセキュリティ認証情報一式 (アクセスキー ID、シークレットアクセスキー、セキュリティトークンで構成) を返します。IAM ユーザーの長期的なセキュリティ認証情報を使用して、GetFederationToken オペレーションを呼び出す必要があります。

```
aws sts get-federation-token \  
  --name Bob \  
  --policy file://myfile.json \  
  --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \  
  --duration-seconds 900
```

myfile.json の内容:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
    }
  ]
}
```

出力:

```
{
  "Credentials": {
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/  
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg  
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QunWMTfKq0DCOP/////////  
wEQAxoMNDUy0TI1MTcwNTA3Igxuy3A0puuoLsk3MJwqgQPg8Q0d9HuoClUxq26wnc/nm  
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
    "Expiration": "2023-12-20T02:06:07+00:00"
  },
}
```

```
"FederatedUser": {
  "FederatedUserId": "111122223333:Bob",
  "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
},
"PackedPolicySize": 36
}
```

詳細については、「AWS IAM ユーザーガイド」の「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetFederationToken](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 「Bob」をフェデレーションユーザーの名前として使用して、1 時間有効なフェデレーショントークンをリクエストします。この名前は、リソースベースのポリシー (Amazon S3 バケットポリシーなど) のフェデレーションユーザー名を参照するために使用できます。提供される JSON 形式の IAM ポリシーを使用して、IAM ユーザーが利用できるアクセス権限の範囲を絞り込みます。指定されたポリシーでは、リクエストしたユーザーに付与されたアクセス権限よりも多くのアクセス権限を付与することはできません。フェデレーションユーザーの最終的なアクセス権限は、渡されたポリシーと IAM ユーザーポリシーの共通部分に基づいて最も制限の厳しい一式となります。

```
Get-STS FederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
3600
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetFederationToken](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetSessionToken** を使用する

以下のコード例は、GetSessionToken の使用方法を示しています。



アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [MFA トークンを必要とするセッショントークンの取得](#)

## CLI

### AWS CLI

IAM ID のために短期間有効な認証情報のセットを取得するには

次の `get-session-token` コマンドは、呼び出しを実行する IAM ID のために短期間有効な認証情報のセットを取得します。結果として得られる認証情報は、ポリシーによって多要素認証 (MFA) が必要とされるリクエストのために使用できます。認証情報は生成されてから 15 分後に失効します。

```
aws sts get-session-token \  
  --duration-seconds 900 \  
  --serial-number "YourMFADeviceSerialNumber" \  
  --token-code 123456
```

出力:

```
{  
  "Credentials": {  
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",  
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY",  
    "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4B1CFFxWNE10PTgk5TthT  
+FvwqnKwRc0IfrrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/  
IvU1dYUg2RVAJBanLiHb4IgrmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgkBN9bkUDNCJiBeb/  
AX1zBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wn0E49mRl/+0tkIKG07fAE",  
    "Expiration": "2020-05-19T18:06:10+00:00"  
  }  
}
```

詳細については、「AWS IAM ユーザーガイド」の「[一時的なセキュリティ認証情報のリクエスト](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetSessionToken](#)」を参照してください。

## PowerShell

### Tools for PowerShell

例 1: 一定期間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。一時的な認証情報をリクエストするために使用される認証情報は、現在のシェルのデフォルトから推測されます。他の認証情報を指定するには、`-ProfileName` または `-AccessKey/-SecretKey` パラメータを使用します。

```
Get-STSSessionToken
```

出力:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokeN.....

例 2: 1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。リクエストを行うために使用される認証情報は、指定されたプロファイルから取得されます。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

出力:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokeN.....

例 3: プロファイル「myprofilename」で認証情報が指定されているアカウントに関連付けられた MFA デバイスの識別番号とデバイスから提供された値を使用して、1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

出力:

```
AccessKeyId                Expiration
SecretAccessKey            SessionToken
-----
-----
EXAMPLEACCESSKEYID        2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleTokenN.....
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「[GetSessionToken](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

MFA トークンを渡してセッショントークンを取得し、それを使用してアカウントの Amazon S3 バケットを一覧表示します。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                               device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
```

```
:param sts_client: A Boto3 STS instance that has permission to assume the
role.
"""
if mfa_serial_number is not None:
    response = sts_client.get_session_token(
        SerialNumber=mfa_serial_number, TokenCode=mfa_totp
    )
else:
    response = sts_client.get_session_token()
temp_credentials = response["Credentials"]

s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)

print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetSessionToken](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用する AWS STS のシナリオ

以下のコード例は、AWS STS で AWS SDK を使用した一般的なシナリオを実装する方法を示しています。これらのシナリオは、AWS STS 内で複数の関数を呼び出すことによって特定のタスクを実行する方法を示しています。それぞれのシナリオには、GitHub へのリンクがあり、コードを設定および実行する方法についての説明が記載されています。

### 例

- [AWS SDK を使用して AWS STS で MFA トークンを必要とする IAM ロールを割り当てる](#)

- [AWS SDK を使用して、フェデレーションユーザー向けに AWS STS で URL を作成する](#)
- [AWS SDK を使用して、AWS STS で MFA トークンを必要とするセッショントークンを取得する](#)

## AWS SDK を使用して AWS STS で MFA トークンを必要とする IAM ロールを割り当てる

次のコード例は、MFA トークンを必要とするロールを引き受ける方法を示しています。

### Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#) などの ID プロバイダーとのフェデレーションを使用してください。

- Amazon S3 バケットを一覧表示するためのアクセス許可を付与する、IAM ロールを作成します。
- MFA 認証情報が指定された場合にのみロールを引き受けることが許可された、IAM ユーザーを作成します。
- ユーザーのための MFA デバイスを登録します。
- ロールを引き受け、Amazon S3 バケットを一覧表示するために一時的な認証情報を使用します。

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーを作成し、MFA デバイスを登録し、Amazon S3 バケットを一覧表示するアクセス許可を付与するためのロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。

```
def setup(iam_resource):  
    """
```

```
Creates a new user with no permissions.
Creates a new virtual MFA device.
Displays the QR code to seed the device.
Asks for two codes from the MFA device.
Registers the MFA device for the user.
Creates an access key pair for the user.
Creates a role with a policy that lets the user assume the role and requires
MFA.
Creates a policy that allows listing Amazon S3 buckets.
Attaches the policy to the role.
Creates an inline policy for the user that lets the user assume the role.
```

For demonstration purposes, the user is created in the same account as the role, but in practice the user would likely be from another account.

Any MFA device that can scan a QR code will work with this demonstration. Common choices are mobile apps like LastPass Authenticator, Microsoft Authenticator, or Google Authenticator.

```
:param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                        that has permissions to create users, roles, and
policies
                        in the account.
:return: The newly created user, user key, virtual MFA device, and role.
"""
user = iam_resource.create_user(Username=unique_name("user"))
print(f"Created user {user.name}.")

virtual_mfa_device = iam_resource.create_virtual_mfa_device(
    VirtualMFADeviceName=unique_name("mfa")
)
print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

print(
    f"Showing the QR code for the device. Scan this in the MFA app of your "
    f"choice."
)
with open("qr.png", "wb") as qr_file:
    qr_file.write(virtual_mfa_device.qr_code_png)
webbrowser.open(qr_file.name)

print(f"Enter two consecutive code from your MFA device.")
```

```
mfa_code_1 = input("Enter the first code: ")
mfa_code_2 = input("Enter the second code: ")
user.enable_mfa(
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2,
)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end="")
progress_bar(10)

role = iam_resource.create_role(
    RoleName=unique_name("role"),
    AssumeRolePolicyDocument=json.dumps(
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"AWS": user.arn},
                    "Action": "sts:AssumeRole",
                    "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                }
            ],
        }
    ),
)
print(f"Created role {role.name} that requires MFA.")

policy = iam_resource.create_policy(
    PolicyName=unique_name("policy"),
    PolicyDocument=json.dumps(
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "s3:ListAllMyBuckets",
```

```
        "Resource": "arn:aws:s3:::*",
    }
    ],
}
),
)
role.attach_policy(PolicyArn=policy.arn)
print(f"Created policy {policy.policy_name} and attached it to the role.")

user.create_policy(
    PolicyName=unique_name("user-policy"),
    PolicyDocument=json.dumps(
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "sts:AssumeRole",
                    "Resource": role.arn,
                }
            ],
        }
    ),
)
print(
    f"Created an inline policy for {user.name} that lets the user assume "
    f"the role."
)

print("Give AWS time to propagate these new resources and connections.",
end="")
progress_bar(10)

return user, user_key, virtual_mfa_device, role
```

MFA トークンなしでロールを引き受けることは許可されていないことを示します。

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
```



```
Shows that attempting to assume the role without sending MFA credentials
results
in an AccessDenied error.

:param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
:param session_name: The name of the STS session.
:param sts_client: A Boto3 STS instance that has permission to assume the
role.
"""
print(f"Trying to assume the role without sending MFA credentials...")
try:
    sts_client.assume_role(RoleArn=assume_role_arn,
RoleSessionName=session_name)
    raise RuntimeError("Expected AccessDenied error.")
except ClientError as error:
    if error.response["Error"]["Code"] == "AccessDenied":
        print("Got AccessDenied.")
    else:
        raise
```

Amazon S3 バケットを一覧表示するためのアクセス許可を付与するロールを引き受け、必要な MFA トークンを渡して、バケットが一覧表示可能なことを出力します。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
        grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
        device, this is an ARN.
```

```
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the
role.
"""
response = sts_client.assume_role(
    RoleArn=assume_role_arn,
    RoleSessionName=session_name,
    SerialNumber=mfa_serial_number,
    TokenCode=mfa_totp,
)
temp_credentials = response["Credentials"]
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

デモ用に作成されたリソースを破棄します。

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

```
for user_pol in user.policies.all():
    user_pol.delete()
    print("Deleted inline user policy.")
for key in user.access_keys.all():
    key.delete()
    print("Deleted user's access key.")
for mfa in user.mfa_devices.all():
    mfa.disassociate()
virtual_mfa_device.delete()
user.delete()
print(f"Deleted {user.name}.")
```

このシナリオは、前に定義した関数を使用して実行します。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
```

```
teardown(user, virtual_mfa_device, role)
print("Thanks for watching!")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AssumeRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、フェデレーションユーザー向けに AWS STS で URL を作成する

次のコードサンプルは、以下の操作方法を示しています。

- 現在のアカウントの Amazon S3 リソースに対する読み取り専用のアクセス権限を付与する IAM ロールを作成します。
- AWS フェデレーションエンドポイントから、セキュリティトークンを取得します。
- フェデレーションされた認証情報を使用して、コンソールにアクセスする際に使用する URL を作成します。

### Python

#### SDK for Python (Boto3)

##### Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

現在のアカウントの Amazon S3 リソースに対する読み取り専用アクセス権限を付与するロールを作成します。

```
def setup(iam_resource):
    """
```

```

Creates a role that can be assumed by the current user.
Attaches a policy that allows only Amazon S3 read-only access.

:param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
instance
                        that has the permission to create a role.
:return: The newly created role.
"""
role = iam_resource.create_role(
    RoleName=unique_name("role"),
    AssumeRolePolicyDocument=json.dumps(
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"AWS": iam_resource.CurrentUser().arn},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
    ),
)
role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
print(f"Created role {role.name}.")

print("Give AWS time to propagate these new resources and connections.",
end="")
progress_bar(10)

return role

```

AWS フェデレーションエンドポイントからセキュリティトークンを取得し、フェデレーション認証情報を使用してコンソールにアクセスするために使用できる URL を作成します。

```

def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
    Management

```

Console.

1. Acquires temporary credentials from AWS Security Token Service (AWS STS) that can be used to assume a role with limited permissions.
2. Uses the temporary credentials to request a sign-in token from the AWS federation endpoint.
3. Builds a URL that can be used in a browser to navigate to the AWS federation endpoint, includes the sign-in token for authentication, and redirects to the AWS Management Console with permissions defined by the role that was specified in step 1.

:param assume\_role\_arn: The role that specifies the permissions that are granted.

The current user must have permission to assume the role.

:param session\_name: The name for the STS session.

:param issuer: The organization that issues the URL.

:param sts\_client: A Boto3 STS instance that can assume the role.

:return: The federated URL.

"""

```
response = sts_client.assume_role(
    RoleArn=assume_role_arn, RoleSessionName=session_name
)
temp_credentials = response["Credentials"]
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

session_data = {
    "sessionId": temp_credentials["AccessKeyId"],
    "sessionKey": temp_credentials["SecretAccessKey"],
    "sessionToken": temp_credentials["SessionToken"],
}
aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

# Make a request to the AWS federation endpoint to get a sign-in token.
# The requests.get function URL-encodes the parameters and builds the query
string
# before making the request.
response = requests.get(
    aws_federated_signin_endpoint,
    params={
        "Action": "getSignInToken",
        "SessionDuration": str(datetime.timedelta(hours=12).seconds),
```

```
        "Session": json.dumps(session_data),
    },
)
signin_token = json.loads(response.text)
print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

# Make a federated URL that can be used to sign into the AWS Management
Console.
query_string = urllib.parse.urlencode(
    {
        "Action": "login",
        "Issuer": issuer,
        "Destination": "https://console.aws.amazon.com/",
        "SigninToken": signin_token["SigninToken"],
    }
)
federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
return federated_url
```

デモ用に作成されたリソースを破棄します。

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

このシナリオは、前に定義した関数を使用して実行します。

```
def usage_demo():
    """Drives the demonstration."""
```

```
print("-" * 88)
print(f"Welcome to the AWS Security Token Service federated URL demo.")
print("-" * 88)
iam_resource = boto3.resource("iam")
role = setup(iam_resource)
sts_client = boto3.client("sts")
try:
    federated_url = construct_federated_url(
        role.arn, "AssumeRoleDemoSession", "example.org", sts_client
    )
    print(
        "Constructed a federated URL that can be used to connect to the "
        "AWS Management Console with role-defined permissions:"
    )
    print("-" * 88)
    print(federated_url)
    print("-" * 88)
    _ = input(
        "Copy and paste the above URL into a browser to open the AWS "
        "Management Console with limited permissions. When done, press "
        "Enter to clean up and complete this demo."
    )
finally:
    teardown(role)
    print("Thanks for watching!")
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AssumeRole](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、AWS STS で MFA トークンを必要とするセッショントークンを取得する

次のコード例では、MFA トークンを必要とするセッショントークンを取得する方法を示します。



**⚠ Warning**

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#) などの ID プロバイダーとのフェデレーションを使用してください。

- Amazon S3 バケットを一覧表示するためのアクセス許可を付与する、IAM ロールを作成します。
- MFA 認証情報が指定された場合にのみロールを引き受けることが許可された、IAM ユーザーを作成します。
- ユーザーのための MFA デバイスを登録します。
- セッショントークンを取得するための MFA 認証情報を指定し、一時的な認証情報により S3 バケットを一覧表示します。

## Python

### SDK for Python (Boto3)

**i Note**

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーの作成と MFA デバイスの登録を行い、ユーザーが Amazon S3 バケットを一覧表示できるようにするアクセス許可を、MFA 認証情報が使用されている場合にのみ付与するロールを作成します。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
    buckets,
    but only when MFA credentials are used.
```

Any MFA device that can scan a QR code will work with this demonstration. Common choices are mobile apps like LastPass Authenticator, Microsoft Authenticator, or Google Authenticator.

```
:param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
    that has permissions to create users, MFA devices, and
    policies in the account.
:return: The newly created user, user key, and virtual MFA device.
"""
user = iam_resource.create_user(Username=unique_name("user"))
print(f"Created user {user.name}.")

virtual_mfa_device = iam_resource.create_virtual_mfa_device(
    VirtualMFADeviceName=unique_name("mfa")
)
print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

print(
    f"Showing the QR code for the device. Scan this in the MFA app of your "
    f"choice."
)
with open("qr.png", "wb") as qr_file:
    qr_file.write(virtual_mfa_device.qr_code_png)
webbrowser.open(qr_file.name)

print(f"Enter two consecutive code from your MFA device.")
mfa_code_1 = input("Enter the first code: ")
mfa_code_2 = input("Enter the second code: ")
user.enable_mfa(
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2,
)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end="")
progress_bar(10)
```

```
user.create_policy(
    PolicyName=unique_name("user-policy"),
    PolicyDocument=json.dumps(
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "s3:ListAllMyBuckets",
                    "Resource": "arn:aws:s3:::*",
                    "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}}},
            ]
        }
    ),
)
print(
    f"Created an inline policy for {user.name} that lets the user list
buckets, "
    f"but only when MFA credentials are present."
)

print("Give AWS time to propagate these new resources and connections.",
end="")
progress_bar(10)

return user, user_key, virtual_mfa_device
```

MFA トークンを渡して一時的なセッション認証情報を取得し、その認証情報を使用してアカウントの S3 バケットを一覧表示します。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.
```

```
:param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                               device, this is an Amazon Resource Name (ARN).
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the
role.
"""
if mfa_serial_number is not None:
    response = sts_client.get_session_token(
        SerialNumber=mfa_serial_number, TokenCode=mfa_totp
    )
else:
    response = sts_client.get_session_token()
temp_credentials = response["Credentials"]

s3_resource = boto3.resource(
    "s3",
    aws_access_key_id=temp_credentials["AccessKeyId"],
    aws_secret_access_key=temp_credentials["SecretAccessKey"],
    aws_session_token=temp_credentials["SessionToken"],
)

print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

デモ用に作成されたリソースを破棄します。

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
```

```
print("Deleted user's access key.")
for mfa in user.mfa_devices.all():
    mfa.disassociate()
virtual_mfa_device.delete()
user.delete()
print(f"Deleted {user.name}.")
```

このシナリオは、前に定義した関数を使用して実行します。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
            mfa_totp = input("Enter the code from your registered MFA device: ")
            list_buckets_with_session_token_with_mfa(
                virtual_mfa_device.serial_number, mfa_totp, sts_client
            )
    finally:
        teardown(user, virtual_mfa_device)
    print("Thanks for watching!")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[GetSessionToken](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「[AWS SDK での IAM の使用](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

# IAM および AWS STS のセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。[AWS コンプライアンスプログラム](#)の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。AWS Identity and Access Management (IAM) に適用されるコンプライアンスプログラムの詳細については、コンプライアンスプログラムによる[AWS 対象範囲内のサービスコンプライアンスプログラム](#)によるを参照してください。
- クラウド内のセキュリティ - お客様の責任は使用する AWS のサービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、AWS Identity and Access Management (IAM) および AWS Security Token Service(AWS STS) を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように IAM および AWS STS を設定する方法について説明します。また、IAM リソースのモニタリングや保護に役立つ、他の AWS のサービスの使用方法についても説明します。

## 目次

- [AWS セキュリティ認証情報](#)
- [AWS セキュリティ監査のガイドライン](#)
- [AWS Identity and Access Management でのデータ保護](#)
- [AWS Identity and Access Managementでのログ記録とモニタリング](#)
- [AWS Identity and Access Management のコンプライアンス検証](#)
- [AWS Identity and Access Management での耐障害性](#)
- [AWS Identity and Access Management でのインフラストラクチャセキュリティ](#)
- [AWS Identity and Access Management での設定と脆弱性の分析](#)

- [AWS Access Analyzer 用の AWS Identity and Access Management マネージドポリシー](#)

## AWS セキュリティ認証情報

AWS の操作時に、自身が誰であるか、そしてリクエストしているリソースに対してアクセス許可を持つかどうかを確認するための AWS セキュリティ認証情報を指定します。AWS は、このセキュリティ認証情報を使用してリクエストを認証、承認します。

たとえば、Amazon Simple Storage Service (Amazon S3) バケットから保護されたファイルをダウンロードする場合、認証情報はそのアクセスを許可する必要があります。認証情報にファイルをダウンロードする権限がないことが示される場合、AWS はリクエストを拒否します。ただし、公開されている Amazon S3 バケット内のファイルのダウンロードに AWS セキュリティ認証情報は必要ありません。

AWS には異なるタイプのユーザーがあります。すべての AWS ユーザーはセキュリティ認証情報を持っています。アカウント所有者 (ルートユーザー)、AWS IAM アイデンティティセンターのユーザー、フェデレーションユーザー、IAM ユーザーがいます。

ユーザーは長期または一時的なセキュリティ認証情報を持っています。ルートユーザー、IAM ユーザー、アクセスキーには、有効期限のない長期セキュリティ認証情報があります。認証情報を長期間保護するために、[アクセスキーの管理](#)、[パスワードの変更](#)、[MFA の有効化](#)を行うためのプロセスを準備する必要があります。

(IAM) ロール、AWS IAM アイデンティティセンターのユーザー またはフェデレーションユーザーは一時的なセキュリティ認証情報をもっています。一時的なセキュリティ認証情報は、定義された期間が経過するか、ユーザーがセッションを終了したときに期限切れになります。一時的認証情報の機能は、長期的な証情報とほとんど同じですが、次の相違点があります。

- 一時的セキュリティ認証情報は、その名前が示すとおり、使用期限が短くなっています。有効期限は数分から数時間に設定できます。認証情報が失効すると、AWS はそれらを認識しなくなります。また、その認証情報によって作成された API リクエストによるあらゆるタイプのアクセスが許可されなくなります。
- 一時的セキュリティ認証情報はユーザーとともに保存されることはなく、ユーザーのリクエストに応じて動的に生成され、提供されます。一時的セキュリティ認証情報が失効すると (または失効する前でも)、ユーザーは新しい認証情報をリクエストできます。ただし、リクエストするユーザーがまだその権限を持っている場合に限りです。

そのため、一時的な認証情報には、長期の認証情報よりも次の利点があります。



- アプリケーションの長期の AWS セキュリティ認証情報を配布したり埋め込んだりする必要がありません。
- ユーザーに対して AWS ID を定義せずに AWS リソースへのアクセスを許可できます。一時的認証情報は[ロールおよび ID フェデレーション](#)の基本となります。
- 一時的セキュリティ認証情報の有効期限は限られているので、認証情報が不要になった際に更新したり、明示的に取り消したりする必要がありません。一時的セキュリティ認証情報の有効期限が切れると、再利用することはできません。認証情報が有効な期間を、最大限度まで指定できます。

## セキュリティに関する考慮事項

後でアクセスする場合は、以下AWS アカウント を考慮することをお勧めします。

- AWS アカウント を作成すると、1つのルートユーザーが作成されます。ルートユーザー (アカウント所有者) の認証情報によって、アカウント内のすべてのリソースへのフルアクセスが許可されます。ルートユーザーが最初に実行するタスクは、ルートユーザーの使用を最小限にするために、別のユーザーに AWS アカウント の管理権限を付与することです。
- IAM ポリシーを使用して、リソースへのルートユーザーアクセスを明示的に拒否することはできません。AWS Organizations [サービスコントロールポリシー](#) (SCP) を使用できるのは、ルートユーザーの許可を制限する場合のみです。
- ルートユーザーのパスワードを忘れてたり、紛失したりした場合、このパスワードをリセットするには、アカウントに関連付けられている E メールアドレスにアクセスする必要があります。
- ルートユーザーアクセスキーを紛失した場合は、アカウントにルートユーザーとしてサインインして、新しいアクセスキーを作成する必要があります。
- ルートユーザーを日常的なタスクに使用しないでください。ルートユーザーのみが実行できるタスクを実行します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。
- 認証情報は、アカウントに固有です。複数の AWS アカウント にアクセスできる場合は、アカウントごとに別々の認証情報があります。
- [ポリシー](#)は、ユーザー、ロール、またはユーザーグループのメンバーが実行できるアクション、AWS リソース、および条件を決定します。ポリシーで、AWS アカウント 内の AWS のサービスやリソースへのアクセスを安全に制御できます。後でアクセス許可の変更または取り消しが必要になったら、ID に直接変更する場合は、ポリシーを削除または変更できます。
- Emergency Access IAM ユーザーのサインイン認証情報と、プログラムによるアクセス用に作成したアクセスキーは、必ず安全な場所に保存してください。アクセスキーを紛失した場合は、アカウントにサインインして新しいアクセスキーを作成する必要があります。

- IAM ユーザーとアクセスキーによって提供される長期認証情報の代わりに、IAM ロールとフェデレーションユーザーによって提供される一時的な認証情報を使用することを強くお勧めします。

## フェデレーテッド ID

フェデレーション ID は、外部のアイデンティティを持つユーザーが、安全な AWS アカウント リソースにアクセスするために使用できる一時的な AWS 認証情報を付与されたものです。外部認証は、企業の ID ストア (LDAP や Windows の Active Directory など) またはサードパーティー (Login with Amazon、Facebook、または Google でのログインなど) から取得できます。フェデレーション ID は AWS Management Console または AWS アクセスポータルではサインインしません。

フェデレーション ID が AWS にサインインできるようにするには、<https://signin.aws.amazon.com/federation> を含むカスタム URL を作成する必要があります。詳細については、「[カスタム ID ブロカーに対する AWS コンソールへのアクセスの許可](#)」を参照してください。

フェデレーション ID の詳細については、「[ID プロバイダーとフェデレーション](#)」を参照してください。

## 多要素認証 (MFA)

多要素認証 (MFA) は、AWS アカウント にアクセスできるユーザーのセキュリティをさらに強化するサービスです。セキュリティを高めるため、AWS アカウントのルートユーザー 認証情報と、すべての IAM ユーザーに対して MFA を必須にすることが推奨されます。詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

MFA をアクティブ化して AWS アカウント にサインインすると、サインインの認証情報に加え、MFA デバイスによって生成されるレスポンス (コード、タッチまたはタップ、生体認証スキャンなど) を求められます。MFA を追加すると、AWS アカウント 設定とリソースの安全性が高まります。

デフォルトでは、MFA は有効化されていません。AWS アカウントのルートユーザー に対する MFA デバイスの有効化と管理は、「[セキュリティ認証情報](#)」ページ、または AWS Management Console の [IAM](#) ダッシュボードにアクセスして実行できます。IAM ユーザー向け MFA のアクセスの有効化の詳細については、「[AWS でのユーザーの MFA デバイスの有効化](#)」を参照してください。

Multi-Factor Authentication (MFA) を使用したサインインについては、「[IAM のサインインページでの MFA デバイスの使用](#)」を参照してください。

## プログラムのアクセス

AWS へのプログラムによる呼び出しを行う場合、または AWS Command Line Interface か AWS Tools for PowerShell を使用する場合は、AWS アクセスキーを指定します。可能な場合は、短期のアクセスキーの使用をお勧めします。

長期的なアクセスキーを作成するときは、アクセスキー ID (AKIAIOSFODNN7EXAMPLE など) とシークレットアクセスキー (wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY など) をセットとして作成します。シークレットアクセスキーは、作成時にのみダウンロードできます。シークレットアクセスキーをダウンロードしない場合や紛失した場合は、新しいシークレットアクセスキーを作成する必要があります。

多くの場合、期限のない長期のアクセスキーは必要ありません (IAM ユーザー用のアクセスキーを作成する場合など)。その代わりに、IAM ロールを作成し、一時的なセキュリティ認証情報を生成できます。一時的なセキュリティ認証情報は、アクセスキー ID とシークレットアクセスキーが含まれていますが、認証情報がいつ無効になるかを示すセキュリティトークンも含まれています。有効期限が切れると、その後は有効にはなりません。

AKIA で始まるアクセスキー ID は、IAM ユーザーまたは AWS アカウントルートユーザーの長期的なアクセスキーです。ASIA で始まるアクセスキー ID は、AWS STS オペレーションを使用して作成される一時的な認証情報アクセスキーです。

AWS Management Console の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラムによるアクセスを許可する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

どのユーザーがプログラムによるアクセスを必要としますか？	To	By
ワークフォース ID (IAM Identity Center で管理されているユーザー)	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。  • AWS CLI については、「AWS Command Line Interface ユーザーガイ

どのユーザーがプログラムによるアクセスを必要としますか？	To	By
		<p>ド」の「<a href="#">AWS IAM Identity Center を使用するための AWS CLI の設定</a>」を参照してください。</p> <ul style="list-style-type: none"><li>• AWS SDK、ツール、および AWS API については、「AWS SDK とツールリファレンスガイド」の「<a href="#">IAM Identity Center 認証</a>」を参照してください。</li></ul>
IAM	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	「IAM ユーザーガイド」の「 <a href="#">AWS リソースでの一時的な認証情報の使用</a> 」の指示に従ってください。

どのユーザーがプログラムによるアクセスを必要としますか？	To	By
IAM	(非推奨) 長期的な認証情報を使用して、AWS CLI、AWS SDK、AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。  <ul style="list-style-type: none"> <li>• AWS CLI については、「AWS Command Line Interface ユーザーガイド」の「<a href="#">IAM ユーザー認証情報を使用した認証</a>」を参照してください。</li> <li>• AWS SDK とツールについては、「AWS SDK とツールリファレンスガイド」の「<a href="#">長期認証情報を使用して認証する</a>」を参照してください。</li> <li>• AWS API については、「IAM ユーザーガイド」の「<a href="#">IAM ユーザーのアクセスキーの管理</a>」を参照してください。</li> </ul>

## 長期的なアクセスキーに対する代替方法

多くの一般的なユースケースでは長期的なアクセスキーの代替方法があります。アカウントのセキュリティを強化するには、以下を検討してください。

- 長期的なアクセスキーやシークレットアクセスキーをアプリケーションコードやコードリポジトリに埋め込まないでください。代わりに AWS Secrets Manager またはその他のシークレット管理ソリューションを使用して、プレーンテキストでキーをハードコードする必要をなくします。アプリケーションまたはクライアントは必要に応じてシークレットを取得できます。詳細については、AWS Secrets Manager ユーザーガイドの「[AWS Secrets Manager とは何か](#)」を参照してください。

- 可能な場合には必ず IAM ロールを使用して一時的なセキュリティ認証情報を生成してください。可能な場合には、長期的なアクセスキーではなく、常に一時的なセキュリティ認証情報を発行するメカニズムを使用してください。一時的セキュリティ認証情報は、ユーザーとともに保存されることはなく、ユーザーのリクエストに応じて動的に生成され、提供されるため、より安全です。一時的セキュリティ認証情報は有効期間が限られているため、それらを管理したり更新したりする必要はありません。一時的なアクセスキーを提供するメカニズムには、IAM ロールや IAM Identity Center ユーザーの認証が含まれます。AWS の外部で実行されるマシンには、[AWS Identity and Access Management Roles Anywhere](#) を使用できます。
- AWS Command Line Interface (AWS CLI) または `aws-shell` の長期的なアクセスキーの代替方法を使用します。代替方法には以下が含まれます。
  - AWS CloudShell はブラウザーベースの事前認証済みシェルで、AWS Management Console から直接起動できます。任意のシェル (Bash、PowerShell、または Z シェル) を使用して、AWS のサービスに対して AWS CLI コマンドを実行できます。この手順を実行すると、コマンドラインツールのダウンロードもインストールも不要です。詳細については、AWS CloudShell ユーザーガイドの「[AWS CloudShell とは何か](#)」を参照してください。
  - AWS CLI バージョン 2 の AWS IAM Identity Center との統合 (IAM Identity Center)。ユーザーを認証し、短期的な認証情報を提供して AWS CLI コマンドを実行します。詳細については、AWS IAM Identity Center ユーザーガイドの「[AWS CLI と IAM Identity Center の統合](#)」および AWS Command Line Interface ユーザーガイドの「[IAM Identity Center を使用するための AWS CLI の設定](#)」を参照してください。
- アプリケーションまたは AWS のサービスにアクセスする必要があるユーザー向けに長期的なアクセスキーを作成しないでください。IAM Identity Center は、AWS のサービスにアクセスする外部 IdP ユーザーに一時的なアクセス認証情報を生成できます。これにより、IAM で長期的な認証情報を作成して管理する必要がなくなります。IAM Identity Center で、外部 IdP ユーザーにアクセス権を付与する IAM Identity Center のアクセス権限セットを作成します。次に、IAM Identity Center のグループを、選択した AWS アカウントのアクセス権限セットに割り当てます。詳細については、AWS IAM Identity Center ユーザーガイドの「[AWS IAM Identity Center とは](#)」、「[外部の ID プロバイダーとの接続](#)」、および「[アクセス権限セット](#)」を参照してください。
- 長期的なアクセスキーを AWS コンピューティングサービス内に保存しないでください。代わりに、コンピューティングリソースに IAM ロールを割り当ててください。これにより、アクセス権を付与する一時的な認証情報が自動的に提供されます。例えば、Amazon EC2 インスタンスにアタッチされるインスタンスプロファイルを作成すると、AWS ロールをインスタンスに割り当て、そのアプリケーションのすべてで使用できるようになります。インスタンスプロファイルにはロールが含まれ、Amazon EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得する



ことができます。詳細については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

## AWS にAWS認証情報を使用してアクセスする

AWS には、AWS へのアクセス方法とAWSユーザータイプに応じて異なる種類のセキュリティ認証情報が必要です。例えば、AWS Management Console にはサインインの認証情報を使用し、プログラムで AWS を呼び出すにはアクセスキーを使用します。また、アカウントのルートユーザー、AWS Identity and Access Management (IAM) ユーザー、AWS IAM Identity Centerユーザー、またはフェデレーション ID など、使用する各 ID はいずれも AWS 内で一意の認証情報を保持します。

ユーザータイプに応じて AWS にサインインする手順については、「AWS サインインユーザーガイド」の「[AWS にサインインするには](#)」を参照してください。

## AWS セキュリティ監査のガイドライン

セキュリティ設定を定期的に監査し、現在のビジネスニーズに対応していることを確認します。監査により、不要な IAM ユーザー、ロール、グループ、ポリシーを削除し、ユーザーとソフトウェアに対する過剰なアクセス許可がないことを確認する機会が得られます。

以下では、セキュリティのベストプラクティスを実践するために、AWS リソースを体系的に確認し、モニタリングするためのガイドラインを示します。

### Tip

[AWS Security Hub](#) を使用して、セキュリティのベストプラクティスに関連する IAM の使用状況をモニタリングできます。Security Hub は、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub を使用して IAM リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「[AWS アイデンティティおよびアクセス管理コントロール](#)」を参照してください。

### コンテンツ

- [セキュリティ監査を実行するタイミング](#)
- [監査のガイドライン](#)

- [AWS アカウントの認証情報の確認](#)
- [IAM ユーザーの確認](#)
- [IAM グループの確認](#)
- [IAM ロールの確認](#)
- [SAML および OpenID Connect \(OIDC\) 用 IAM プロバイダの確認](#)
- [モバイルアプリの確認](#)
- [IAM ポリシーを確認するためのヒント](#)

## セキュリティ監査を実行するタイミング

次の状況で、セキュリティ設定を監査します。

- 定期的な監査。セキュリティのベストプラクティスとして、このドキュメントで説明されている手順を定期的に行います。
- 従業員が退職するなど、組織に変更があった場合。
- 1つ以上の個別の AWS サービスの使用を停止したことで、アカウントのユーザーにとって不要になったアクセス許可を削除したことを検証する場合。
- Amazon EC2 インスタンス上のアプリケーション、AWS OpsWorks スタック、AWS CloudFormation テンプレートなど、アカウントでソフトウェアの追加または削除を行った場合。
- 権限のないユーザーがアカウントにアクセスした疑いがある場合。

## 監査のガイドライン

アカウントのセキュリティ設定を確認する際は、以下のガイドラインに従います。

- 徹底して行う。ほとんど使用しないものを含め、セキュリティ設定のあらゆる面について調べます。
- 推測しない。セキュリティ設定の一部 (例: 特定のポリシーやロールの存在の背後にある根拠) が不明な場合は、潜在的リスクを把握するまでビジネスニーズを調査します。
- 作業を単純にする。監査 (および管理) を容易にするために、IAM グループ、IAM ロール、一貫した命名スキーム、単純なポリシーを使用します。



## AWS アカウントの認証情報の確認

AWS アカウント認証情報を監査するときは、次の手順に従います。

1. 使用していないルートユーザーのアクセスキーがある場合は、削除できます。AWS での日常的な作業には、ルートアクセスキーを使用するのではなく、AWS IAM アイデンティティセンターのユーザーなどの一時的な認証情報を持つユーザーを使用することを[強くお勧めします](#)。
2. アカウントのアクセスキーが必要な場合は、[必要に応じて更新してください](#)。

## IAM ユーザーの確認

既存の IAM ユーザーを監査するときは、以下の手順に従います。

1. [ユーザーを一覧表示し](#)、不要な[ユーザーを削除します](#)。
2. アクセスを必要としない[グループからユーザーを削除します](#)。
3. ユーザーが所属するグループにアタッチされたポリシーを確認します。「[IAM ポリシーを確認するためのヒント](#)」を参照してください。
4. ユーザーが必要としない場合や公開された可能性がある場合、セキュリティ認証情報を削除します。例えば、アプリケーションに使用される IAM ユーザーはパスワードを必要としません (パスワードは AWS ウェブサイトへのサインインにのみ必要です)。同様に、ユーザーがアクセスキーを使用しない場合、ユーザーはそのキーを持つ理由がありません。詳細については、「[IAM ユーザーのパスワードの管理](#)」および「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。

アカウント内のすべての IAM ユーザーと、ユーザーの各種認証情報 (パスワード、アクセスキー、MFA デバイスなど) のステータスが示された認証情報レポートを生成し、ダウンロードできます。パスワードやアクセスキーでは、パスワードやアクセスキーを最近使用した日時が、認証情報レポートに表示されます。最近使用していない認証情報をアカウントから削除することを検討してください。(緊急アクセスユーザーを削除しないでください)。詳細については、「[AWS アカウントの認証情報レポートの取得](#)」を参照してください。

5. 長期的な認証情報を必要とするユースケースのために、必要に応じてパスワードとアクセスキーを更新してください。詳細については、「[IAM ユーザーのパスワードの管理](#)」および「[IAM ユーザーのアクセスキーの管理](#)」を参照してください。
6. ベストプラクティスでは、人間のユーザーが一時的な認証情報を使用して AWS にアクセスする際、アイデンティティプロバイダーとのフェデレーションを使用することが求められます。可能であれば、IAM ユーザーから IAM Identity Center のユーザーなどのフェデレーションユーザーに移行してください。アプリケーションに必要な最小限の IAM ユーザー数を保持してください。

## IAM グループの確認

IAM グループを監査するときは、次の手順に従います。

1. [グループを一覧表示](#)し、使用していない[グループを削除](#)します。
2. 各グループの[ユーザーを確認](#)し、属していない[ユーザーを削除](#)します。
3. グループにアタッチされるポリシーを確認します。「[IAM ポリシーを確認するためのヒント](#)」を参照してください。

## IAM ロールの確認

IAM ロールを監査するときは、次の手順に従います。

1. [ロールを一覧表示](#)し、使用していない[ロールを削除](#)します。
2. ロールの信頼ポリシーを[確認](#)します。プリンシパルが誰かを把握し、なぜアカウントまたはユーザーがロールを引き受けることができるようにする必要があるのかを理解しておいてください。
3. ロールを引き受けるユーザーすべてに適切なアクセス許可を付与できるよう、アクセスポリシーを[確認](#)します。[IAM ポリシーを確認するためのヒント](#)を参照してください。

## SAML および OpenID Connect (OIDC) 用 IAM プロバイダの確認

[SAML または OIDC ID プロバイダー \(IdP\)](#) との信頼を確立するために IAM エンティティを作成した場合は、以下の手順に従います。

1. 使用していないプロバイダーを削除します。
2. SAML IdP ごとの AWS メタデータドキュメントをダウンロードして確認し、ドキュメントが現在のビジネスニーズを反映していることを確認します。
3. SAML IdP から最新のメタデータドキュメントを取得して、[IAM のプロバイダーを更新](#)します。

## モバイルアプリの確認

AWS にリクエストを送信するモバイルアプリを作成する場合、以下の手順に従います。

1. 暗号化ストレージ内であっても、モバイルアプリケーションに埋め込みアクセスキーが含まれていないことを確認します。

2. その目的のために設計されている API を使用して、アプリケーションの一時的な認証情報を取得します。

### Note

[Amazon Cognito](#) を使用して、アプリでユーザー ID を管理することをお勧めします。このサービスでは、Login with Amazon、Facebook、Google、または OpenID Connect (OIDC) に対応している任意の ID プロバイダを使用してユーザーを認証できます。詳細については、「Amazon Cognito デベロッパーガイド」の「[Amazon Cognito ID プール](#)」を参照してください。

## IAM ポリシーを確認するためのヒント

ポリシーは強力かつ微妙であるため、各ポリシーによって付与されるアクセス許可を確認して理解することが重要です。ポリシーを確認するときは、以下のガイドラインを使用します。

- 個々のユーザーにではなくグループにポリシーをアタッチします。個々のユーザーにポリシーがある場合、なぜそのユーザーはポリシーを必要としているのかを理解しておいてください。
- IAM ユーザー、グループ、ロールに必要なアクセス許可が付与され、追加のアクセス許可が付与されていないことを確認します。
- [IAM ポリシーシミュレーター](#) を使用して、ユーザーまたはグループにアタッチされたポリシーをテストします。
- ユーザーのアクセス許可は、該当するすべてのポリシー、つまり (ユーザー、グループまたはロールの) アイデンティティベースのポリシーと、(Amazon S3 バケット、Amazon SQS キュー、Amazon SNS トピック、および AWS KMS キーなどのリソースに対する) リソースベースのポリシーの結果であることに注意してください。ユーザーに適用されるすべてのポリシーを調べ、個々のユーザーに付与されるアクセス許可一式を理解することが重要です。
- IAM ユーザー、グループ、ロール、またはポリシーの作成とそのプリンシパルエンティティへのポリシーの付与をユーザーに許可すると、実質的に、お客様のアカウント内のすべてのリソースに対するすべてのアクセス権限をそのユーザーに付与することになることに注意してください。ポリシーの作成とユーザー、グループ、またはロールへのポリシーのアタッチを許可されたユーザーは、すべてのアクセス許可をユーザー自身に付与することができます。通常、信頼できないユーザーまたはロールには、アカウントのリソースへのフルアクセスを含む IAM のアクセス許可を付与しないでください。セキュリティ監査を実行する際は、信頼できるアイデンティティに次の IAM アクセス許可が付与されていることを確認してください。

- iam:PutGroupPolicy
- iam:PutRolePolicy
- iam:PutUserPolicy
- iam:CreatePolicy
- iam:CreatePolicyVersion
- iam:AttachGroupPolicy
- iam:AttachRolePolicy
- iam:AttachUserPolicy
- 使用しないサービスにはポリシーがアクセス許可を付与しないようにします。たとえば、[AWS 管理ポリシー](#)を使用する場合は、アカウントで使用されている AWS 管理ポリシーが、実際に使用するサービス用であることを確認します。アカウントで使用されている AWS 管理ポリシーを確認するには、IAM [GetAccountAuthorizationDetails](#) API (AWS CLI コマンド: [aws iam get-account-authorization-details](#)) を使用します。
- ポリシーが、Amazon EC2 インスタンスを起動するアクセス許可をユーザーに付与する場合、iam:PassRole アクションの実行も許可する可能性があります。この場合、ユーザーが Amazon EC2 インスタンスに渡すことができる[ルールが明示的に一覧表示](#)されます。
- \* を含む Action 要素または Resource 要素の値を検証します。可能な場合は、ユーザーが必要とする個別のアクションおよびリソースへの Allow アクセス許可を付与します。ただし、\* をポリシーで使用するのが適切であること理由は、以下のとおりです。
  - ポリシーは、管理レベルのアクセス許可を付与するように設計されています。
  - ワイルドカード文字は、便宜上の理由から、類似するアクションのセット (例: Describe\*) に使用します。この方法で全アクションのリストを参照すれば便利です。
  - ワイルドカード文字は、リソースのクラスまたはリソースパス (例: arn:aws:iam::**account-id**:users/division\_abc/\*) を表示するために使用します。これは、そのクラスまたはパスのすべてのリソースへのアクセス許可を付与する際に便利です。
  - サービスアクションは、リソースレベルのアクセス許可をサポートしないため、リソースの唯一の選択肢は \* です。
- ポリシー名を調べて、ポリシーの機能を反映していることを確認します。たとえば、ポリシーに「読み取り専用」を含む名前が付いていても、そのポリシーは実際には書き込みや変更を許可することもあります。

セキュリティ監査の計画の詳細については、AWS Architecture Centerの「[セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス](#)」を参照してください。

## AWS Identity and Access Management でのデータ保護

AWS [責任共有モデル](#)は、AWS Identity and Access Management でのデータ保護に適用されます。このモデルで説明されているように、AWS には、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任があります。ユーザーには、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「[データプライバシーのよくある質問](#)」を参照してください。欧州でのデータ保護の詳細については、「AWS セキュリティブログ」に投稿された「[AWS 責任共有モデルおよび GDPR](#)」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみを各ユーザーに付与できます。また、次の方法でデータを保護することをおすすめします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須です。TLS 1.3 が推奨されます。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。
- AWS のサービス内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API により AWS にアクセスするときに FIPS 140-2 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-2](#)」を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや名前フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK で IAM または他の AWS のサービスを使用する場合も同様です。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

## IAM および AWS STSでのデータの暗号化

データ暗号化は、通常、保管時の暗号化と転送中の暗号化の2つのカテゴリに分類されます。

### 保管中の暗号化

IAM によって収集および保存されるデータは、保存時に暗号化されます。

- IAM – IAM 内で収集および保存されるデータには、IP アドレス、顧客アカウントのメタデータ、およびパスワードなどの顧客識別データが含まれます。顧客アカウントメタデータと顧客識別データは、AES 256 を使用して保管時に暗号化されるか、または SHA 256 を使用してハッシュされません。
- AWS STS – AWS STSは、サービスへの成功、誤り、および障害のリクエストを記録するサービスログを除き、お客様のコンテンツを収集しません。

### 転送中の暗号化

パスワードなどの顧客識別データは、転送に際して TLS 1.2 および 1.3 を使用して暗号化されます。すべての AWS STS エンドポイントは、転送中のデータを暗号化するために HTTPS をサポートしています。AWS STS エンドポイントのリストについては、「[リージョンとエンドポイント](#)」を参照してください。

## IAM および AWS STS のキーの管理

IAM または AWS STS を使用して暗号化キーを管理することはできません。暗号化キーの詳細については、AWS Key Management Service デベロッパーガイドの「[AWS KMS とは](#)」を参照してください。

## IAM および AWS STS のネットワーク間トラフィックプライバシー

IAM へのリクエストは、Transport Layer Security プロトコル (TLS) を使用して行う必要があります。VPC エンドポイントを使用して、AWS STS サービスへの接続を保護できます。詳細については、「[インターフェイス VPC エンドポイント](#)」を参照してください。



# AWS Identity and Access Managementでのログ記録とモニタリング

モニタリングは、AWS Identity and Access Management ( IAM )、AWS Security Token Service (AWS STS) およびその他の AWS ソリューションの信頼性、可用性、およびパフォーマンスを維持するための重要な部分です。AWSは、AWS リソースをモニタリングし、潜在的なインシデントに対応するためのいくつかのツールを提供します。

- AWS CloudTrail は、IAM および AWS STS のすべての API コールをイベントとしてキャプチャします。これには、コンソールからの呼び出しと API コールからの呼び出しも含まれます。CloudTrail を IAM および AWS STS で使用方法の詳細については、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」を参照してください。CloudTrail の詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。
- AWS Identity and Access Management Access Analyzer の機能は、外部エンティティと共有されている Amazon S3 バケットや IAM ロールなど、組織とアカウントのリソースを識別するのに役立ちます。これにより、セキュリティ上のリスクであるリソースやデータへの意図しないアクセスを特定できます。詳細については、「[IAM Access Analyzer とは](#)」を参照してください。
- Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「[Amazon CloudWatch ユーザーガイド](#)」を参照してください。
- [Amazon CloudWatch Logs] は、Amazon EC2 インスタンス、CloudTrail、およびその他のソースからのログファイルをモニタリング、保存、およびアクセスするのに役立ちます。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「[Amazon CloudWatch Logs ユーザーガイド](#)」を参照してください。

IAM に関するその他のリソースとセキュリティのベストプラクティスについては、「[AWS Identity and Access Management のセキュリティのベストプラクティスとユースケース](#)」を参照してください。

# AWS Identity and Access Management のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS Identity and Access Management コンプライアンスプログラムの一環として AWS (IAM) のセキュリティとコンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、ISO などがあります。

AWS のサービスが特定のコンプライアンスプログラムの対象範囲内にあるかどうかを確認するには、「[コンプライアンスプログラムによる対象範囲内の AWS のサービス](#)」を参照し、目的のコンプライアンスプログラムを選択してください。一般的な情報については、[AWS コンプライアンスプログラム](#)を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「[Downloading Reports in AWS Artifact](#)」を参照してください。

AWS のサービスを使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。AWS では、コンプライアンスに役立つ次のリソースを提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) - これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするためのステップを示します。
- 「[Amazon Web Services での HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ](#)」 - このホワイトペーパーは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法を説明しています。

## Note

すべての AWS のサービスが HIPAA 適格であるわけではありません。詳細については、「[HIPAA 対応サービスのリファレンス](#)」を参照してください。

- [AWS コンプライアンスのリソース](#) - このワークブックおよびガイドのコレクションは、顧客の業界と拠点に適用されるものである場合があります。
- [AWS Customer Compliance Guide](#) — コンプライアンスの観点から見た責任共有モデルを理解できます。このガイドは、AWS のサービスを保護するためのベストプラクティスを要約したものであり、複数のフレームワーク (米国標準技術研究所 (NIST)、ペイメントカード業界セキュリティ標準評議会 (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティ統制へのガイダンスがまとめられています。



- 「AWS Config デベロッパガイド」の「[ルールでのリソースの評価](#)」 - AWS Config サービスは、自社のプラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- [AWS Security Hub](#) - この AWS のサービスは、AWS 内のセキュリティ状態の包括的なビューを提供します。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「[Security Hub のコントロールリファレンス](#)」を参照してください。
- [Amazon GuardDuty](#) - この AWS のサービスは、環境をモニタリングして、疑わしいアクティビティや悪意のあるアクティビティがないか調べることで、AWS アカウント、ワークロード、コンテナ、データに対する潜在的な脅威を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- [AWS Audit Manager](#) - この AWS のサービスは、AWS の使用状況を継続的に監査して、リスクの管理方法や、規制および業界標準へのコンプライアンスの管理方法を簡素化するために役立ちます。

## AWS Identity and Access Management での耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティゾーンを中心に構築されます。AWS リージョンは物理的に分離、隔離された複数の可用性ゾーンを持ち、それらは低レイテンシー、高スループット、高冗長のネットワークで接続されています。AWS リージョンとアベイラビリティゾーンの詳細については、[AWS グローバルインフラストラクチャ](#)を参照してください。

AWS Identity and Access Management (IAM) および AWS Security Token Service (AWS STS) は、自立したリージョンベースのサービスで、全世界で利用可能です。

IAM は非常に重要な AWS のサービスです。AWS で行うすべての操作は、IAM によって認証および認可される必要があります。IAM は各リクエストを、IAM に保存されている ID とポリシーに照らし合わせて確認し、リクエストが許可されるか拒否されるかを判断します。IAM は、コントロールプレーンとデータプレーンを分離して設計されているため、予期しない障害が発生した場合でもサービスが認証されます。ロールやポリシーなど、認可に使用される IAM リソースは、コントロールプレーンに保存されます。IAM のお客様は、DeletePolicy および AttachRolePolicy のような IAM オペレーションを使用して、これらのリソースの設定を変更できます。これらの設定変更のリクエストは、コントロールプレーンに送られます。IAM のコントロールプレーンは、すべての

商用 AWS リージョンを対象とし、米国東部 (バージニア北部) リージョンに 1 つ設置されています。次に、IAM システムは、[有効化されたすべての AWS リージョン](#) の IAM データプレーンに設定変更を伝達します。IAM データプレーンは、基本的に IAM コントロールプレーンの設定データの読み取り専用レプリカです。各 AWS リージョンには、IAM データプレーンの完全に独立したインスタンスがあり、同じリージョンからのリクエストに対して認証と認可を実行します。各リージョンでは、IAM データプレーンは少なくとも 3 つのアベイラビリティゾーンに分散されており、アベイラビリティゾーンの損失を許容する十分な容量があり、顧客に障害を与えることはありません。IAM のコントロールプレーンとデータプレーンの両方が、ダウンタイムゼロを目指して構築され、すべてのソフトウェア更新とスケーリング操作が、顧客には見えない方法で実行されます。

AWS STS リクエストはデフォルトで単一のグローバルエンドポイントに送信されます。リージョナル AWS STS エンドポイントを使用して、レイテンシーを削減したり、アプリケーションの冗長性を高めたりすることができます。詳細については、「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

特定のイベントにより、ネットワークを介した AWS リージョン間の通信が中断されることがあります。しかし、グローバル IAM エンドポイントと通信できない場合でも、AWS STS は引き続き IAM プリンシパルを認証でき、IAM はリクエストを承認できます。通信を中断するイベントの具体的な詳細によって、AWS サービスにアクセスできるかどうかが決まります。ほとんどの場合、ご使用の AWS 環境で IAM 認証情報を使用し続けることができます。通信を中断させるイベントには、次のような条件が当てはまる場合があります。

## IAM ユーザーのアクセスキー

長期[アクセスキーを持つ IAM ユーザー](#)に対して、リージョン内で無期限認証が可能です。AWS Command Line Interface および API を使用する場合、AWS アクセスキーを提供することで、AWS がプログラム上のリクエストで ID を確認することができます。

### Important

[ベストプラクティス](#)として、長期的なアクセスキーではなく、[一時的な認証情報](#)でサインインすることをお勧めします。

## 一時的な認証情報

AWS STS リージョン別[サービスエンドポイント](#)では、少なくとも 24 時間、[新しい一時的な認証情報を要求する](#)ことができます。以下の API オペレーションにより、一時的な認証情報が生成されます。

- AssumeRole
- AssumeRoleWithWebIdentity
- AssumeRoleWithSAML
- GetFederationToken
- GetSessionToken

### プリンシパルおよびアクセス許可

- IAM では、プリンシパルまたは アクセス許可の追加、変更、削除ができない場合があります。
- 認証情報には、最近 IAM で適用したアクセス許可の変更が反映されていない可能性があります。詳細については、「[行った変更がすぐに表示されないことがある](#)」を参照してください。

### AWS Management Console

- IAM ユーザーとして AWS Management Console にサインインするために、リージョナルのサインインエンドポイントを使用できるかもしれません。リージョンのサインインエンドポイントの URL 形式は以下の通りです。

`https://{Account ID}.signin.aws.amazon.com/console?region={Region}`

例: `https://111122223333.signin.aws.amazon.com/console?region=us-west-2`

- [ユニバーサル第 2 ファクター \(U2F\)](#) 多要素認証 (MFA) が完了しない場合があります。

## IAM レジリエンスのためのベストプラクティス

AWS は、AWS リージョン およびアベイラビリティゾーンにレジリエンスを組み込んでいます。お客様の環境と相互作用するシステムにおいて、以下の IAM ベストプラクティスを遵守することで、そのレジリエンスを活用することができるのです。

1. デフォルトのグローバルエンドポイントではなく、AWS STS リージョン別[サービスエンドポイント](#)を使用します。
2. IAM リソースを日常的に作成または変更する重要なリソースの環境設定を確認し、既存の IAM リソースを使用するフォールバックソリューションを準備します。

# AWS Identity and Access Management でのインフラストラクチャセキュリティ

マネージドサービスである AWS Identity and Access Management は、AWS のグローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「[AWS クラウドセキュリティ](#)」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「[インフラストラクチャ保護](#)」を参照してください。

AWS が公開している API コールを使用し、ネットワーク経由で IAM にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、[AWS Security Token Service](#) (AWS STS) を使用して、一時セキュリティ認証情報を生成し、リクエストに署名することもできます。

サービスに HTTPS リクエストを直接発行できる IAM HTTPS API を使用して、プログラムにより IAM にアクセスできます。クエリ API は、セキュリティ認証情報を含む機密情報を返します。したがって、すべての API リクエストで HTTPS を使用する必要があります。HTTPS API を使用する場合は、認証情報を使用してリクエストにデジタル署名するコードを含める必要があります。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができますが、IAM ではリソースベースのアクセスポリシーがサポートされているため、ソース IP アドレスに基づく制限を含めることができます。また、IAM ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを制御することもできます。これにより、実質的に AWS ネットワーク内の特定の VPC からの特定の IAM リソースへのネットワークアクセスが分離されます。

## AWS Identity and Access Management での設定と脆弱性の分析

AWS は、ゲストオペレーティングシステム (OS) やデータベースへのパッチ適用、ファイアウォール設定、災害対策などの基本的なセキュリティタスクを処理します。これらの手順は適切な第三者によって確認され、証明されています。詳細については、以下のリソースを参照してください。

- [責任共有モデル](#)
- [Amazon Web Services: セキュリティプロセスの概要](#) (ホワイトペーパー)

次のリソースは、AWS Identity and Access Management (IAM) の構成と脆弱性の分析にも対処します。

- [AWS Identity and Access Management のコンプライアンス検証](#)
- [AWS Identity and Access Management のセキュリティのベストプラクティスとユースケース](#)

## AWS Access Analyzer 用の AWS Identity and Access Management マネージドポリシー

AWS マネージドポリシーは、AWS が作成および管理するスタンドアロンポリシーです。AWS マネージドポリシーは、多くの一般的なユースケースで権限を提供できるように設計されているため、ユーザー、グループ、ロールへの権限の割り当てを開始できます。

AWS マネージドポリシーは、ご利用の特定のユースケースに対して最小特権の権限を付与しない場合があることにご注意ください。AWS のすべてのお客様が使用できるようになるのを避けるためです。ユースケース別に [カスタマーマネージドポリシー](#) を定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS マネージドポリシーで定義したアクセス権限は変更できません。AWS が AWS マネージドポリシーに定義されている権限を更新すると、更新はポリシーがアタッチされているすべてのプリンシパルアイデンティティ (ユーザー、グループ、ロール) に影響します。新しい AWS のサービスを起動するか、既存のサービスで新しい API オペレーションが使用可能になると、AWS が AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

## IAMReadOnlyAccess

IAM リソースへの読み取り専用アクセスを許可するには、IAMReadOnlyAccess 管理ポリシーを使用します。このポリシーは、すべての IAM リソースを取得して一覧表示するアクセス許可を付与します。これにより、ユーザー、グループ、ロール、ポリシー、ID プロバイダー、および MFA デバイスについての詳細と、アクティビティレポートを表示できます。リソースを作成または削除したり、IAM Access Analyzer リソースにアクセスしたりする機能は含まれません。この[ポリシー](#)がサポートしているサービスとアクションの、詳細なリストに対するポリシーを表示します。

## IAMUserChangePassword

この IAMUserChangePassword 管理ポリシーを使用して、パスワード変更を IAM ユーザーに許可します。

IAM ユーザーが IAM アカウントのパスワードを変更できるように、IAM のアカウント設定とパスワードポリシーを設定します。このアクションを許可すると、IAM は次のポリシーを各ユーザーに添付します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword"
      ],
      "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

## IAMAccessAnalyzerFullAccess

IAMAccessAnalyzerFullAccess AWS管理ポリシーを使用して、管理者が IAM Access Analyzer にアクセスできるようにします。

### アクセス権限のグループ化

このポリシーは、提供された一連の許可に基づくステートメントごとにグループ化されます。

- IAM Access Analyzer — IAM Access Analyzer のすべてのリソースに対する完全な管理アクセス許可を許可します。
- サービスリンクロールの作成 — 管理者が[サービスにリンクされたロール](#)を作成できるようにします。これにより、IAM AccessAnalyzer がユーザーに代わって他のサービスのリソースを分析できるようになります。このアクセス許可では、IAM Access Analyzer が使用するサービスリンクロールのみを作成できます。
- AWS Organizations — 管理者による AWS Organizations の組織への IAM Access Analyzer の使用を許可します。AWS Organizations でIAM Access Analyzer の[信頼できるアクセスを有効にする](#)と、管理アカウントのメンバーは組織全体の調査結果を表示できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "access-analyzer:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "access-analyzer.amazonaws.com"
        }
      }
    }
  ],
  {
```



```
"Effect": "Allow",
"Action": [
  "organizations:DescribeAccount",
  "organizations:DescribeOrganization",
  "organizations:DescribeOrganizationalUnit",
  "organizations:ListAccounts",
  "organizations:ListAccountsForParent",
  "organizations:ListAWSServiceAccessForOrganization",
  "organizations:ListChildren",
  "organizations:ListDelegatedAdministrators",
  "organizations:ListOrganizationalUnitsForParent",
  "organizations:ListParents",
  "organizations:ListRoots"
],
"Resource": "*"
}
]
}
```

## IAMAccessAnalyzerReadOnlyAccess

IAM Access Analyzer に対して読み取り専用アクセスを許可するには、IAMAccessAnalyzerReadOnlyAccess AWS 管理ポリシーを使用します。

AWS Organizations 用の IAM Access Analyzer への読み取り専用アクセスも許可するには、[IAMAccessAnalyzerFullAccess](#) AWS 管理ポリシーからの Describe および List アクションを許可するカスタマー管理ポリシーを作成します。

### サービスレベルのアクセス許可

このポリシーは IAM Access Analyzer への読み取り専用アクセスを提供します。このポリシーには、他のサービス権限は含まれていません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAccessAnalyzerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "access-analyzer:CheckAccessNotGranted",
        "access-analyzer:CheckNoNewAccess",
```



```
        "access-analyzer:Get*",
        "access-analyzer:List*",
        "access-analyzer:ValidatePolicy"
    ],
    "Resource": "*"
}
]
```

## AccessAnalyzerServiceRolePolicy

IAM エンティティに AccessAnalyzerServiceRolePolicy をアタッチすることはできません。このポリシーは、ユーザーに代わって IAM Access Analyzer がアクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「AWS Identity and Access Management Access Analyzer」の「[サービスにリンクされたロールの使用](#)」を参照してください。

### アクセス許可グルーピング

このポリシーは、複数の AWS のサービスからのリソースメタデータを分析するための IAM Access Analyzer へのアクセスを許可します。

- Amazon DynamoDB – DynamoDB ストリームとテーブルを表示するアクセス許可を付与します。
- Amazon Elastic Compute Cloud – IP アドレス、スナップショット、および VPC を記述するためのアクセス許可を許可します。
- Amazon Elastic Container Registry – イメージリポジトリを記述し、リポジトリポリシーを取得するためのアクセス許可を許可します。
- Amazon Elastic File System – Amazon EFS ファイルシステムの記述と、Amazon EFS ファイルシステムのリソースレベルのポリシーを表示するためのアクセス許可を許可します。
- AWS Identity and Access Management – 指定されたロールに関する情報を取得して、指定されたパスプレフィックスを持つ IAM ロールをリストするためのアクセス許可を許可します。ユーザー、ユーザーグループ、ログインプロフィール、アクセスキー、およびサービスの最終アクセスデータに関する情報を取得するアクセス許可を付与します。
- AWS Key Management Service – KMS キー、およびそのキーポリシーとグラントに関する詳細情報を表示するためのアクセス許可を許可します。
- AWS Lambda – Lambda のエイリアス、関数、およびレイヤーに関する情報を表示するためのアクセス許可を許可します。
- AWS Organizations – Organizations に対するアクセス許可を許可し、AWS 組織内での信頼ゾーンとしてのアナライザーの作成を許可します。

- Amazon Relational Database Service – Amazon RDS DB スナップショットと Amazon RDS DB クラスター スナップショットに関する詳細情報を表示するアクセス許可を許可します。
- Amazon Simple Storage Service – Amazon S3 Express One ストレージクラスを使用する Amazon S3 Access Points、バケット、Amazon S3 ディレクトリバケットに関する詳細情報を表示するアクセス許可を付与します。
- AWS Secrets Manager – シークレットと、シークレットにアタッチされているリソースポリシーに関する詳細情報を表示するためのアクセス許可を許可します。
- Amazon Simple Notification Service – トピックに関する詳細情報を表示するためのアクセス許可を許可します。
- Amazon Simple Queue Service – 指定されたキューに関する詳細情報を表示するためのアクセス許可を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessAnalyzerServiceRolePolicy",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetResourcePolicy",
        "dynamodb:ListStreams",
        "dynamodb:ListTables",
        "ec2:DescribeAddresses",
        "ec2:DescribeByoipCidrs",
        "ec2:DescribeSnapshotAttribute",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:GetSnapshotBlockPublicAccessState",
        "ecr:DescribeRepositories",
        "ecr:GetRepositoryPolicy",
        "elasticfilesystem:DescribeFileSystemPolicy",
        "elasticfilesystem:DescribeFileSystems",
        "iam:GetRole",
        "iam:ListEntitiesForPolicy",
        "iam:ListRoles",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:GetGroup",
        "iam:GenerateServiceLastAccessedDetails",

```

```
"iam:GetServiceLastAccessedDetails",
"iam:ListAccessKeys",
"iam:GetLoginProfile",
"iam:GetAccessKeyLastUsed",
"iam:ListRolePolicies",
"iam:GetRolePolicy",
"iam:ListAttachedRolePolicies",
"iam:ListUserPolicies",
"iam:GetUserPolicy",
"iam:ListAttachedUserPolicies",
"iam:GetPolicy",
"iam:GetPolicyVersion",
"iam:ListGroupsForUser",
"kms:DescribeKey",
"kms:GetKeyPolicy",
"kms:ListGrants",
"kms:ListKeyPolicies",
"kms:ListKeys",
"lambda:GetFunctionUrlConfig",
"lambda:GetLayerVersionPolicy",
"lambda:GetPolicy",
"lambda:ListAliases",
"lambda:ListFunctions",
"lambda:ListLayers",
"lambda:ListLayerVersions",
"lambda:ListVersionsByFunction",
"organizations:DescribeAccount",
"organizations:DescribeOrganization",
"organizations:DescribeOrganizationalUnit",
"organizations:ListAccounts",
"organizations:ListAccountsForParent",
"organizations:ListAWSServiceAccessForOrganization",
"organizations:ListChildren",
"organizations:ListDelegatedAdministrators",
"organizations:ListOrganizationalUnitsForParent",
"organizations:ListParents",
"organizations:ListRoots",
"rds:DescribeDBClusterSnapshotAttributes",
"rds:DescribeDBClusterSnapshots",
"rds:DescribeDBSnapshotAttributes",
"rds:DescribeDBSnapshots",
"s3:DescribeMultiRegionAccessPointOperation",
"s3:GetAccessPoint",
"s3:GetAccessPointPolicy",
```

```

    "s3:GetAccessPointPolicyStatus",
    "s3:GetAccountPublicAccessBlock",
    "s3:GetBucketAcl",
    "s3:GetBucketLocation",
    "s3:GetBucketPolicyStatus",
    "s3:GetBucketPolicy",
    "s3:GetBucketPublicAccessBlock",
    "s3:GetMultiRegionAccessPoint",
    "s3:GetMultiRegionAccessPointPolicy",
    "s3:GetMultiRegionAccessPointPolicyStatus",
    "s3:ListAccessPoints",
    "s3:ListAllMyBuckets",
    "s3:ListMultiRegionAccessPoints",
    "s3express:GetBucketPolicy",
    "s3express:ListAllMyDirectoryBuckets",
    "sns:GetTopicAttributes",
    "sns:ListTopics",
    "secretsmanager:DescribeSecret",
    "secretsmanager:GetResourcePolicy",
    "secretsmanager:ListSecrets",
    "sqs:GetQueueAttributes",
    "sqs:ListQueues"
  ],
  "Resource": "*"
}
]
}

```

## IAM と IAM Access Analyzer による AWS 管理ポリシーの更新

サービスが変更の追跡を開始してからの、IAM と AWS 管理ポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知については、「RSS feed on the IAM and IAM Access Analyzer Document history」ページの RSS フィードを購読してください。

変更	説明	日付
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	IAM Access Analyzer は、IAM ユーザーポリシーとロールポリシーに関する情報を取得	2024 年 5 月 8 日

変更	説明	日付
	するアクセス許可のサポートを AccessAnalyzerServiceRolePolicy のサービスレベルのアクセス許可に追加しました。	
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	IAM Access Analyzer は、Amazon EC2 スナップショットのブロックパブリックアクセスに関する現況を取得するためのアクセス許可を、AccessAnalyzerServiceRolePolicy のサービスレベルのアクセス許可に追加しました。	2024 年 1 月 23 日
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	IAM Access Analyzer は、DynamoDB ストリームとテーブルのサポートを、AccessAnalyzerServiceRolePolicy のサービスレベルのアクセス許可に追加しました。	2024 年 1 月 11 日
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	IAM Access Analyzer は、Amazon S3 ディレクトリバケットのサポートを、AccessAnalyzerServiceRolePolicy のサービスレベルのアクセス許可に追加しました。	2023 年 12 月 1 日

変更	説明	日付
<a href="#">IAMAccessAnalyzerReadOnlyAccess</a> – アクセス許可を追加	<p>IAM Access Analyzer で、ポリシーの更新によって追加のアクセス権限が付与されるかどうかをユーザーが確認できるようにするためのアクセス許可が追加されました。</p> <p>このアクセス許可は、IAM Access Analyzer でポリシーチェックを実行するために必要です。</p>	2023 年 11 月 26 日
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	<p>IAM Access Analyzer で、以下のアクションをサポートするための IAM アクションが AccessAnalyzerServiceRolePolicy のサービスレベルアクセス許可に追加されました。</p> <ul style="list-style-type: none"><li>• ポリシーのエンティティをリストする</li><li>• サービスの最終アクセス時間の詳細を生成する</li><li>• アクセスキー情報をリストする</li></ul>	2023 年 11 月 26 日

変更	説明	日付
<a href="#">AccessAnalyzerServiceRolePolicy</a> – アクセス許可を追加	<p>IAM Access Analyzer は、以下のリソースタイプのサポートを AccessAnalyzerServiceRolePolicy のサービスレベルアクセス許可に追加しました。</p> <ul style="list-style-type: none"><li>• Amazon EBS ボリュームスナップショット</li><li>• Amazon ECR リポジトリ</li><li>• Amazon EFS ファイルシステム</li><li>• Amazon RDS DB スナップショット</li><li>• Amazon RDS DB クラスタースナップショット</li><li>• Amazon SNS トピック</li></ul>	2022 年 10 月 25 日
<a href="#">AccessAnalyzerServiceRolePolicy</a> — アクセス許可を追加済み	<p>IAM Access Analyzer は、AccessAnalyzerServiceRolePolicy のサービスレベル権限に対して <code>lambda:GetFunctionUrlConfig</code> アクションを追加しました。</p>	2022 年 4 月 6 日
<a href="#">AccessAnalyzerServiceRolePolicy</a> — アクセス許可を追加済み	<p>IAM Access Analyzer には、マルチリージョンアクセスポイントに関連付けられたメタデータを分析する新しい Amazon S3 アクションが追加されました。</p>	2021 年 9 月 2 日

変更	説明	日付
<a href="#">IAMAccessAnalyzerReadOnlyAccess</a> – アクセス許可を追加	<p>IAM Access Analyzer は、検証にポリシーチェックを使用できるようにする <code>ValidatePolicy</code> アクセス許可を付与する新しいアクションを追加しました。</p> <p>このアクセス許可は、IAM Access Analyzer でポリシーチェックを実行するために必要です。</p>	2021 年 3 月 16 日
IAM Access Analyzer は変更の追跡を開始しました	IAM Access Analyzer が AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 3 月 1 日



# AWS Identity and Access Management Access Analyzer を使用する

AWS Identity and Access Management Access Analyzer には、次の機能が用意されています。

- IAM Access Analyzer の外部アクセスアナライザーは、外部エンティティと共有されている組織およびアカウントの [リソースを特定する](#) のに役立ちます。
- IAM Access Analyzer の未使用のアクセスアナライザーは、組織およびアカウント内の [使用されていないアクセス権限を特定する](#) のに役立ちます。
- IAM Access Analyzer は、ポリシーの文法および AWS ベストプラクティスに照らして [IAM ポリシーを検証します](#)。
- IAM Access Analyzer のカスタムポリシーチェックは、[指定したセキュリティ基準に照らして IAM ポリシーを検証する](#) のに役立ちます。
- IAM Access Analyzer は、AWS CloudTrail ログでのアクセスアクティビティに基づいた [IAM ポリシーを生成](#) します。

## 外部エンティティと共有されているリソースを識別する

IAM Access Analyzer の機能は、外部エンティティと共有されている Amazon S3 バケットや IAM ロールなど、組織とアカウントのリソースを識別するのに役立ちます。これにより、セキュリティ上のリスクであるリソースやデータへの意図しないアクセスを特定できます。IAM Access Analyzer は、ロジックベースの推論を使用して AWS 環境のリソースベースのポリシーを分析することにより、外部プリンシパルと共有されているリソースを識別します。アカウントの外部で共有されているリソースのインスタンスごとに、IAM Access Analyzer は結果を生成します。結果には、アクセスと付与される外部プリンシパルに関する情報が含まれます。結果を確認して、アクセスが意図的で安全なものであるか、またはアクセスが意図しないものでセキュリティ上のリスクがあるかを判断できます。IAM Access Analyzer の結果を使用すると、外部エンティティと共有されているリソースを識別できるだけでなく、リソースのアクセス許可を展開する前に、ポリシーがリソースへのパブリックアクセスおよびクロスアカウントアクセスにどのように影響するかをプレビューできます。検出結果は視覚的な概要ダッシュボードにまとめられます。ダッシュボードでは、パブリックアクセスとクロスアカウントアクセスの検出結果が分割して強調表示され、それぞれの検出結果がリソースタイプごとに分類されます。ダッシュボードの詳細については、「[IAM Access Analyzer の検出結果ダッシュボードの表示](#)」を参照してください。

**Note**

外部エンティティとは、別の AWS アカウント、ルートユーザー、IAM ユーザーまたはロール、フェデレーテッドユーザー、AWS のサービス、匿名ユーザー、その他フィルターの作成に使用できるエンティティです。詳細については、「[AWS IAM JSON ポリシーエレメント: Principal](#)」を参照してください。

IAM Access Analyzer を有効にしたら、組織全体またはアカウントのアナライザーを作成します。選択した組織またはアカウントは、アナライザーの信頼ゾーンと呼ばれます。アナライザーは、信頼ゾーン内で[サポートされているすべてのリソース](#)をモニタリングします。信頼ゾーン内のプリンシパルによるリソースへのアクセスは、信頼できると見なされます。有効にすると、IAM Access Analyzer は信頼ゾーン内のすべてのサポートされているリソースに適用されているポリシーを分析します。最初の分析後、IAM Access Analyzer はこれらのポリシーを定期的に分析します。新しいポリシーが追加されるか、既存のポリシーが変更されると、IAM Access Analyzer は約 30 分以内に新しいポリシーまたは更新されたポリシーを分析します。

IAM Access Analyzer は、ポリシーの分析時に、信頼ゾーン外の外部プリンシパルに対してアクセスを許可するポリシーを見つけると、結果を生成します。各結果には、適切なアクションを実行できるように、リソース、リソースにアクセスできる外部エンティティ、および付与されているアクセス許可に関する詳細が含まれています。結果に含まれている詳細を表示して、リソースへのアクセスが意図的であるか、解決すべき潜在的なリスクであるかを判断できます。リソースにポリシーを追加するか、既存のポリシーを更新すると、IAM Access Analyzer はポリシーを分析します。また、IAM Access Analyzer は、すべてのリソースベースのポリシーを定期的に分析します。

特定の条件下でまれに、IAM Access Analyzer がポリシーの追加や更新の通知を受信しないことがあり、これにより、生成された検出結果に遅延が発生する可能性があります。Amazon S3 バケットに関連付けられたマルチリージョンアクセスポイントを作成または削除する場合、またはマルチリージョンアクセスポイントのポリシーを更新する場合、IAM Access Analyzer で検出結果を生成または解決するのに最大 6 時間かかる場合があります。また、AWS CloudTrail ログ配信で配信の問題がある場合、ポリシーを変更しても、検出結果でレポートされたリソースの再スキャンはトリガーされません。このような場合、IAM Access Analyzer は、次の定期スキャン時 (24 時間以内) に新しいポリシーまたは更新されたポリシーを分析します。結果でレポートされたアクセスの問題が、ポリシーを変更することで解決されることを確認する場合は、結果の詳細ページの [Rescan] (再スキャン) リンクを使用するか、IAM Access Analyzer API の [StartResourceScan](#) オペレーションを使用して、結果でレポートされたリソースを再スキャンできます。詳細については、「[結果の解決](#)」を参照してください。

**⚠ Important**

IAM Access Analyzer は、それが有効になっている AWS リージョンでのみ、リソースに適用されているポリシーを分析します。AWS 環境のすべてのリソースをモニタリングするには、サポートされている AWS リソースを使用している各リージョンでアナライザーを作成して IAM Access Analyzer を有効にする必要があります。

IAM Access Analyzer は、以下のリソースタイプを分析します。

- [Amazon Simple Storage Service バケット](#)
- [Amazon Simple Storage Service ディレクトリバケット](#)
- [AWS Identity and Access Management ロール](#)
- [AWS Key Management Service キー](#)
- [AWS Lambda の関数とレイヤー](#)
- [Amazon Simple Queue Service キュー](#)
- [AWS Secrets Manager シークレット](#)
- [Amazon Simple Notification Service トピック](#)
- [Amazon Elastic Block Store ボリュームスナップショット](#)
- [Amazon Relational Database Service DB スナップショット](#)
- [Amazon Relational Database Service DB クラスタースナップショット](#)
- [Amazon Elastic Container Registry リポジトリ](#)
- [Amazon Elastic File System ファイルシステム](#)
- [Amazon DynamoDB Streams](#)
- [Amazon DynamoDB テーブル](#)

## IAM ユーザーおよびロールに付与された未使用のアクセスを特定する

IAM Access Analyzer は、AWS 組織およびアカウント内の使用されていないアクセス権限を特定して確認するのに役立ちます。IAM Access Analyzer は、AWS 組織およびアカウント内のすべての IAM ロールとユーザーを継続的にモニタリングし、未使用のアクセスに関する検出結果を生成します。検出結果では、未使用のロール、IAM ユーザーの未使用のアクセスキー、IAM ユーザーの未使

用のパスワードが強調表示されます。これらの検出結果では、アクティブな IAM ロールとユーザーで未使用のサービスとアクションが表示されます。

外部のアクセスアナライザーと未使用のアクセスアナライザーの両方の検出結果が、視覚的な概要ダッシュボードにまとめられます。ダッシュボードでは、検出結果が最も多い AWS アカウントが強調表示され、その結果がタイプごとに分類されます。ダッシュボードのページの詳細については、「[IAM Access Analyzer の検出結果ダッシュボードの表示](#)」を参照してください。

IAM Access Analyzer では、AWS 組織およびアカウント内のすべてのロールについて、最終アクセス時間情報を確認することができ、使用されていないアクセス権限を特定するのに役立ちます。IAM アクションの最終アクセス時間情報は、AWS アカウント内のロールで使用されていないアクションを特定するのに役立ちます。詳細については、「[最終アクセス情報を使用した AWS のアクセス許可の調整](#)」を参照してください。

## AWS ベストプラクティスに照らしてポリシーを検証する

IAM Access Analyzer のポリシー検証で提供される基本的なポリシーチェックを使用して、IAM [ポリシーの文法](#)や [AWS ベストプラクティス](#)に照らしてポリシーを検証できます。IAM コンソールの AWS CLI、AWS API、または JSON ポリシーエディタを使用して、ポリシーを作成または編集できます。セキュリティ警告、エラー、一般的な警告、ポリシーの提案を含むポリシー検証チェックの結果を表示できます。これらの検出結果から、機能的で AWS ベストプラクティスに準拠したポリシーの作成に役立つ実用的な推奨事項が示されます。ポリシー検証を使用したポリシーの検証の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

## 指定したセキュリティ標準に照らしてポリシーを検証する

IAM Access Analyzer カスタムポリシーチェックを使用して、指定したセキュリティ標準に照らしてポリシーを検証できます。IAM コンソールの AWS CLI、AWS API、または JSON ポリシーエディタを使用して、ポリシーを作成または編集できます。コンソールでは、更新したポリシーで新しいアクセス権限が付与されるかどうかを、既存のバージョンとの比較で確認できます。AWS CLI および AWS API を使用して、重要と考える特定の IAM アクションがポリシーによって許可されていないことを確認することもできます。これらのチェックでは、新しいアクセスを許可するポリシーステートメントが強調表示されます。ポリシーステートメントを更新し、ポリシーがセキュリティ標準に準拠するまでチェックを再実行できます。カスタムポリシーチェックを使用したポリシーの検証の詳細については、「[IAM Access Analyzer カスタムポリシーチェック](#)」を参照してください。

## ポリシーの生成

IAM Access Analyzer は AWS CloudTrail ログを分析して、IAM エンティティ (ユーザーまたはロール) が使用しているアクションとサービスを特定します。次に、そのアクセスアクティビティに基づく IAM ポリシーを生成します。生成されたポリシーを使用して、エンティティのアクセス許可を IAM ユーザーまたはロールにアタッチすることで、そのエンティティのアクセス許可を絞り込むことができます。IAM Access Analyzer を使用したポリシーの生成の詳細については、「[IAM Access Analyzer ポリシーの生成](#)」を参照してください。

## IAM Access Analyzer の価格設定

IAM Access Analyzer では、1 か月あたりにアナライザーごとに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。

- 作成した未使用のアクセスアナライザーに対して料金が発生します。
- 複数のリージョンにまたがる未使用のアクセスアナライザーを作成すると、アナライザーごとに料金が発生します。
- サービスにリンクされたロールは、未使用のアクセスアクティビティについては分析されず、分析される IAM ロールの総数には含まれません。

IAM Access Analyzer では、新しいアクセスをチェックするために IAM Access Analyzer に対して行った API リクエストの数に基づいて、カスタムポリシーチェックに対する料金が発生します。

IAM Access Analyzer の料金および価格設定の完全なリストについては、「[IAM Access Analyzer pricing](#)」を参照してください。

請求を表示するには、[AWS Billing and Cost Management コンソール](#)で請求およびコスト管理ダッシュボードに移動します。請求書には、料金の明細が記載された使用状況レポートへのリンクが記載されています。AWS アカウントの請求の詳細については、「[AWS Billing ユーザーガイド](#)」を参照してください。

AWSの請求、アカウント、イベントについてのご質問は、[AWS Supportにお問い合わせください](#)。

## 外部アクセスと未使用のアクセスに関する検出結果

IAM Access Analyzer は、AWS アカウントまたは組織の外部アクセスと未使用のアクセスに関する検出結果を生成します。外部アクセスの場合、IAM Access Analyzer は、信頼ゾーン内にはないプリン



シパルに対して信頼ゾーン内のリソースへのアクセスを許可する、リソースベースのポリシーの各インスタンスの結果を生成します。外部アクセスアナライザーを作成する際には、分析する組織または AWS アカウントを選択します。アナライザー用に選択した組織またはアカウント内のプリンシパルは、信頼済みと見なされます。同じ組織またはアカウント内のプリンシパルは信頼されるため、組織またはアカウント内のリソースとプリンシパルは、アナライザーの信頼ゾーンを構成します。信頼ゾーン内での共有は安全と見なされるため、IAM Access Analyzer は結果を生成しません。たとえば、アナライザーの信頼ゾーンとして組織を選択した場合、その組織内のすべてのリソースとプリンシパルは信頼ゾーン内にあります。ある組織メンバーアカウント内の Amazon S3 バケットへのアクセス許可を、別の組織メンバーアカウント内のプリンシパルに付与した場合、IAM Access Analyzer は結果を生成しません。ただし、組織のメンバーではないアカウントのプリンシパルにアクセス許可を付与すると、IAM Access Analyzer によって結果が生成されます。

IAM Access Analyzer は、AWS 組織およびアカウントで付与されている未使用のアクセス権限に関する検出結果も生成します。未使用のアクセスアナライザーを作成すると、IAM Access Analyzer は AWS 組織およびアカウント内のすべての IAM ロールとユーザーを継続的にモニタリングし、使用されていないアクセス権限に関する検出結果を生成します。IAM Access Analyzer は、未使用のアクセスについて以下のタイプの検出結果を生成します。

- 未使用のロール – 指定された使用期間内にアクセスアクティビティがないロール。
- 未使用の IAM ユーザーアクセスキーとパスワード – IAM ユーザーに属している認証情報。ユーザーは、これを使用して AWS アカウントにアクセスできます。
- 未使用のアクセス許可 – 指定された使用期間内にロールによって使用されなかったサービスレベルおよびアクションレベルのアクセス許可。IAM Access Analyzer は、ロールにアタッチされた ID ベースのポリシーを使用して、そのロールがアクセスできるサービスとアクションを決定します。IAM Access Analyzer は、すべてのサービスレベルのアクセス許可について、未使用のアクセス許可のレビューをサポートしています。未使用のアクセスの検出結果でサポートされるアクションレベルのアクセス許可の完全なリストについては、「[IAM アクションの最終アクセス情報サービスとアクション](#)」を参照してください。

#### Note

IAM Access Analyzer では、外部アクセスの検出結果については無料で提供されますが、未使用のアクセスの結果については 1 か月あたりにアナライザーごとに分析された IAM ロールとユーザーの数に基づいて料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

## トピック

- [IAM Access Analyzer の結果](#)
- [AWS Identity and Access Management Access Analyzer の結果の使用開始](#)
- [IAM Access Analyzer の検出結果ダッシュボードの表示](#)
- [結果を使用する](#)
- [調査結果を確認する](#)
- [検出結果のフィルタリング](#)
- [結果のアーカイブ](#)
- [結果の解決](#)
- [外部アクセスの IAM Access Analyzer リソースタイプ](#)
- [IAM Access Analyzer の設定](#)
- [アーカイブルール](#)
- [Amazon EventBridge を使用した AWS Identity and Access Management Access Analyzer のモニタリング](#)
- [Access Analyzer と AWS Security Hub を統合する](#)
- [AWS CloudTrail を使用した IAM Access Analyzer API コールのログ記録](#)
- [IAM Access Analyzer フィルターキーにアクセスする](#)
- [AWS Identity and Access Management Access Analyzer のサービスにリンクされたロールの使用](#)

## IAM Access Analyzer の結果

このトピックでは、AWS リソースへのアクセスを IAM Access Analyzer でモニタリングする方法の理解に役立つ IAM Access Analyzer の概念と用語について説明します。

### 外部アクセス

外部アクセスアナライザーの場合、AWS Identity and Access Management Access Analyzer は [Zelkova](#) 上に構築されており、IAM ポリシーを同等の論理ステートメントに変換し、問題に対して汎用および特殊な論理ソルバー (充足可能性モジュロ理論) のスイートを実行します。IAM Access Analyzer は、ポリシーが許可する動作のクラスの特徴を明確にするクエリを使用して Zelkova を繰り返しポリシーに適用します。充足可能性モジュロ理論の詳細については、「[Satisfiability Modulo Theories](#)」を参照してください。

外部アクセスアナライザーの場合、IAM Access Analyzer は、外部エンティティが信頼ゾーン内のリソースにアクセスしたかどうかを確認するためにアクセスログを調べることはありません。リソースベースのポリシーがリソースへのアクセスを許可している場合は、リソースが外部エンティティからアクセスされていなくても、Access Analyzer は検出結果を生成します。また、IAM Access Analyzer は、外部アカウントの状態を決定する際に考慮しません。つまり、アカウント 111122223333 が Amazon S3 バケットにアクセスできることを示していても、そのアカウントのユーザー、ロール、サービスコントロールポリシー (SCP)、その他の関連設定の状態については一切関知しません。これは、顧客のプライバシーのためです (IAM Access Analyzer は誰が他のアカウントを所有しているかについて関知しません)。また、セキュリティのためでもあります。アカウントが IAM Access Analyzer の顧客によって所有されていない場合、リソースにアクセスできるプリンシパルがアカウント内に現在なくても、外部エンティティがリソースにアクセスできることを知っておくことは依然として重要です。

IAM Access Analyzer が考慮する IAM 条件キーは、外部ユーザーが直接影響を与えることができないものか、それ以外の方法で承認に影響を及ぼすものに限られます。IAM Access Analyzer が考慮する条件キーの例については、「[IAM Access Analyzer フィルターキーにアクセスする](#)」を参照してください。

現在、IAM Access Analyzer は AWS のサービスプリンシパルまたは内部サービスアカウントからの結果を報告しません。まれに IAM Access Analyzer がポリシーステートメントが外部エンティティにアクセスを許可するかどうかを十分に判断できない場合は、誤検知結果を宣言する側でエラーになります。IAM Access Analyzer は、アカウント内のリソース共有の包括的なビューを提供するように設計されており、偽陰性は最小限に抑えられます。

## 未使用のアクセス

リソースの外部アクセスの検出結果を生成するアナライザーを既に作成している場合でも、ロールに関する未使用のアクセスの結果用のアナライザーを作成する必要があります。アナライザーを作成すると、IAM Access Analyzer はアクセスアクティビティを確認して、使用されていないアクセス権限を特定します。IAM Access Analyzer では、AWS 組織およびアカウント内のすべてのロール、ユーザーアクセスキー、ユーザーパスワードについて、最終アクセス時間情報を確認することができ、未使用のアクセスを特定するのに役立ちます。アクティブな IAM ロールとユーザーについては、IAM Access Analyzer は IAM サービスとアクションの最終アクセス時間情報を使用して、未使用のアクセス許可を特定します。未使用のアクセスアナライザーを使用して、AWS 組織レベルやアカウントレベルでレビュープロセスを拡張できます。アクションの最終アクセス時間情報を使用して、個々のロールをより詳細に調査できます。



## [要約] ダッシュボード

外部アクセスと未使用のアクセスの両方について、IAM Access Analyzer は検出結果を概要ダッシュボードにまとめます。外部アクセスの場合、概要ダッシュボードでは、パブリックアクセスとクロスアカウントアクセスの検出結果の違いが強調表示され、結果はリソースタイプごとに分類されています。未使用のアクセスの場合、ダッシュボードでは、検出結果が最も多い AWS アカウントが強調表示され、その結果がタイプごとに分類されます。外部アクセスまたは未使用のアクセス用のアナライザーを作成すると、IAM Access Analyzer は、未使用のアクセス許可を持つロールに焦点を当てて、新しい検出結果をダッシュボードに自動的に追加します。

## AWS Identity and Access Management Access Analyzer の結果の使用開始

このトピックでは、AWS Identity and Access Management Access Analyzer を使用および管理するために必要な条件と、IAM Access Analyzer を有効にする方法について説明します。IAM Access Analyzer のサービスにリンクされたロールの詳細については、「[AWS Identity and Access Management Access Analyzer のサービスにリンクされたロールの使用](#)」を参照してください。

### IAM Access Analyzer を使用するために必要なアクセス許可

IAM Access Analyzer を正常に設定して使用するには、お使いのアカウントに対して必要なアクセス許可が付与されている必要があります。

#### IAM Access Analyzer の AWS 管理ポリシー

AWS Identity and Access Management Access Analyzer で提供される AWS 管理ポリシーを使用して、すぐに作業を開始できます。

- [IAMAccessAnalyzerFullAccess](#) - 管理者に IAM Access Analyzer へのフルアクセスを許可します。また、このポリシーによって、IAM Access Analyzer がアカウント内または AWS 組織のリソースを分析できるようにするために必要なサービスリンクされたロールを作成することもできます。
- [IAMAccessAnalyzerReadOnlyAccess](#) - IAM Access Analyzer への読み取り専用アクセスを許可します。IAM ID (ユーザー、ユーザーのグループ、またはロール) にポリシーを追加して、それらが表示できるようにする必要があります。

#### IAM Access Analyzer で定義したリソース

IAM Access Analyzer によって定義されるリソースを表示するには、「サービス認証リファレンス」の「[IAM Access Analyzer で定義されるリソースタイプ](#)」を参照してください。

## 必要な IAM Access Analyzer サービスの許可

IAM Access Analyzer は、`AWSServiceRoleForAccessAnalyzer` という名前の IAM サービスリンクロール (SLR) を使用します。この SLR は、リソースベースのポリシーを使用して AWS リソースを分析し、ユーザーに代わって未使用のアクセスを分析するための読み取り専用アクセスをサービスに付与します。以下のシナリオで、サービスによってアカウント内のロールが作成されます。

- 自分のアカウントを信頼ゾーンとして持つ外部アクセスアナライザーを作成します。
- 自分のアカウントを選択アカウントとして持つ未使用のアクセスアナライザーを作成します。

詳細については、「[AWS Identity and Access Management Access Analyzer のサービスにリンクされたロールの使用](#)」を参照してください。

### Note

IAM Access Analyzer はリージョンに基づきます。外部アクセスの場合、IAM Access Analyzer は、各リージョンで個別に有効にする必要があります。

未使用のアクセスの場合、アナライザーの検出結果はリージョンによって変わりません。リソースがある各リージョンにアナライザーを作成する必要はありません。

場合によっては、IAM Access Analyzer で外部アクセスアナライザーまたは未使用のアクセスアナライザーを作成した後に、[結果] ページまたはダッシュボードを読み込んでも検出結果や概要が表示されないことがあります。これは、コンソールで結果の反映が遅延しているためと考えられます。必要に応じてブラウザを手動で更新するか、後で検出結果や概要が表示されるか確認してください。それでも外部アクセスアナライザーの検出結果が表示されない場合は、外部エンティティからアクセスできるサポート対象のリソースがアカウント内にないことが原因です。外部エンティティにアクセスを許可するポリシーをリソースに適用すると、IAM Access Analyzer で結果が生成されます。

### Note

外部アクセスアナライザーの場合、ポリシーが変更されてから、IAM Access Analyzer がリソースを分析して、新しい外部アクセスの結果を生成したり、リソースへのアクセスに関する既存の結果を更新したりするまでに、最大で 30 分かかる場合があります。外部アクセスアナライザーと未使用のアクセスアナライザーのどちらの場合も、検出結果の更新がすぐにダッシュボードに反映されないことがあります。

検出結果ダッシュボードを表示するために必要な IAM Access Analyzer アクセス許可

[IAM Access Analyzer の検出結果ダッシュボード](#)を表示するには、使用するアカウントに対して、以下の必要なアクションを実行するためのアクセス許可が付与されている必要があります。

- [GetAnalyzer](#)
- [ListAnalyzers](#)
- `GetFindingsStatistics`

IAM Access Analyzer によって定義されるすべてのアクションを表示するには、「サービス認証リファレンス」の「[IAM Access Analyzer で定義されるアクション](#)」を参照してください。

## IAM Access Analyzer の有効化

AWS アカウントを信頼ゾーンとして持つ外部アクセスアナライザーを作成する方法

リージョンで外部アクセスアナライザーを有効にするには、そのリージョンでアナライザーを作成する必要があります。外部アクセスアナライザーは、リソースへのアクセスをモニタリングするリージョンごとに作成する必要があります。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。
3. [アナライザー設定] を選択します。
4. [Create analyzer] (アナライザーの作成) を選択します。
5. [分析] セクションで [外部アクセス分析] を選択します。
6. [アナライザーの詳細] セクションで、表示されているリージョンが、IAM Access Analyzer を有効にするリージョンであることを確認します。
7. アナライザーの名前を入力します。
8. アナライザーの信頼ゾーンとして [現在の AWS アカウント] を選択します。

### Note

アカウントが AWS Organizations 管理アカウントまたは [代理管理者](#) アカウントでない場合は、アカウントを信頼ゾーンとして持つアナライザーは 1 つだけ作成できます。

9. オプション。アナライザーに適用するタグを追加します。
10. 送信 を選択します。

外部アクセスアナライザーを作成して IAM Access Analyzer を有効にする  
と、`AWSServiceRoleForAccessAnalyzer` という名前のサービスリンクロールがアカウント内に作成されます。

組織を信頼ゾーンとして持つ外部アクセスアナライザーを作成する方法

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。
3. [アナライザー設定] を選択します。
4. [Create analyzer] (アナライザーの作成) を選択します。
5. [分析] セクションで [外部アクセス分析] を選択します。
6. [アナライザーの詳細] セクションで、表示されているリージョンが、IAM Access Analyzer を有効にするリージョンであることを確認します。
7. アナライザーの名前を入力します。
8. アナライザーの信頼ゾーンとして [現在の組織] を選択します。
9. オプション。アナライザーに適用するタグを追加します。
10. 送信 を選択します。

組織を信頼ゾーンとして持つ外部アクセスアナライザーを作成する

と、`AWSServiceRoleForAccessAnalyzer` という名前のサービスリンクロールが組織の各アカウントに作成されます。


現在のアカウント用の未使用のアクセスアナライザーを作成する方法

1 つの AWS アカウント用の未使用のアクセスアナライザーを作成するには、次の手順を使用します。未使用のアクセスの場合、アナライザーの検出結果はリージョンによって変わりません。リソースがある各リージョンにアナライザーを作成する必要はありません。

IAM Access Analyzer では、1 か月あたりにアナライザーごとに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。
3. [アナライザー設定] を選択します。
4. [Create analyzer] (アナライザーの作成) を選択します。

5. [分析] セクションで [未使用のアクセス分析] を選択します。
6. アナライザーの名前を入力します。
7. [追跡期間] に、使用されていないアクセス許可に関する検出結果を生成する対象の日数を入力します。例えば、90 日と入力した場合、アナライザーは、選択したアカウント内の IAM エンティティについて、アナライザーが最後にスキャンしてから 90 日以上使用されていないすべてのアクセス許可に関する検出結果を生成します。1~180 日の値を選択できます。
8. [選択したアカウント] で [現在の AWS アカウント] を選択します。

 Note

アカウントが AWS Organizations 管理アカウントでも [代理管理者](#) アカウントでもない場合は、自分のアカウントを選択したアカウントとして持つアナライザーは 1 つだけ作成できます。

9. オプション。アナライザーに適用するタグを追加します。
10. 送信 を選択します。


未使用のアクセスアナライザーを作成して IAM Access Analyzer を有効にする

と、AWSServiceRoleForAccessAnalyzer という名前のサービスリンクロールがアカウント内に作成されます。

現在の組織で未使用のアクセスアナライザーを作成する方法

組織用の未使用のアクセスアナライザーを作成し、組織内のすべての AWS アカウントを一元的に確認できるようにするには、次の手順を使用します。未使用のアクセス分析の場合、アナライザーの検出結果はリージョンによって変わりません。リソースがある各リージョンにアナライザーを作成する必要はありません。

IAM Access Analyzer では、1 か月あたりにアナライザーごとに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

 Note

メンバーアカウントが組織から削除されると、未使用のアクセスアナライザーは、24 時間後にそのアカウントの新しい検出結果の生成と既存の結果の更新を停止します。組織から削除されたメンバーアカウントに関連する検出結果は、90 日後に完全に削除されます。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。
3. [アナライザー設定] を選択します。
4. [Create analyzer] (アナライザーの作成) を選択します。
5. [分析] セクションで [未使用のアクセス分析] を選択します。
6. アナライザーの名前を入力します。
7. [追跡期間] に、使用されていないアクセス許可に関する検出結果を生成する対象の日数を入力します。例えば、90 日と入力した場合、アナライザーは、選択した組織のアカウント内の IAM エンティティについて、アナライザーが最後にスキャンしてから 90 日以上使用されていないすべてのアクセス許可に関する検出結果を生成します。1~180 日の値を選択できます。
8. [選択したアカウント] で、アナライザーに対して選択したアカウントとして [現在の組織] を選択します。
9. オプション。アナライザーに適用するタグを追加します。
10. 送信 を選択します。

未使用のアクセスアナライザーを作成して IAM Access Analyzer を有効にする

と、`AWSServiceRoleForAccessAnalyzer` という名前のサービスリンクロールがアカウント内に作成されます。

## IAM Access Analyzer のステータス

アナライザーのステータスを表示するには、[アナライザー] を選択します。組織またはアカウント用に作成されたアナライザーは、次のステータスを持つことができます。

ステータス	説明
[アクティブ]	<p>外部アクセスアナライザーの場合、アナライザーは、信頼ゾーン内のリソースをアクティブにモニタリングします。アナライザーは、新しい結果をアクティブに生成し、既存の結果を更新します。</p> <p>未使用のアクセスアナライザーの場合、アナライザーは、指定された追跡期間における選択した組織または AWS アカウント内で使用されていないアクセス権限をアクティブにモニタリン</p>

ステータス	説明
	グします。アナライザーは、新しい結果をアクティブに生成し、既存の結果を更新します。
[作成中]	アナライザーの作成はまだ進行中です。作成が完了すると、アナライザーがアクティブになります。
無効	AWS Organizations 管理者が実行したアクションにより、アナライザーは無効になっています。たとえば、IAM Access Analyzer の代理管理者としてアナライザーのアカウントが削除されています。アナライザーが無効化された状態の場合、新しい検出結果の生成も、既存の結果の更新も行われません。
[失敗]	設定の問題により、アナライザーの作成に失敗しました。アナライザーにより結果が生成されません。アナライザーを削除し、新しいアナライザーを作成します。

## IAM Access Analyzer の検出結果ダッシュボードの表示

AWS Identity and Access Management Access Analyzer は、外部アクセスと未使用のアクセスの検出結果を視覚的な概要ダッシュボードにまとめます。このダッシュボードは、アクセス許可の効果的な使用を大規模に把握し、注意が必要なアカウントを特定するのに役立ちます。ダッシュボードを使用して、AWS 組織、アカウント、および検出結果タイプごとに結果を確認できます。

外部アクセスアナライザーの概要ダッシュボードを表示する方法

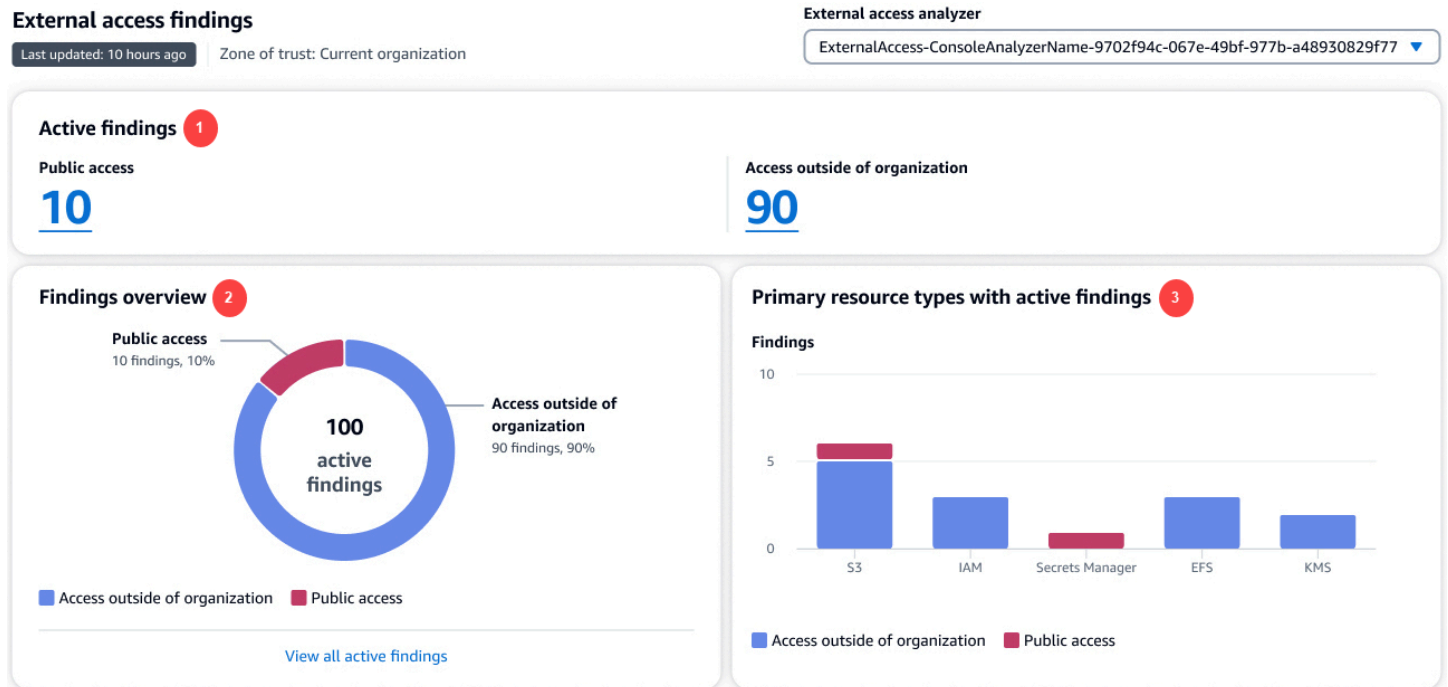
### Note

アナライザーを作成または更新した後、概要ダッシュボードに検出結果の更新が反映されるまでに時間がかかることがあります。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。



2. [Access analyzer] (アクセスアナライザー) を選択します。[概要] ウィンドウが表示されます。
3. [外部アクセスアナライザー] ドロップダウンからアナライザーを選択します。アナライザーの検出結果の概要が [外部アクセスの結果] セクションに表示されます。



上のイメージで、外部アクセスの検出結果ダッシュボードは [概要] ページ内に表示されます。

1. [アクティブな結果] セクションには、パブリックアクセスのアクティブな検出結果の数と、アカウントまたは組織の外部へのアクセスを提供するアクティブな結果の数が表示されます。数字を選択すると、各タイプのすべてのアクティブな検出結果がリストされます。
2. [検出結果の概要] セクションには、アクティブな検出結果のタイプの内訳が表示されます。[すべてのアクティブな結果を表示] を選択すると、アナライザーのアカウントまたは組織のアクティブな検出結果の完全なリストが表示されます。
3. [アクティブな結果があるプライマリリソースタイプ] セクションには、アクティブな検出結果があるプライマリリソースタイプの内訳が表示されます。この情報は、プライマリリソースの検出結果を最初に優先するのに役立ちます。例えば、Amazon S3、DynamoDB、AWS KMS などです。これは、すべてのリソースタイプを網羅したリストではありません。アナライザーには、このセクションにリストされていないリソースタイプに関するアクティブな検出結果がある場合があります。



## 未使用のアクセスアナライザーの概要ダッシュボードを表示する方法

IAM Access Analyzer では、1 か月あたりに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

### Note

アナライザーを作成または更新した後、ユーザーとロールの数に応じて、概要ダッシュボードに検出結果の更新が反映されるまでに時間がかかることがあります。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。[概要] ウィンドウが表示されます。
3. [未使用のアクセスアナライザー] ドロップダウンからアナライザーを選択します。アナライザーの検出結果の概要が [未使用のアクセスの結果] セクションに表示されます。

## Unused access findings

Unused access analyzer

Last updated: 10 hours ago

Tracking period: 90 days

Current organization

UsedAccess-ConsoleAnalyzerName-9702f94c-067e-49bf-977b-a48930829f77

Active findings 1

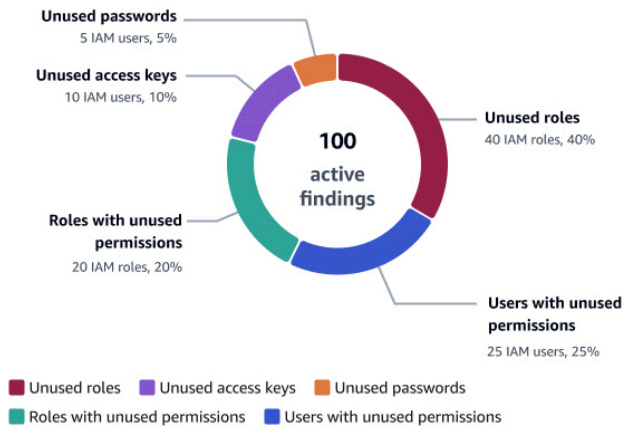
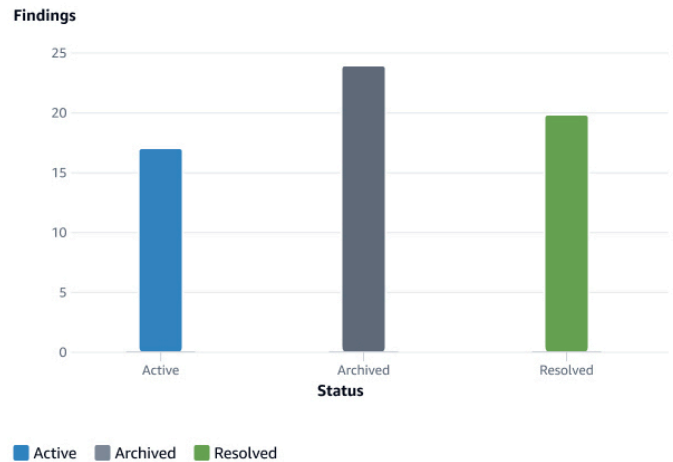
Unused roles

**40**

Unused credentials

**15**

Unused permissions

**45**Findings overview 2[View all active findings](#)Finding status 3Accounts with the most findings for unused access 4

Account	Active findings	Findings by type
<a href="#">Audit</a> 11111111111111	15	Unused roles, Unused access keys, Unused passwords, Roles with unused permissions, Users with unused permissions
<a href="#">Log</a> 22222222222222	10	Unused roles, Unused access keys, Unused passwords, Roles with unused permissions, Users with unused permissions
<a href="#">Security</a> 33333333333333	10	Unused roles, Unused access keys, Unused passwords, Roles with unused permissions, Users with unused permissions
<a href="#">Production</a> 44444444444444	10	Unused roles, Unused access keys, Unused passwords, Roles with unused permissions, Users with unused permissions
<a href="#">Sandbox</a> 55555555555555	5	Unused access keys, Roles with unused permissions, Users with unused permissions

上のイメージで、外部アクセスの検出結果ダッシュボードは [概要] ページ内に表示されます。

- [アクティブな結果] セクションには、アカウントまたは組織内の未使用のロール、未使用の認証情報、および未使用のアクセス許可に関するアクティブな検出結果の数が表示されます。[未使用の認証情報] には、未使用のアクセスキーと未使用のパスワードの検出結果の両方が含まれます。[未使用のアクセス許可] には、未使用のアクセス許可を持つユーザーとロールの両方が含まれます。数字を選択すると、各タイプのすべてのアクティブな検出結果がリストされます。

2. [検出結果の概要] セクションには、アクティブな検出結果のタイプの内訳が表示されます。[すべてのアクティブな結果を表示] を選択すると、アナライザーのアカウントまたは組織のアクティブな検出結果の完全なリストが表示されます。
3. [検索ステータス] セクションには、アカウントまたは組織の検出結果のステータス (アクティブ、アーカイブ済み、解決済み) の内訳が表示されます。
4. [未使用のアクセスの結果が最も多いアカウント] セクションは、未使用のアクセスアナライザーで選択したアカウントが組織レベルの場合にのみ表示されます。これには、組織内でアクティブな検出結果が最も多いアカウントの内訳が含まれます。これは、組織内のすべてのアカウントを網羅したリストではありません。アナライザーには、このセクションにリストされていない他のアカウントに関するアクティブな検出結果がある場合があります。

## 結果を使用する

### 外部アクセスの検出結果

外部アクセスの検出結果は、信頼ゾーンの外部で共有されているリソースのインスタンスごとに 1 回だけ生成されます。リソースベースのポリシーが変更されるたびに、IAM Access Analyzer はポリシーを分析します。更新されたポリシーが共有しているリソースが、結果内で識別済みであっても、アクセス許可や条件が異なっている場合は、そのリソース共有のインスタンスに対して新しい結果が生成されます。最初の結果でアクセスが削除されると、その結果のステータスは [解決済み] に更新されます。

すべての検出結果のステータスは、結果をアーカイブするか、結果を生成したアクセスを削除するまでは、[アクティブ] のままです。アクセスを削除すると、結果のステータスは [解決済み] に更新されます。

#### Note

ポリシーが変更されてから IAM Access Analyzer がリソースを分析して外部アクセスの結果を更新するまで、最大で 30 分かかる場合があります。

### 未使用のアクセスに関する検出結果

アナライザーの作成時に指定された日数に基づいて、選択したアカウントまたは組織内の IAM エンティティについて、未使用のアクセスに関する検出結果が生成されます。以下のいずれかの条件が満たされている場合、次にアナライザーがエンティティをスキャンしたときに、新しい結果が生成されます。

- 指定された日数の間、ロールが非アクティブである。
- 未使用のアクセス許可、未使用のユーザーパスワード、または未使用のユーザーアクセスキーが、指定された日数を超えている。

アカウント内のすべての検出結果を確認して、外部アクセスまたは未使用のアクセスが想定および承認されているかどうかを判断する必要があります。結果で識別された外部アクセスまたは未使用のアクセスが想定されたものである場合は、その結果をアーカイブできます。結果をアーカイブすると、そのステータスは [アーカイブ済み] に変わり、検出結果はアクティブな結果のリストから削除されます。結果は削除されません。アーカイブした結果はいつでも表示できます。アクティブな結果がゼロになるまで、アカウント内のすべての結果を処理してください。検出結果がゼロになると、新しく生成された [アクティブ] な結果は、環境内の最近の変更によるものであることがわかります。

#### Note

未使用のアクセスの検出結果は、[ListFindingsV2](#) API アクションを使用してのみ利用可能です。

## 調査結果を確認する

[IAM Access Analyzer を有効化](#)したら、次のステップとして結果を確認し、結果で識別されたアクセスが意図的なものであるか、意図的なものでないかを判断します。また、検出結果を確認して、意図的なアクセスに対して類似する結果を特定し、その結果を自動的にアーカイブするための[アーカイブルールを作成](#)することもできます。アーカイブ済みの結果と解決済みの結果を確認することもできます。

結果を確認するには

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [Access analyzer] (アクセスアナライザー) を選択します。
3. 検出結果のダッシュボードが表示されます。外部のアクセスアナライザーまたは未使用のアクセスアナライザーのアクティブな検出結果を選択します。

検出結果のダッシュボードの表示について詳しくは、「[IAM Access Analyzer の検出結果ダッシュボードの表示](#)」を参照してください。

**Note**

結果が表示されるのは、アナライザーの結果を表示するアクセス許可がある場合だけです。

アナライザーに関するすべての検出結果が表示されます。アナライザーによって生成されたその他の検出結果を表示するには、[ステータス] ドロップダウンから該当する結果タイプを選択します。

- [Active (アクティブ)] を選択すると、アナライザーによって生成されたすべてのアクティブな結果が表示されます。
- [Archived (アーカイブ済み)] を選択すると、アナライザーによって生成された結果のうち、アーカイブ済みの結果のみが表示されます。詳細については、「[結果のアーカイブ](#)」を参照してください。
- [Resolved (解決済み)] を選択すると、アナライザーによって生成された結果のうち、解決済みの結果のみが表示されます。結果を生成した問題を修正すると、結果のステータスが [解決済み] に変わります。

**Important**

解決済みの結果は、結果の最終更新から 90 日後に削除されます。アクティブな結果とアーカイブ済みの結果は、これらを生成したアナライザーを削除しない限り、削除されません。

- [All (すべて)] を選択すると、アナライザーによって生成されたすべての結果が、ステータスを問わず、表示されます。

## 外部アクセスの検出結果

[外部アクセス] を選択し、[アナライザーを表示] ドロップダウンから外部のアクセスアナライザーを選択します。外部アクセスアナライザーの [結果] ページには、その検出結果を生成した共有リソースおよびポリシーステートメントに関する以下の詳細が表示されます。

### 検出結果 ID

結果に割り当てられた一意の ID。結果 ID を選択すると、その結果を生成したリソースとポリシーステートメントに関する追加の詳細が表示されます。

## [リソース]

信頼ゾーン内にない外部エンティティに対してアクセスを許可するポリシーが適用されているリソースのタイプと部分的な名。

### リソース所有者のアカウント

この列は、信頼ゾーンとして組織を使用している場合にのみ表示されます。結果で報告されたリソースを所有する組織内のアカウント。

### 外部プリンシパル

分析されたポリシーからアクセス許可を付与されている、信頼ゾーン外のプリンシパル。有効な値を次に示します。

- AWS アカウント - リストされた AWS アカウント内のプリンシパルのうち、そのアカウントの管理者からアクセス許可を付与されているすべてのプリンシパルがリソースにアクセスできます。
- 任意のプリンシパル - 任意の AWS アカウント内のプリンシパルのうち、[条件] 列に示された条件を満たしているすべてのプリンシパルがリソースへのアクセス許可を持ちます。たとえば、VPC がリストされている場合は、このリストされた VPC へのアクセス許可を持つ任意のアカウント内のすべてのプリンシパルがリソースにアクセスできることを意味します。
- 正規ユーザー - AWS アカウント内のプリンシパルのうち、リストされた正規ユーザー ID を持つすべてのプリンシパルがリソースへのアクセス許可を持ちます。
- IAM ロール - リストされた IAM ロールは、リソースへのアクセス許可を持ちます。
- IAM ユーザー - リストされた IAM ユーザーは、リソースへのアクセス許可を持ちます。

### 条件

アクセスを許可するポリシーステートメントの条件。たとえば、[Condition (条件)] フィールドに [Source VPC (ソース VPC)] が含まれている場合は、リソースが、リストされた VPC にアクセスできるプリンシパルと共有されていることを意味します。条件は、グローバルまたはサービス固有です。[グローバル条件キー](#)には aws: プレフィックスが付きます。

### 共有方法

[共有方法] フィールドは、結果を生成したアクセスの許可方法を示します。有効な値を次に示します。

- バケットポリシー - Amazon S3 バケットにアタッチされているバケットポリシーです。
- アクセスコントロールリスト - Amazon S3 バケットにアタッチされているアクセスコントロールリスト (ACL)。

- アクセスポイント — Amazon S3 バケットに関連付けられたアクセスポイントまたはマルチリージョンのアクセスポイント。アクセスポイントの ARN が [結果] の詳細に表示されます。

## アクセスレベル

リソーススペースのポリシーのアクションによって外部エンティティに許可されるアクセスのレベル。詳細については、結果の詳細を参照してください。アクセスレベルには、以下の値が含まれます。

- リスト - オブジェクトが存在するかどうかを判断するためにサービス内のリソースをリストするためのアクセス許可。このレベルのアクセス権を持つアクションはオブジェクトをリストできますが、リソースのコンテンツは表示されません。
- 読み取り - サービス内のリソースのコンテンツと属性を読み取るためのアクセス許可。ただし、これらを編集することはできません。
- 書き込み - サービス内のリソースを作成、削除、または変更するためのアクセス許可。
- アクセス許可 - サービス内のリソースに対するアクセス許可を付与または変更するためのアクセス許可。
- タグ付け - リソースタグの状態のみを変更するアクションを実行するためのアクセス許可。

## 更新

結果のステータスに対する最終更新のタイムスタンプ、または更新が行われていない場合は結果が生成された日時。

### Note

ポリシーが変更されてから IAM Access Analyzer がリソースを再度分析して結果を更新するまでに、最大で 30 分かかる場合があります。

## [ステータス]

結果のステータス。[Active (アクティブ)]、[Archived (アーカイブ済み)]、または [Resolved (解決済み)] のいずれかです。

## 未使用のアクセスに関する検出結果

IAM Access Analyzer では、1 か月あたりに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。



[未使用のアクセス] を選択し、[アナライザーを表示] ドロップダウンから未使用のアクセスアナライザーを選択します。未使用のアクセスアナライザーの [検出結果] ページには、その結果を生成した IAM エンティティに関する以下の詳細が表示されます。

### 検出結果 ID

結果に割り当てられた一意の ID。結果 ID を選択すると、その結果を生成した IAM エンティティに関する追加の詳細が表示されます。

### 結果タイプ

未使用のアクセスに関する検出結果のタイプは、[未使用のアクセスキー]、[未使用のパスワード]、[未使用の権限]、または [未使用のロール] のいずれかです。

### IAM エンティティ

検出結果で報告された IAM エンティティ。これは IAM ロールまたはユーザーのいずれかです。

### AWS アカウント ID

この列は、組織内のすべての AWS アカウントに対してアナライザーを設定した場合にのみ表示されます。組織内の AWS アカウントのうち、結果で報告された IAM エンティティを所有するアカウント。

### 最終更新日

検出結果で報告された IAM エンティティが最後に更新された日時、または更新が行われていない場合はエンティティが作成された日時。

### [ステータス]

結果のステータス (アクティブ、アーカイブ済み、解決済みのいずれか)。

## 検出結果のフィルタリング

検出結果ページのデフォルトのフィルタリングでは、すべての検出結果が表示されます。アクティブな検出結果を表示するには、[ステータス] ドロップダウンから [アクティブ] ステータスを選択します。アーカイブされた検出結果を表示するには、[ステータス] ドロップダウンから [アーカイブ済み] ステータスを選択します。IAM Access Analyzer を初めて使用した時点では、アーカイブ済み結果はありません。

フィルターを使用して、指定したプロパティ基準を満たす検出結果のみを表示します。フィルターを作成するには、フィルターを適用するプロパティを選択し、プロパティが値に等しいか値を含むかを選択し、フィルターを適用するプロパティ値を入力または選択します。例えば、特定の AWS アカウ



ントの検出結果のみを表示するフィルターを作成するには、プロパティに [AWS アカウント] を選択し、[AWS アカウント =] を選択し、検出結果を表示する AWS アカウントのアカウント番号を入力します。

アーカイブ規則の作成または更新に使用できるフィルターキーのリストについては、「[IAM Access Analyzer フィルターキーにアクセスする](#)」を参照してください。

## 外部アクセスの検出結果のフィルタリング

### 外部アクセスの検出結果をフィルタリングする方法

1. [外部アクセス] を選択し、[アナライザーを表示] ドロップダウンからアナライザーを選択します。
2. 検索ボックスを選択して、使用可能なプロパティのリストを表示します。
3. 表示された結果をフィルターするために使用するプロパティを選択します。
4. プロパティと一致させる値を選択します。すべての結果のうち、この値を持つ結果のみが表示されます。

例えば、プロパティとして [リソース] を選択し、[リソース:] を選択し、バケットの名前の一部または全部を入力し、Enter キーを押します。フィルター基準に一致するバケットの検出結果のみが表示されます。パブリックアクセスを許可するリソースの検出結果のみを表示するフィルターを作成するには、[パブリックアクセス] プロパティを選択し、[パブリックアクセス =] を選択し、[パブリックアクセス = true] を選択します。

他のプロパティを追加して、表示された結果をさらに絞り込むことができます。他のプロパティを追加すると、フィルターのすべての条件に一致する結果のみが表示されます。1 つのプロパティまたは別のプロパティの OR 条件を満たす結果を表示するフィルターの定義は、サポートされていません。[フィルターをクリア] を選択すると、定義済みのすべてのフィルターがクリアされ、アナライザーで指定されたステータスを持つすべての検出結果が表示されます。

一部のフィールドは、組織を信頼ゾーンとして持つアナライザーの結果を表示している場合にのみ表示されます。

フィルターを定義するには、以下のプロパティを使用できます。

- Public access (パブリックアクセス) - パブリックアクセスを許可するリソースの結果でフィルタリングするには、[Public access (パブリックアクセス)] でフィルタリングした後、[Public access: true (パブリックアクセス: true)] を選択します。

- リソース - リソースでフィルターするには、リソース名の全部または一部を入力します。
- リソースタイプ - リソースタイプでフィルターするには、表示されたリストからタイプを選択します。
- リソース所有者アカウント - 結果で報告されたリソースを所有する組織内のアカウントでフィルタリングするには、このプロパティを使用します。
- AWS アカウント - ポリシーステートメントの [プリンシパル] セクションでアクセスを許可された AWS アカウントでフィルタリングするには、このプロパティを使用します。AWS アカウントでフィルタリングするには、12 桁の AWS アカウント ID の全部または一部、あるいは現在のアカウントのリソースにアクセスできる外部 AWS ユーザーまたはロールのフルアカウント ARN の全部または一部を入力します。
- 正規ユーザー - 正規ユーザーでフィルタリングするには、Amazon S3 バケットに定義されている正規ユーザー ID を入力します。詳細については、「[AWS Account 識別子](#)」を参照してください。
- フェデレーテッドユーザー - フェデレーテッドユーザーでフィルターするには、フェデレーテッド ID の ARN の全部または一部を入力します。詳細については、「[ID プロバイダーとフェデレーション](#)」を参照してください。
- 結果 ID - 結果 ID でフィルタリングするには、結果 ID の全部または一部を入力します。
- プリンシパル ARN - このプロパティを使用して、aws:PrincipalArn 条件キーで使用されるプリンシパル (IAM ユーザー、ロール、またはグループ) の ARN をフィルターします。プリンシパル ARN でフィルタリングするには、結果で報告された外部 AWS アカウントの IAM ユーザー、ロール、グループの ARN の全部または一部を入力します。
- プリンシパルの組織 ID - プリンシパルの組織 ID でフィルターするには、結果の条件として指定された AWS 組織に属する外部プリンシパルに関連付けられている組織 ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- プリンシパル OrgPaths - プリンシパル OrgPaths でフィルターするには、ポリシーの条件として、指定した組織または組織単位 (OU) のアカウントメンバーであるすべての外部プリンシパルにアクセスを許可する AWS 組織または OU の ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ソースアカウント - ソースアカウントでフィルタリングするには、AWS のクロスサービスのアクセス許可で使用されるリソースに関連付けられた AWS アカウント ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ソース ARN - ソース ARN でフィルターするには、結果の条件として指定された ARN の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

- ソース IP - ソース IP でフィルターするには、指定した IP アドレスを使用するときに、現在のアカウント内のリソースへのアクセスを外部エンティティに許可する IP アドレスの全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ソース VPC - ソース VPC でフィルターするには、指定した VPC を使用するときに、現在のアカウント内のリソースへのアクセスを外部エンティティに許可する VPC ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ソース OrgID - ソース OrgID でフィルタリングするには、AWS のクロスサービスのアクセス許可で使用されるリソースに関連付けられた組織 ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ソース OrgPaths - ソース OrgPaths でフィルタリングするには、AWS のクロスサービスのアクセス許可で使用されるリソースに関連付けられた組織単位 (OU) の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- ユーザー ID - ユーザー ID でフィルタリングするには、現在のアカウント内のリソースへのアクセスを許可されている外部 AWS アカウントの IAM ユーザーのユーザー ID の全部または一部を入力します。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。
- KMS キー ID - KMS キー ID でフィルタリングするには、現在のアカウントで AWS KMS 暗号化された Amazon S3 オブジェクトへのアクセスの条件として指定されている KMS キーのキー ID の全部または一部を入力します。
- Google オーディエンス - Google オーディエンスでフィルターするには、現在のアカウントで IAM ロールへのアクセスの条件として指定されている Google アプリケーション ID の全部または一部を入力します。詳細については、「[IAM および AWS STS の条件コンテキストキー](#)」を参照してください。
- Cognito オーディエンス - Amazon Cognito オーディエンスでフィルタリングするには、現在のアカウントで IAM ロールへのアクセスの条件として指定されている Amazon Cognito ID プールの ID の全部または一部を入力します。詳細については、「[IAM および AWS STS の条件コンテキストキー](#)」を参照してください。
- 呼び出し元のアカウント - IAM ロール、ユーザー、アカウントルートユーザーなど、呼び出し元のエンティティを所有または含有するアカウントの AWS アカウント ID。これは、AWS KMS を呼び出すサービスによって使用されます。呼び出し元のアカウントでフィルタリングするには、AWS アカウント ID の全部または一部を入力します。
- Facebook アプリ ID - Facebook アプリ ID でフィルターするには、現在のアカウントで IAM ロールへの Login with Facebook フェデレーションアクセスを許可する条件として指定されている Facebook アプリケーション ID (またはサイト ID) の全部または一部を入力します。詳細については、「[IAM および AWS STS 条件コンテキストキー](#)」の ID のセクションを参照してください。

- Amazon アプリ ID - Amazon アプリ ID でフィルターするには、現在のアカウントで IAM ロールへの Login with Amazon フェデレーションアクセスを許可する条件として指定されている Amazon アプリケーション ID (またはサイト ID) の全部または一部を入力します。詳細については、「[IAM および AWS STS 条件コンテキストキー](#)」の ID のセクションを参照してください。
- Lambda イベントソーストークン - Alexa 統合で渡された Lambda イベントソーストークンでフィルターするには、トークン文字列の全部または一部を入力します。

## 未使用のアクセスの検出結果のフィルタリング

### 未使用のアクセスの検出結果をフィルタリングする方法

1. [未使用のアクセス] を選択し、[アナライザーを表示] ドロップダウンからアナライザーを選択します。
2. 検索ボックスを選択して、使用可能なプロパティのリストを表示します。
3. 表示された結果をフィルターするために使用するプロパティを選択します。
4. プロパティと一致させる値を選択します。すべての結果のうち、この値を持つ結果のみが表示されます。

例えば、プロパティとして [検出結果タイプ] を選択し、[検出結果タイプ =] を選択し、[検出結果タイプ = UnusedIAMRole] を選択すると、タイプが [UnusedIAMRole] の検出結果のみが表示されます。

他のプロパティを追加して、表示された結果をさらに絞り込むことができます。他のプロパティを追加すると、フィルターのすべての条件に一致する結果のみが表示されます。1 つのプロパティまたは別のプロパティの OR 条件を満たす結果を表示するフィルターの定義は、サポートされていません。[フィルターをクリア] を選択すると、定義済みのすべてのフィルターがクリアされ、アナライザーで指定されたステータスを持つすべての検出結果が表示されます。

以下のフィールドは、未使用のアクセスをモニタリングするアナライザーの検出結果を表示する場合にのみ表示されます。

- 検出結果タイプ - 検出結果タイプでフィルターするには、[検出結果タイプ] でフィルターし、検出結果のタイプを選択します。
- リソース - リソースでフィルターするには、リソース名の全部または一部を入力します。
- リソースタイプ - リソースタイプでフィルターするには、表示されたリストからタイプを選択します。

- リソース所有者アカウント – 結果で報告されたリソースを所有する組織内のアカウントでフィルタリングするには、このプロパティを使用します。
- 検出結果 ID – 検出結果 ID でフィルターするには、検出結果 ID の全部または一部を入力します。

## 結果のアーカイブ

リソースへの意図的なアクセスに関する検出結果を取得したら、その結果をアーカイブできます。例えば、承認されたワークフローに対して複数のユーザーによって使用される IAM ロールに関する外部アクセスの結果や、まだ必要になる可能性のあるアクセスキーに関する未使用のアクセスの結果などです。検出結果をアーカイブすると、その結果はアクティブな検出結果のリストから消去されません。アーカイブされた検出結果は削除されません。[結果] ページをフィルタリングしてアーカイブ済みの検出結果を表示し、いつでもそれらのアーカイブを解除できます。

[Findings (結果)] ページの結果をアーカイブするには

1. アーカイブする 1 つ以上の結果の横にあるチェックボックスをオンにします。
2. [アクション] を選択し、[アーカイブ] を選択します。

画面上部に確認メッセージが表示されます。

[結果の詳細] ページから検出結果をアーカイブする方法

1. アーカイブする結果の [Finding ID (結果 ID)] を選択します。
2. [Archive] (アーカイブ) を選択します。

画面上部に確認メッセージが表示されます。

検索結果のアーカイブを解除するには、上記の手順を繰り返します。ただし、[Unarchive] (アーカイブ) ではなく、[Unarchive] (アーカイブを解除) を選択します。結果のアーカイブを解除すると、ステータスがアクティブに設定されます。

## 結果の解決

### 外部アクセスの検出結果

許可していないアクセスから生成された外部アクセスの検出結果を解決するには、ポリシーステートメントを変更して、該当するリソースへのアクセスを許可するアクセス許可を削除します。例え

ば、Amazon S3 バケットに関する検出結果の場合、Amazon S3 コンソールを使用してバケットに対するアクセス許可を設定します。IAM ロールの場合、IAM コンソールを使用して、リストされた IAM ロールの [信頼ポリシーを変更](#) します。その他のサポートされているリソースの場合は、コンソールを使用して、結果を生成したポリシーステートメントを変更します。

IAM ロールに適用されているポリシーを変更するなど、外部アクセスの結果を解決するための変更を行うと、IAM Access Analyzer はリソースを再度スキャンします。リソースが信頼ゾーン外で共有されなくなると、結果のステータスは [Resolved (解決済み)] に変更されます。検出結果はアクティブな結果のリストに表示されなくなり、代わりに解決済みの結果のリストに表示されます。

#### Note

これは、[エラー] の検出結果には適用されません。IAM Access Analyzer がリソースを解析できないときは、エラー検出が生成されます。IAM Access Analyzer がリソースを解析できない問題を解決すると、解決済みの検出結果に変更されるのではなく、エラーの検出結果が完全に削除されます。

変更に伴って、プリンシパルやアクセス許可が異なるなど、異なる方法でリソースが信頼ゾーン外で共有されるようになった場合、IAM Access Analyzer は新しいアクティブな結果を生成します。

#### Note

ポリシーが変更されてから IAM Access Analyzer がリソースを再度分析して結果を更新するまでに、最大で 30 分かかる場合があります。解決済みの結果は、検出結果のステータスの最終更新から 90 日後に削除されます。

## 未使用のアクセスに関する検出結果

未使用のアクセスアナライザーに関する検出結果に対し、IAM Access Analyzer は検出結果のタイプに基づいて、その検出結果を解決するための推奨手順を提供します。

未使用のアクセスの結果を解決するための変更を行うと、次に未使用のアクセスアナライザーを実行したときに、結果のステータスが [解決済み] に変わります。検出結果はアクティブな検出結果のリストに表示されなくなり、代わりに解決済みの検出結果のリストに表示されます。未使用のアクセスの結果の一部のみに対処する変更を行った場合、既存の結果は [解決済み] に変わりますが、新しい結果が生成されます。例えば、結果から未使用のアクセス許可の一部だけを削除し、すべてを削除しない場合などです。



IAM Access Analyzer では、1 か月あたりに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセス分析に対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

## 未使用のアクセス許可に関する検出結果の解決

未使用のアクセス許可に関する検出結果に対し、IAM Access Analyzer は IAM ユーザーまたはロールから削除するポリシーを推奨し、既存のアクセス許可ポリシーを置き換える新しいポリシーを提供します。ポリシーの推奨は、以下のシナリオではサポートされていません。

- 未使用のアクセス許可に関する検出結果は、ユーザーグループに属する IAM ユーザーに対するものです。
- 未使用のアクセス許可に関する検出結果は、IAM Identity Center の IAM ロールに対するものです。
- 未使用のアクセス許可に関する検出結果には、notAction 要素を含む既存のアクセス許可ポリシーがあります。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [未使用のアクセス] を選択します。
3. [検出結果タイプ] が [未使用のアクセス許可] の検出結果を選択します。
4. [推奨事項] セクションで、[推奨ポリシー] 列にポリシーが表示されている場合は、[ポリシーのプレビュー] を選択して、既存のポリシーを置き換える推奨ポリシーとともに既存のポリシーを表示します。推奨ポリシーが複数ある場合は、[次のポリシー] と [前のポリシー] を選択すると、既存のポリシーと推奨ポリシーをそれぞれ表示できます。
5. [JSON をダウンロード] を選択して、すべての推奨ポリシーの JSON ファイルを含む .zip ファイルをダウンロードします。
6. 推奨ポリシーを作成して IAM ユーザーまたはロールにアタッチします。詳細については、「[ユーザーのアクセス許可の変更 \(コンソール\)](#)」および「[ロールのアクセス許可ポリシーの変更 \(コンソール\)](#)」を参照してください。
7. [既存のアクセス許可ポリシー] 列に表示されているポリシーを IAM ユーザーまたはロールから削除します。詳細については、「[ユーザーからのアクセス許可の削除 \(コンソール\)](#)」および「[ロールアクセス許可ポリシーの変更 \(コンソール\)](#)」を参照してください。

## 未使用のロールに関する検出結果の解決

未使用のロールに関する検出結果に対し、IAM Access Analyzer は未使用の IAM ロールを削除することを推奨します。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [未使用のアクセス] を選択します。
3. [検出結果タイプ] が [未使用のロール] の検出結果を選択します。
4. [推奨事項] セクションで、IAM ロールの詳細を確認します。
5. IAM ロールを削除します。詳細については、「[IAM ロールの削除 \(コンソール\)](#)」を参照してください。

## 未使用のアクセスキーに関する検出結果の解決

未使用のアクセスキーに関する検出結果に対し、IAM Access Analyzer は未使用のアクセスキーを非アクティブ化または削除することを推奨します。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [未使用のアクセス] を選択します。
3. [検出結果タイプ] が [未使用のアクセスキー] の検出結果を選択します。
4. [推奨事項] セクションで、アクセスキーの詳細を確認します。
5. アクセスキーを非アクティブ化または削除します。詳細については、「[アクセスキーの管理 \(コンソール\)](#)」を参照してください。

## 未使用のパスワードに関する検出結果の解決

未使用のパスワードに関する検出結果に対し、IAM Access Analyzer は IAM ユーザーの未使用のパスワードを削除することを推奨します。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [未使用のアクセス] を選択します。
3. [検出結果タイプ] が [未使用のパスワード] の検出結果を選択します。
4. [推奨事項] セクションで、IAM ユーザーの詳細を確認します。
5. IAM ユーザーのパスワードを削除します。詳細については、「[IAM ユーザーパスワードの作成、変更、削除 \(コンソール\)](#)」を参照してください。



## 外部アクセスの IAM Access Analyzer リソースタイプ

外部アクセスアナライザーの場合、IAM Access Analyzer は、IAM Access Analyzer を有効にしたリージョンで AWS リソースに適用されているリソースベースのポリシーを分析します。分析されるのは、リソースベースのポリシーのみです。IAM Access Analyzer が各リソースタイプの結果を生成する方法の詳細については、各リソースに関する情報を参照してください。

### Note

リストされているサポート対象のリソースタイプは、外部アクセスアナライザー用です。未使用のアクセスアナライザーは、IAM ユーザーとロールのみをサポートします。詳細については、「[結果を使用する](#)」を参照してください。

外部アクセスでサポートされるリソースタイプ:

- [Amazon Simple Storage Service バケット](#)
- [Amazon Simple Storage Service ディレクトリバケット](#)
- [AWS Identity and Access Management ロール](#)
- [AWS Key Management Service キー](#)
- [AWS Lambda の関数とレイヤー](#)
- [Amazon Simple Queue Service キュー](#)
- [AWS Secrets Manager シークレット](#)
- [Amazon Simple Notification Service トピック](#)
- [Amazon Elastic Block Store ボリュームスナップショット](#)
- [Amazon Relational Database Service DB スナップショット](#)
- [Amazon Relational Database Service DB クラスタースナップショット](#)
- [Amazon Elastic Container Registry リポジトリ](#)
- [Amazon Elastic File System ファイルシステム](#)
- [Amazon DynamoDB Streams](#)
- [Amazon DynamoDB テーブル](#)

## Amazon Simple Storage Service バケット

IAM Access Analyzer が Amazon S3 バケットを分析するときは、バケットに適用された Amazon S3 バケットポリシー、ACL、またはアクセスポイント (マルチリージョンアクセスポイントを含む) が外部エンティティにアクセス権を付与する場合に結果を生成します。外部エンティティは、[フィルタの作成](#)に使用できる、信頼ゾーン外のプリンシパルや他のエンティティです。例えば、バケットポリシーが別のアカウントにアクセスを許可したり、パブリックアクセスを許可したりすると、IAM Access Analyzer は結果を生成します。ただし、バケットで[パブリックアクセスのブロック](#)を有効にすると、アカウントレベルまたはバケットレベルでアクセスをブロックできます。

### Note

IAM Access Analyzer は、クロスアカウントアクセスポイントにアタッチされたアクセスポイントポリシーを分析しません。これは、対象のアクセスポイントとポリシーが、アナライザーのアカウント外にあるためです。IAM Access Analyzer は、バケットからクロスアカウントアクセスポイントへのアクセスが委任され、そのバケットまたはアカウントでパブリックアクセスのブロックが有効になっていない場合に、結果を生成し公開します。パブリックアクセスのブロックを有効にしている場合、この公開結果は解決されます。また、IAM Access Analyzer により、クロスアカウントアクセスポイントでのアカウント間の結果が生成されます。

Amazon S3 でのパブリックアクセスのブロック設定は、バケットに適用されているバケットポリシーより優先されます。また、この設定は、バケットのアクセスポイントに適用されているアクセスポイントポリシーを上書きします。IAM Access Analyzer は、ポリシーが変更されるたびに、パブリックアクセスのブロックの設定を、バケットレベルで分析します。また、6 時間ごとに 1 回のみアカウントレベルで、パブリックアクセスのブロックに関する設定の評価を行います。つまり、最大 6 時間、IAM Access Analyzer はバケットへのパブリックアクセスに関する結果を生成または解決しない場合があります。例えば、パブリックアクセスを許可するバケットポリシーがある場合、IAM Access Analyzer はそのアクセスに関する結果を生成します。その後、パブリックアクセスのブロックを有効化し、バケットへのすべてのパブリックアクセスを (アカウントレベルで) ブロックすると、パブリックアクセスがまったくない状態であっても、IAM Access Analyzer は、そのバケットのポリシーに関する結果を最大 6 時間解決しません。アカウントレベルでパブリックアクセスのブロックを有効化した後は、クロスアカウントアクセスポイントで公開された結果の解決にも、最大 6 時間かかる場合があります。

マルチリージョンアクセスポイントの場合、IAM Access Analyzer は、結果の生成に確立されたポリシーを使用します。IAM Access Analyzer は、マルチリージョンアクセスポイントに対する変更を 6

時間ごとに 1 回評価します。つまり、マルチリージョンアクセスポイントの作成もしくは削除、またはそのポリシーの更新を行った場合でも、IAM Access Analyzer は結果の生成または解決を最大 6 時間実行しないことになります。

## Amazon Simple Storage Service ディレクトリバケット

Amazon S3 ディレクトリバケットは Amazon S3 Express One ストレージクラスを使用します。これは、パフォーマンスが重要なワークロードやアプリケーションに推奨されます。Amazon S3 ディレクトリバケットの場合、IAM Access Analyzer は外部エンティティに対してディレクトリバケットへのアクセスを許可するためのディレクトリバケットポリシー (ポリシー内の条件ステートメントを含む) を分析します。Amazon S3 ディレクトリバケットの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Directory buckets](#)」を参照してください。

## AWS Identity and Access Management ロール

IAM ロールの場合、IAM Access Analyzer は[信頼ポリシー](#)を分析します。ロールの信頼ポリシーでは、ロールを引き受けるために信頼するプリンシパルを定義します。ロールの信頼ポリシーは、IAM のロールに添付される必須のリソースベースのポリシーです。IAM Access Analyzer は、信頼ゾーン外の外部エンティティからアクセスできる信頼ゾーン内のロールに関する結果を生成します。

### Note

IAM ロールはグローバルリソースです。ロールの信頼ポリシーが外部エンティティにアクセスを許可すると、IAM Access Analyzer は有効な各リージョンで結果を生成します。

## AWS Key Management Service キー

AWS KMS keys の場合、IAM Access Analyzer はキーポリシーとキーに適用された許可を分析します。IAM Access Analyzer は、外部エンティティに対してキーへのアクセスを許可すると、結果を生成します。例えば、ポリシーステートメントで [km:CallerAccount](#) 条件キーを使用して、特定の AWS アカウントですべてのユーザーへのアクセスを許可し、現在のアカウント (現在のアナライザーの信頼ゾーン) 以外のアカウントを指定すると、IAM Access Analyzer は結果を生成します。AWS KMS IAM ポリシーステートメントにおける条件キーの詳細については、「[AWS KMS 条件キー](#)」を参照してください。

IAM Access Analyzer は、KMS キーを分析することで、キーポリシーや許可のリストなど、キーのメタデータを読み取ります。キーポリシーが IAM Access Analyzer ロールに対してキーのメタデータの読み取りを許可しない場合、結果としてアクセス拒否エラーが生成されます。例えば、以下のポリ

シーステートメント例がキーに適用されている唯一のポリシーである場合は、IAM Access Analyzer でアクセス拒否エラーの結果が生成されることとなります。

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Admin"
  },
  "Action": "kms:*",
  "Resource": "*"
}
```

このステートメントでは、AWS アカウント 111122223333 の「Admin」という名前のロールのみがキーへのアクセスを許可されるため、IAM Access Analyzer はキーを完全には分析できず、結果としてアクセス拒否エラーが生成されます。エラーの結果は、[Findings (結果)] テーブルに赤いテキストで表示されます。結果は以下のようになります。

```
{
  "error": "ACCESS_DENIED",
  "id": "12345678-1234-abcd-dcba-111122223333",
  "analyzedAt": "2019-09-16T14:24:33.352Z",
  "resource": "arn:aws:kms:us-west-2:1234567890:key/1a2b3c4d-5e6f-7a8b-9c0d-1a2b3c4d5e6f7g8a",
  "resourceType": "AWS::KMS::Key",
  "status": "ACTIVE",
  "updatedAt": "2019-09-16T14:24:33.352Z"
}
```

KMS キーを作成する場合、キーにアクセスするために付与されるアクセス許可は、キーの作成方法によって異なります。キーリソースへのアクセス拒否エラーが結果として返された場合は、次のポリシーステートメントをキーリソースに適用して、キーへのアクセス許可を IAM Access Analyzer に付与します。

```
{
  "Sid": "Allow IAM Access Analyzer access to key metadata",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/aws-service-role/access-analyzer.amazonaws.com/AWSServiceRoleForAccessAnalyzer"
  },
}
```

```
"Action": [  
    "kms:DescribeKey",  
    "kms:GetKeyPolicy",  
    "kms:List*",  
],  
"Resource": "*" ]},
```

KMS キーリソースへのアクセス拒否が結果として返された場合、キーポリシーを更新して結果を解決すると、結果は解決済みの状態に更新されます。外部エンティティに対してキーへのアクセスを許可するポリシーステートメントやキー付与がある場合、キーリソースに関する追加の結果が表示されることがあります。

## AWS Lambda の関数とレイヤー

AWS Lambda 関数の場合、IAM Access Analyzer は、外部エンティティに対して関数へのアクセスを許可するポリシー (ポリシー内の条件ステートメントを含む) を分析します。Lambda では、独自のリソースベースのポリシーを関数、バージョン、エイリアス、レイヤーにアタッチできます。IAM Access Analyzer は、関数とレイヤーにアタッチされたリソースベースのポリシーに基づいて外部アクセスを報告します。IAM Access Analyzer は、エイリアスにアタッチされたリソースベースのポリシーや、修飾された ARN を使用して呼び出された特定のバージョンに基づいて外部アクセスを報告することはありません。

詳細については、「AWS Lambda デベロッパーガイド」の「[Using resource-based policies for Lambda](#)」と「[Using versions](#)」を参照してください。

## Amazon Simple Queue Service キュー

Amazon SQS キューの場合、IAM Access Analyzer は外部エンティティに対してキューへのアクセスを許可するポリシー (ポリシー内の条件ステートメントを含む) を分析します。

## AWS Secrets Manager シークレット

AWS Secrets Manager シークレットの場合、IAM Access Analyzer は外部エンティティにシークレットへのアクセスを許可するポリシー (ポリシー内の条件ステートメントを含む) を分析します。

## Amazon Simple Notification Service トピック

IAM Access Analyzer は、Amazon SNS トピックにアタッチされたリソースベースのポリシーを分析します。これには、外部からのトピックへのアクセスを許可するポリシー内の条件ステートメン

トが含まれます。リソースベースのポリシーを使用して、トピックのサブスクライブや発行などの Amazon SNS アクションの実行を外部アカウントに許可できます。信頼ゾーン外のアカウントからのプリンシパルが Amazon SNS トピックに対して操作を実行できる場合、そのトピックは外部からのアクセスが可能です。Amazon SNS トピックの作成時にポリシーで Everyone を選択すると、そのトピックは一般公開されます。AddPermission は、外部アクセスを許可するリソースベースのポリシーを Amazon SNS トピックに追加するもう 1 つの方法です。

## Amazon Elastic Block Store ボリュームスナップショット

Amazon Elastic Block Store ボリュームスナップショットにリソースベースのポリシーはありません。スナップショットは Amazon EBS の共有アクセス許可を通じて共有されます。Amazon EBS ボリュームスナップショットの場合、IAM Access Analyzer は外部エンティティによるスナップショットへのアクセスを許可するアクセスコントロールリストを分析します。Amazon EBS ボリュームスナップショットは、暗号化されている場合に外部アカウントと共有できます。暗号化されていないボリュームスナップショットは、外部アカウントと共有してパブリックアクセス権を付与することができます。共有設定は、スナップショットの CreateVolumePermissions 属性にあります。お客様が Amazon EBS スナップショットの外部アクセスをプレビューするときは、スナップショットが暗号化されていることを示すものとして、暗号化キーを指定することができます。これは、IAM Access Analyzer プレビューが Secrets Manager シークレットを処理する方法に似ています。

## Amazon Relational Database Service DB スナップショット

Amazon RDS DB スナップショットにリソースベースのポリシーはありません。DB スナップショットは Amazon RDS データベースのアクセス許可を通じて共有され、共有できるのは手動の DB スナップショットのみです。Amazon RDS DB スナップショットの場合、IAM Access Analyzer は外部エンティティによるスナップショットへのアクセスを許可するアクセスコントロールリストを分析します。暗号化されていない DB スナップショットはパブリックにすることができます。暗号化された DB スナップショットをパブリック共有することはできませんが、最大 20 個の他のアカウントと共有できます。詳細については、「[DB スナップショットの作成](#)」を参照してください。IAM Access Analyzer は、データベースの手動スナップショットを Amazon S3 バケットなどにエクスポートする機能を、信頼できるアクセスと見なします。

### Note

IAM Access Analyzer は、データベース自体に直接設定されたパブリックまたはクロスアカウントアクセスを識別しません。IAM Access Analyzer は、Amazon RDS DB スナップショットに設定されたパブリックまたはクロスアカウントアクセスの結果のみを識別します。



## Amazon Relational Database Service DB クラスタースナップショット

Amazon RDS DB クラスタースナップショットにリソースベースのポリシーはありません。スナップショットは Amazon RDS DB クラスターのアクセス許可を通じて共有されます。Amazon RDS DB クラスターの場合、IAM Access Analyzer は外部エンティティによるスナップショットへのアクセスを許可するアクセスコントロールリストを分析します。暗号化されていないクラスタースナップショットはパブリックにすることができます。暗号化されたクラスタースナップショットをパブリック共有することはできません。暗号化されていないクラスタースナップショットと暗号化されたクラスタースナップショットは、どちらも最大 20 個の他のアカウントと共有できます。詳細については、「[DB クラスタースナップショットの作成](#)」を参照してください。IAM Access Analyzer は、DB クラスタースナップショットを Amazon S3 バケットなどにエクスポートする機能を、信頼できるアクセスと見なします。

### Note

IAM Access Analyzer の結果には、AWS Resource Access Manager を使用した Amazon RDS DB クラスターおよびクローンの別の AWS アカウントまたは組織との共有のモニタリングは含まれません。IAM Access Analyzer は、Amazon RDS DB クラスタースナップショットに設定されたパブリックまたはクロスアカウントアクセスの結果のみを識別します。

## Amazon Elastic Container Registry リポジトリ

Amazon ECR リポジトリの場合、IAM Access Analyzer はリソースベースのポリシーを分析します。これには、外部エンティティによるリポジトリへのアクセスを許可するポリシー内の条件ステートメントが含まれます (Amazon SNS トピックおよび Amazon EFS ファイルシステムといった他のリソースタイプと同様です)。Amazon ECR リポジトリの場合、外部からの利用が可能であるとみなされるには、プリンシパルがアイデンティティベースのポリシーを通じて `ecr:GetAuthorizationToken` へのアクセス許可を持っている必要があります。

## Amazon Elastic File System ファイルシステム

Amazon EFS ファイルシステムの場合、IAM Access Analyzer はポリシーを分析します。これには、外部エンティティによるファイルシステムへのアクセスを許可するポリシーの条件ステートメントが含まれます。信頼ゾーン外のアカウントからのプリンシパルが Amazon EFS ファイルシステムに対して操作を実行できる場合、そのファイルシステムは外部からのアクセスが可能です。アクセス権は、IAM を使用するファイルシステムポリシー、およびファイルシステムのマウント方法によって

定義されます。例えば、Amazon EFS ファイルシステムを別のアカウントにマウントすると、そのアカウントが組織内にあり、その組織を信頼ゾーンとして定義した場合を除き、外部からのアクセスが可能であると見なされます。パブリックサブネットを持つ仮想プライベートクラウドからファイルシステムをマウントする場合、そのファイルシステムは外部からのアクセスが可能です。Amazon EFS を AWS Transfer Family で使用する場合、ファイルシステムがパブリックアクセスを許可していると、ファイルシステムとは異なるアカウントが所有する Transfer Family サーバーから受信したファイルシステムのアクセス要求がブロックされます。

## Amazon DynamoDB Streams

外部エンティティが DynamoDB ストリームにアクセスすることを許可するクロスアカウントアクションが DynamoDB ポリシーによって少なくとも 1 つ許可される場合、IAM Access Analyzer は検出結果を生成します。DynamoDB でサポートされるクロスアカウントアクションの詳細については、「Amazon DynamoDB デベロッパーガイド」の「[リソースベースのポリシーでサポートされる IAM アクション](#)」を参照してください。

## Amazon DynamoDB テーブル

DynamoDB ポリシーで外部エンティティが DynamoDB テーブルまたはインデックスにアクセスすることを許可するクロスアカウントアクションを少なくとも 1 つ許可している場合、IAM Access Analyzer は、DynamoDB テーブルの検出結果を生成します。DynamoDB でサポートされるクロスアカウントアクションの詳細については、「Amazon DynamoDB デベロッパーガイド」の「[リソースベースのポリシーでサポートされる IAM アクション](#)」を参照してください。

## IAM Access Analyzer の設定

AWS Organizations 管理アカウントで AWS Identity and Access Management Access Analyzer を設定している場合は、組織の IAM Access Analyzer を管理するための代理管理者として組織内のメンバーアカウントを追加できます。代理管理者には、組織内でアナライザーを作成および管理するためのアクセス許可があります。管理アカウントのみが代理管理者を追加できます。

### IAM Access Analyzer の代理管理者

IAM Access Analyzer の代理管理者は、組織全体のアクセスを分析するアナライザーを作成および管理するためのアクセス許可を持つ、組織内のメンバーアカウントです。代理管理者を追加、削除、または変更できるのは、管理アカウントのみです。

代理管理者を追加した場合は、後で代理管理者のアカウントを別のものに変更できます。これを行うと、以前の代理管理者アカウントは、組織全体のアクセスを分析するためにそのアカウントを使用し



て作成されたすべてのアナライザーに対するアクセス許可を失います。これらのアナライザーは、無効状態に移行し、新しい結果を生成したり、既存の結果を更新したりしなくなります。これらのアナライザーの既存の調果にもアクセスできなくなります。後で、このアカウントを代理管理者として設定することで、アナライザーと結果に再度アクセスできます。同じアカウントを代理管理者として使用しないことがわかっている場合は、代理管理者を変更する前にアナライザーを削除することもできます。これにより、すべての生成された結果が削除されます。新しい代理管理者で新しいアナライザーを作成すると、同じ結果の新しいインスタンスが生成されます。結果は、失われることなく、別のアカウントの新しいアナライザー用に生成されます。また、組織の管理アカウントには管理者権限もあるため、このアカウントを使用して組織の結果に引き続きアクセスできます。新しい代理管理者は、組織内のリソースのモニタリングを開始するために、IAM Access Analyzer の新しいアナライザーを作成する必要があります。

代理管理者が AWS 組織から離れると、代理管理権限はアカウントから削除されます。信頼ゾーンとして組織を持つアカウント内のすべてのアナライザーは、無効状態に移行します。これらのアナライザーの既存の調果にもアクセスできなくなります。

管理アカウントで初めてアナライザーを設定する場合は、IAM Access Analyzer コンソールの [アナライザー設定] ページで [代理管理者を追加] を選択できます。

#### Note

IAM Access Analyzer では、1 か月あたりにアナライザーごとに分析された IAM ロールとユーザーの数に基づいて、未使用のアクセスアナライザーに対する料金が発生します。管理アカウントと代理管理者アカウントで未使用のアクセスアナライザーを作成すると、両方の未使用のアクセスアナライザーに対する料金が発生します。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

コンソールを使用して代理管理者を追加するには

1. 組織のマスターアカウントを使用して AWS コンソールにログインします。
2. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
3. [アクセスアナライザー] で、[アナライザー設定] を選択します。
4. [Add delegated administrator (代理管理者の追加)] を選択します。
5. [代理管理者] フィールドに、代理管理者にする組織のメンバーアカウントの AWS アカウント番号を入力します。

アカウントは、組織のメンバーである必要があります。

## 6. [Save changes] (変更の保存) をクリックします。

AWS CLI または AWS SDK を使用して代理管理者を追加するには

AWS CLI、AWS API (AWS SDK を使用)、または AWS CloudFormation を使用して、代理管理者アカウントで組織全体のアクセスを分析するためのアナライザーを作成する場合、AWS Organizations API を使用して IAM Access Analyzer のサービスアクセスを有効にし、メンバーアカウントを代理管理者として登録する必要があります。

1. AWS Organizations で IAM Access Analyzer の信頼されたサービスアクセスを有効にします。  
『AWS Organizations ユーザーガイド』の「[信頼されたアクセスを有効または無効にする方法](#)」を参照してください。
2. AWS [AWS Organizations](#) API オペレーションまたは RegisterDelegatedAdministrator register-delegated-administrator コマンドを使用して、AWS CLI 組織の有効なメンバーアカウントを代理管理者として登録します。

代理管理者を変更した場合、新しい管理者は組織内のリソースへのアクセスをモニタリングするために、まずアナライザーを作成する必要があります。

## アナライザーの削除

[アナライザー設定] ページから、既存の外部アクセスアナライザーや未使用のアクセスアナライザーを削除できます。アナライザーを削除すると、そのアナライザーで指定されているリソースはモニタリングされなくなり、新しい結果は生成されなくなります。アナライザーによって生成された結果はすべて削除されます。

結果を生成したアナライザーが削除されたために結果が削除された場合は、アナライザーが削除されてから 2 日以内に EventBridge にイベントが送信されます。アナライザーが削除されてから Security Hub の結果が削除されるまでには、最大 90 日かかることがあります。

### アナライザーを削除する方法

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [アクセスアナライザー] で、[アナライザー設定] を選択します。
3. 削除するアナライザーを選択し、[削除] を選択します。
4. 確認のテキストボックスで「delete」と入力し、[削除] を選択します。

## アーカイブルール

アーカイブルールは、ルールの作成時に定義した基準を満たす新しい結果を自動的にアーカイブします。また、アーカイブルール基準を満たす既存の結果をアーカイブするために、アーカイブルールを遡及的に適用することもできます。例えば、定期的にアクセスを許可する特定の Amazon S3 バケットに関する結果を自動的にアーカイブするためのアーカイブルールを作成できます。または、特定のプリンシパルに対して複数のリソースへのアクセスを許可する場合は、そのプリンシパルに許可したアクセスに関して生成される新しい結果を自動的にアーカイブするルールを作成できます。これにより、セキュリティ上のリスクを示している可能性があるアクティブな結果にのみ集中できます。

アーカイブルールを作成すると、ルールの基準に一致する新しい結果のみが自動的にアーカイブされます。既存の結果は自動的にアーカイブされません。ルールを作成するときは、基準ごとに最大 20 個の値をルールに含めることができます。アーカイブルールの作成または更新に使用できるフィルターキーのリストについては、「[IAM Access Analyzer フィルターキーにアクセスする](#)」を参照してください。

### Note

アーカイブルールを作成または編集する場合、ルールのフィルターに含めた値は IAM Access Analyzer で検証されません。例えば、AWS アカウントに一致するルールを追加すると、IAM Access Analyzer はフィールドの任意の値を (有効な AWS アカウント番号でなくても) 受け入れます。

アーカイブルールを作成するには

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [アクセスアナライザー] を選択し、[アナライザー設定] を選択します。
3. [アナライザー] セクションで、アーカイブルールを作成する対象のアナライザーを選択します。
4. [アーカイブルール] タブで [アーカイブルールを作成] を選択します。
5. デフォルト名を変更する場合は、ルールの名前を入力します。
6. [Rule (ルール)] セクションの [Criteria (基準)] で、ルールに一致させるプロパティを選択します。
7. プロパティ値の条件 ([含む]、[である]、[等しくない] など) を選択します。

使用できる演算子は、選択したプロパティによって異なります。

- 必要に応じて、プロパティに別の値を追加するか、ルールに別の基準を追加します。外部アクセスの結果の場合、ルールによってパブリックアクセスの新しい結果がアーカイブされないようにするには、条件 [パブリックアクセス] も追加し、[false] に設定します。

基準に別の値を追加するには、[Add another value (別の値を追加)] を選択します。ルールに別の条件を追加するには、[条件を追加] を選択します。

- 基準と値の追加が完了したら、[Create rule (ルールの作成)] を選択して、新しい結果にのみルールを適用します。ルール基準に基づいて新規および既存の結果をアーカイブするには、[Create and archive active findings (アクティブな結果を作成してアーカイブする)] を選択します。[Results (結果)] セクションでは、アーカイブルールが適用されるアクティブな結果のリストを確認できます。

例えば、外部アクセスの結果に対して、Amazon S3 バケットに関するすべての結果を自動的にアーカイブするためのルールを作成するには、[リソースタイプ] を選択し、条件として [である] を選択します。次に、[値] リストから [S3 バケット] を選択します。

未使用のアクセスの結果に対して、特定のアカウントに関するすべての結果を自動的にアーカイブするためのルールを作成するには、[リソース所有者アカウント] を選択し、条件として [等しい] を選択します。[値] テキストボックスに AWS アカウント ID を入力します。

引き続き、環境に応じてルールをカスタマイズするための基準を定義し、[ルールを作成] を選択します。

新しいルールを作成して複数の基準を追加した場合、ルールから 1 つの基準を削除するには、[Remove this criterion (この基準を削除)] を選択できます。基準に追加した値を削除するには、[Remove value (値の削除)] を選択できます。

アーカイブルールを編集するには

- [名前] 列で、編集するルールの名前を選択します。

一度に編集できるアーカイブルールは 1 つだけです。

- 各条件について、新しい基準の追加や、既存の基準と値の削除を行います。
- 新しい結果にのみルールを適用するには、[Save changes (変更を保存)] を選択します。ルール基準に基づいて新規および既存の結果をアーカイブするには、[Save and archive active findings (アクティブな結果を保存してアーカイブする)] を選択します。

## アーカイブルールを削除するには

1. 削除するルールのチェックボックスをオンにします。
2. [Delete] (削除) をクリックします。
3. [Delete archive rule (アーカイブルールの削除)] 確認ダイアログに「**delete**」と入力し、[Delete (削除)] を選択します。

ルールは、現在のリージョンのアナライザーからのみ削除されます。他のリージョンで作成したアーカイブルールは、アナライザーごとに個別に削除する必要があります。

## Amazon EventBridge を使用した AWS Identity and Access Management Access Analyzer のモニタリング

このトピックでは、Amazon EventBridge を使用した IAM Access Analyzer の結果のモニタリングと、アクセスプレビュー方法について説明します。EventBridge は、Amazon CloudWatch Events の新しいバージョンです。

### 結果のイベント

IAM Access Analyzer は、各結果の生成時、既存の結果の状態の変更時、および結果の削除時に EventBridge にイベントに送信します。結果および結果に関する通知を受信するには、Amazon EventBridge でイベントルールを作成する必要があります。イベントルールを作成するときに、ルールに基づいてトリガーするターゲットアクションを指定することもできます。例えば、新しい結果のイベントが IAM Access Analyzer から返されたときに Amazon SNS トピックをトリガーするイベントルールを作成できます。

### アクセスプレビューイベント

IAM Access Analyzer は、アクセスプレビューごとにイベントを EventBridge に送信し、ステータスを変更します。これには、アクセスプレビューが最初に作成されたとき (ステータス Creating)、アクセスプレビューが完了したとき (ステータス Completed)、またはアクセスプレビューの作成が失敗したとき (ステータス Failed) が含まれます。Access Preview に関する通知を受信するには、EventBridge でイベントルールを作成する必要があります。イベントルールを作成するときに、ルールに基づいてトリガーするターゲットアクションを指定することができます。たとえば、完了したアクセスプレビューのイベントが IAM Access Analyzer から返されたときに Amazon SNS トピックをトリガーするイベントルールを作成できます。

## イベントの通知頻度

IAM Access Analyzer は、アカウント内でイベントが発生してから 1 時間以内に新しい結果や、ステータス更新に関するイベントを EventBridge に送信します。IAM Access Analyzer は、保存期間の期限が切れたために解決済みの結果が削除されたときにも EventBridge にイベントを送信します。結果を生成したアナライザーが削除されたために結果が削除された場合は、アナライザーが削除されてから約 24 時間後に Eventbridge にイベントが送信されます。結果が削除されても、結果のステータスは変更されません。代わりに、isDeleted 属性が true に設定されます。また、IAM Access Analyzer は、新しく作成されたアクセスプレビューおよびアクセスプレビューステータスの変更に関するイベントを EventBridge に送信します。

## 外部アクセスの検出結果イベントの例

次に示すのは、EventBridge に送信される IAM Access Analyzer 外部アクセスの結果イベントの例です。表示されている id は、Eventbridge のイベントの ID です。詳細については、「[EventBridge のイベントとイベントパターン](#)」を参照してください。

detail オブジェクトでは、accountId 属性と region 属性の値は、結果で報告されたアカウントとリージョンを参照します。isDeleted 属性は、イベントが削除対象の結果に関連するものであるかどうかを示します。id は結果 ID です。resources 配列は、結果を生成したアナライザーの ARN を持つシングルトンです。

```
{
  "account": "111122223333",
  "detail": {
    "accountId": "111122223333",
    "action": [
      "s3:GetObject"
    ],
    "analyzedAt": "2019-11-21T01:22:22Z",
    "condition": {},
    "createdAt": "2019-11-20T04:58:50Z",
    "id": "22222222-dcba-4444-dcba-333333333333",
    "isDeleted": false,
    "isPublic": false,
    "principal": {
      "AWS": "999988887777"
    },
    "region": "us-west-2",
    "resource": "arn:aws:s3:::my-bucket",
    "resourceType": "AWS::S3::Bucket",
```

```
    "status": "ACTIVE",
    "updatedAt": "2019-11-21T01:14:07Z",
    "version": "1.0"
  },
  "detail-type": "Access Analyzer Finding",
  "id": "11111111-2222-4444-aaaa-333333333333",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2019-11-21T01:22:33Z",
  "version": "0"
}
```

IAM Access Analyzer は、エラー結果に関するイベントも EventBridge に送信します。エラー結果は、IAM Access Analyzer が分析対象のリソースにアクセスできない場合に生成されます。エラー結果のイベントには、次の例に示すように `error` 属性が含まれます。

```
{
  "account": "111122223333",
  "detail": {
    "accountId": "111122223333",
    "analyzedAt": "2019-11-21T01:22:22Z",
    "createdAt": "2019-11-20T04:58:50Z",
    "error": "ACCESS_DENIED",
    "id": "22222222-dcba-4444-dcba-333333333333",
    "isDeleted": false,
    "region": "us-west-2",
    "resource": "arn:aws:s3::my-bucket",
    "resourceType": "AWS::S3::Bucket",
    "status": "ACTIVE",
    "updatedAt": "2019-11-21T01:14:07Z",
    "version": "1.0"
  },
  "detail-type": "Access Analyzer Finding",
  "id": "11111111-2222-4444-aaaa-333333333333",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2019-11-21T01:22:33Z",
}
```



```
"version": "0"  
}
```

## 未使用のアクセスの検出結果関連イベントの例

次に示すのは、EventBridge に送信される IAM Access Analyzer 未使用アクセスの結果イベントの例です。表示されている id は、Eventbridge のイベントの ID です。詳細については、「[EventBridge のイベントとイベントパターン](#)」を参照してください。

detail オブジェクトでは、accountId 属性と region 属性の値は、結果で報告されたアカウントとリージョンを参照します。isDeleted 属性は、イベントが削除対象の結果に関連するものであるかどうかを示します。id は結果 ID です。

```
{  
  "version": "0",  
  "id": "dc7ce3ee-114b-3243-e249-7f10f9054b21",  
  "detail-type": "Unused Access Finding for IAM entities",  
  "source": "aws.access-analyzer",  
  "account": "123456789012",  
  "time": "2023-09-29T17:31:40Z",  
  "region": "us-west-2",  
  "resources": [  
    "arn:aws:access-analyzer:us-west-2:123456789012:analyzer/  
integTestLongLivingAnalyzer-D0-NOT-DELETE"  
  ],  
  "detail": {  
    "findingId": "b8ae0460-5d29-4922-b92a-ba956c986277",  
    "resource": "arn:aws:iam::111122223333:role/FindingIntegTestFakeRole",  
    "resourceType": "AWS::IAM::Role",  
    "accountId": "111122223333",  
    "createdAt": "2023-09-29T17:29:18.758Z",  
    "updatedAt": "2023-09-29T17:29:18.758Z",  
    "analyzedAt": "2023-09-29T17:29:18.758Z",  
    "previousStatus": "",  
    "status": "ACTIVE",  
    "version": "62160bda-8e94-46d6-ac97-9670930d8ffb",  
    "isDeleted": false,  
    "findingType": "UnusedPermission",  
    "numberOfUnusedServices": 0,  
    "numberOfUnusedActions": 1  
  }  
}
```



IAM Access Analyzer は、エラー結果に関するイベントも EventBridge に送信します。エラー結果は、IAM Access Analyzer が分析対象のリソースにアクセスできない場合に生成されます。エラー結果のイベントには、次の例に示すように `error` 属性が含まれます。

```
{
  "version": "0",
  "id": "c2e7aa1a-4df7-7652-f33e-64113b8997d4",
  "detail-type": "Unused Access Finding for IAM entities",
  "source": "aws.access-analyzer",
  "account": "111122223333",
  "time": "2023-10-31T20:26:12Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/ba811f91-
de99-41a4-97c0-7481898b53f2"
  ],
  "detail": {
    "findingId": "b01a34f2-e118-46c9-aef8-0d8526b495c7",
    "resource": "arn:aws:iam::123456789012:role/TestRole",
    "resourceType": "AWS::IAM::Role",
    "accountId": "444455556666",
    "createdAt": "2023-10-31T20:26:08.647Z",
    "updatedAt": "2023-10-31T20:26:09.245Z",
    "analyzedAt": "2023-10-31T20:26:08.525Z",
    "previousStatus": "",
    "status": "ACTIVE",
    "version": "7c7a72a2-7963-4c59-ac71-f0be597010f7",
    "isDeleted": false,
    "findingType": "UnusedIAMRole",
    "error": "INTERNAL_ERROR"
  }
}
```

## アクセスプレビューイベントの例

次の例では、Access Preview を作成するときに EventBridge に送信された最初のイベントのデータを示しています。resources 配列は、アクセスプレビューが関連付けられているアナライザーの ARN を持つシングルトンです。detail オブジェクトでは、id はアクセスプレビューIDを参照し、configuredResources はアクセスプレビューが作成されたリソースを参照します。status は Creating で、アクセスプレビューのステータスを参照します。previousStatus は、アクセスプレビューが作成されたばかりであるため、指定されていません。

```
{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.00Z",
    "region": "us-west-2",
    "status": "CREATING",
    "version": "1.0"
  },
  "detail-type": "Access Preview State Change",
  "id": "aaaabbbb-2222-3333-4444-555566667777",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2020-02-20T00:00:00.00Z",
  "version": "0"
}
```

次の例は、ステータスが `Creating` から `Completed` に変更されたアクセスプレビューのために EventBridge に送信されるイベントのデータを示しています。詳細オブジェクトでは、`id` は、アクセスプレビュー ID を参照します。`status` および `previousStatus` は、以前のステータスが `Creating` であり、現在のステータスは `Completed` である、アクセスプレビューのステータスを参照します。

```
{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.000Z",
    "previousStatus": "CREATING",
    "region": "us-west-2",
    "status": "COMPLETED",
    "version": "1.0"
  },
}
```

```

"detail-type": "Access Preview State Change",
"id": "11112222-3333-4444-5555-666677778888",
"region": "us-west-2",
"resources": [
  "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
],
"source": "aws.access-analyzer",
"time": "2020-02-20T00:00:00.00Z",
"version": "0"
}

```

次の例は、ステータスが `Creating` から `Failed` に変更されたアクセスプレビューのために EventBridge に送信されるイベントのデータを示しています。detail オブジェクトでは、id は、アクセスプレビュー ID を参照します。status および previousStatus は、以前のステータスが `Creating` であり、現在のステータスは `Failed` である、アクセスプレビューのステータスを参照します。statusReason フィールドには、無効なリソース構成のためにアクセスプレビューが失敗したことを示す理由コードが表示されます。

```

{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.00Z",
    "previousStatus": "CREATING",
    "region": "us-west-2",
    "status": "FAILED",
    "statusReason": {
      "code": "INVALID_CONFIGURATION"
    },
    "version": "1.0"
  },
  "detail-type": "Access Preview State Change",
  "id": "99998888-7777-6666-5555-444433332222",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2020-02-20T00:00:00.00Z",
}

```

```
"version": "0"  
}
```

## コンソールを使用したイベントルールの作成

次の手順では、コンソールを使用してイベントルールを作成する方法について説明します。

1. Amazon EventBridge コンソール (<https://console.aws.amazon.com/events/>) を開きます。
2. 次の値を使用して、検索イベントまたはアクセスプレビューイベントをモニタリングする EventBridge ルールを作成します。
  - ルールタイプ では、イベントパターンを持つルール を選択します。
  - Event source] (イベントソース) では、Other] (その他) を選択します。
  - [Event pattern] (イベントパターン) で [Custom patterns (JSON editor)] (カスタムパターン (JSON エディター)) を選択し、次のイベントパターン例のいずれかをテキストエリアに貼り付けます。
  - 外部アクセスまたは未使用のアクセスの検出結果イベントに基づいてルールを作成するには、次のパターン例を使用します。

```
{  
  "source": [  
    "aws.access-analyzer"  
  ],  
  "detail-type": [  
    "Access Analyzer Finding"  
  ]  
}
```

- 未使用のアクセスの検出結果イベントのみに基づいてルールを作成するには、次のパターン例を使用します。

```
{  
  "source": [  
    "aws.access-analyzer"  
  ],  
  "detail-type": [  
    "Unused Access Finding for IAM entities"  
  ]  
}
```

**Note**

外部アクセスの検出結果イベントのみに基づいてルールを作成することはできません。

- アクセプレビューイベントに基づいてルールを作成するには、次のパターン例を使用します。

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Preview State Change"
  ]
}
```

- [Target types] (ターゲットタイプ) で [AWS service] AWS Lambda のサービス) を選択し、[Select a target] (ターゲットを選択) でターゲット (Amazon SNS のトピックや 関数など) を選択します。ターゲットは、ルールで定義したイベントパターンに一致するイベントが返されたときにトリガーされます。

ルールの作成に関する詳細については、「Amazon EventBridge ユーザーガイド」の「[イベントに反応する Amazon EventBridge ルールの作成](#)」を参照してください。

## CLI を使用したイベントルールの作成

1. 以下を使用して、AWS CLI を使用した Amazon EventBridge のルールを作成します。ルール名 *TestRule* を、使用するルールの名前に置き換えます。

```
aws events put-rule --name TestRule --event-pattern "{\"source\": [\"aws.access-analyzer\"]}"
```

2. ルールをカスタマイズして、生成された結果のサブセット (特定の属性を持つ結果など) に対してのみターゲットアクションをトリガーできます。次の例は、ステータスが Active の結果に対してのみターゲットアクションをトリガーするルールを作成する方法を示しています。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-analyzer\"],\"detail-type\":[\"Access Analyzer Finding\"],\"detail\":{\"status\":[\"ACTIVE\"]}}"
```

次の例は、ステータスが `Creating` から `Completed` のアクセスプレビューに対してのみターゲットアクションをトリガーするルールを作成する方法を示しています。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-analyzer\"],\"detail-type\":[\"Access Preview State Change\"],\"detail\":{\"status\":[\"COMPLETED\"]}}"
```

- 作成したルールのターゲットとして Lambda 関数を定義するには、次のコマンド例を使用します。環境に応じて、ARN のリージョンと関数名を置き換えます。

```
aws events put-targets --rule TestRule --targets Id=1,Arn=arn:aws:lambda:us-east-1:111122223333:function:MyFunction
```

- ルールのターゲットを呼び出すために必要なアクセス許可を追加します。次の例は、前述の例に従って、Lambda 関数にアクセス許可を付与する方法を示しています。

```
aws lambda add-permission --function-name MyFunction --statement-id 1 --action 'lambda:InvokeFunction' --principal events.amazonaws.com
```

## Access Analyzer と AWS Security Hub を統合する

[AWS Security Hub](#) では、AWS のセキュリティ状態を総合的に把握することができ、セキュリティ業界標準およびベストプラクティスに照らし合わせて環境をチェックすることができます。Security Hub は、AWS アカウント、サービス、およびサポートされているサードパーティーパートナー製品からセキュリティデータを収集するため、セキュリティの傾向を分析し、最も優先度の高いセキュリティ問題を特定するのに役立てることができます。

AWS Identity and Access Management Access Analyzer を Security Hub と統合すると、IAM Access Analyzer から Security Hub に検出結果を送信できるようになります。Security Hub では、このような検出結果をセキュリティ体制の分析に含めることができます。

### 目次

- [IAM Access Analyzer が Security Hub に検出結果を送信する方法](#)

- [IAM Access Analyzer が送信する検出結果のタイプ](#)
- [結果が送信されるまでのレイテンシー](#)
- [Security Hub が使用できない場合の再試行](#)
- [Security Hub の既存の結果を更新する](#)
- [Security Hub での IAM Access Analyzer の検出結果の表示](#)
  - [Security Hub での IAM Access Analyzer の結果名の解釈](#)
- [IAM Access Analyzer からの一般的な検出結果](#)
- [統合の有効化と構成](#)
- [検出結果の送信を停止する方法](#)

## IAM Access Analyzer が Security Hub に検出結果を送信する方法

Security Hub では、セキュリティの問題が調査結果として追跡されます。結果の中には、他の AWS のサービスやサードパーティーのパートナーが検出した問題に由来するものもあります。Security Hub には、セキュリティの問題を検出し、調査結果を生成するために使用する一連のルールもあります。

Security Hub には、これらすべてのソースからの結果を管理するためのツールが用意されています。検出結果の一覧を表示およびフィルタリングして、検出結果の詳細を表示できます。「AWS Security Hub ユーザーガイド」の「[検出結果の表示](#)」を参照してください。検出結果の調査状況を追跡することもできます。「AWS Security Hub ユーザーガイド」の「[Taking action on findings](#)」(検出結果に対するアクションの実行)を参照してください。

Security Hub のすべての検出結果で、AWS Security Finding 形式 (ASFF) と呼ばれる標準の JSON 形式が使用されます。ASFF には、問題のソース、影響を受けるリソース、および検出結果の現在のステータスに関する詳細が含まれます。[AWS ユーザーガイド](#) の「AWS Security Hub Security Finding 形式 (ASFF)」を参照してください。

AWS Identity and Access Management Access Analyzer は Security Hub に結果を送信する AWS サービスの 1 つです。未使用のアクセスの場合、IAM Access Analyzer は、IAM ユーザーまたはロールに付与されているが使用されていないアクセス権限を検出し、それぞれの検出結果を生成します。その後、IAM Access Analyzer は検出結果を Security Hub に送信します。外部アクセスの場合、IAM Access Analyzer は、組織またはアカウント内の[サポートされているリソース](#)への外部プリンシパルへのパブリックアクセスまたはクロスアカウントアクセスを許可するポリシーステートメントを検出します。IAM Access Analyzer はパブリックアクセスの検出結果を生成し、それを Security Hub に送信します。クロスアカウントアクセスの場合、IAM Access Analyzer は、一度に 1 つの外部プリンシ

パルに対して 1 つの検出結果を Security Hub に送信します。IAM Access Analyzer に複数のクロスアカウント検出結果がある場合、IAM Access Analyzer が次のクロスアカウント検出結果を提供する前に、単一の外部プリンシパルに対する Security Hub 検出結果を解決する必要があります。アナライザーの信頼ゾーン外のクロスアカウントアクセスを持つ外部プリンシパルの完全なリストについては、IAM Access Analyzer で検出結果を表示する必要があります。

### IAM Access Analyzer が送信する検出結果のタイプ

IAM Access Analyzer は、[AWS Security Finding Format \(ASFF\)](#) を使用して検出結果を Security Hub に送信します。ASFF では、Types フィールドが検出結果タイプを提供します。IAM Access Analyzer からの検出結果では、Types に対して次の値が示される可能性があります。

- 外部アクセスの検出結果 – 影響/データ漏えい/外部アクセスの許可
- 外部アクセスの検出結果 – ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/外部アクセスの許可
- 未使用のアクセスの検出結果 – ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用のアクセス許可
- 未使用のアクセスの検出結果 – ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ロール
- 未使用のアクセスの検出結果 – ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ユーザーパスワード
- 未使用のアクセスの検出結果 – ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ユーザーアクセスキー

### 結果が送信されるまでのレイテンシー

IAM Access Analyzer が新しい結果を作成すると、通常は 30 分以内に Security Hub に送信されます。まれに、特定の条件下で、ポリシーの追加や更新が IAM Access Analyzer に通知されないことがあります。たとえば、Amazon S3 アカウントレベルのブロックパブリックアクセス設定への変更には、最大 12 時間かかることがあります。また、AWS CloudTrail ログ配信で配信の問題がある場合、ポリシーを変更しても、検出結果でレポートされたリソースの再スキャンはトリガーされません。このような場合、IAM Access Analyzer は、次の定期スキャン時に新しいポリシーまたは更新されたポリシーを分析します。

### Security Hub が使用できない場合の再試行

Security Hub を使用できない場合、IAM Access Analyzer は定期的に検出結果の送信を再試行します。



## Security Hub の既存の結果を更新する

結果を Security Hub に送信した後、AWS Identity and Access Management Access Analyzer は結果アクティビティの追加の観測を反映するために、更新を Security Hub に送信します。更新は同じ検出結果に反映されます。

IAM Access Analyzer は外部アクセスの検出結果をリソースごとにグループ化するため、Security Hub 内のリソースに関する検出結果は、IAM Access Analyzer 内のリソースに関する結果の少なくとも 1 つがアクティブである場合にのみアクティブになります。リソースに関する IAM Access Analyzer のすべての検出結果がアーカイブまたは解決されると、Security Hub の検出結果がアーカイブされます。パブリックアクセスとクロスアカウントアクセスの間でポリシーアクセスを変更すると、Security Hub の検出結果も更新されます。この更新には、検出結果の種類、タイトル、説明、および重大度の変更を含めることができます。

IAM Access Analyzer は未使用のアクセスの検出結果をリソースごとにグループ化しないため、IAM Access Analyzer で未使用のアクセスの検出結果が解決されると、Security Hub の検出結果が解決されます。未使用のアクセスの結果を生成した IAM ユーザーまたはロールを更新すると、Security Hub の結果が更新されます。

## Security Hub での IAM Access Analyzer の検出結果の表示

Security Hub で IAM Access Analyzer の検出結果を表示するには、概要ページの [AWS: IAM Access Analyzer] セクションの [結果を参照] を選択します。または、ナビゲーションパネルから [検出結果] を選択することもできます。次に、値が **IAM Access Analyzer** である [Product name:] フィールドを選択して、AWS Identity and Access Management Access Analyzer の検出結果のみを表示するようにフィルタリングすることができます。

## Security Hub での IAM Access Analyzer の結果名の解釈

AWS Identity and Access Management Access Analyzer は、AWS Security Finding Format (ASFF) を使用して検出結果を Security Hub に送信します。ASFF では、[Types] フィールドに検出結果タイプが表示されます。ASFF タイプは、AWS Identity and Access Management Access Analyzer とは異なる命名規則を使用します。次の表は、Security Hub に表示される AWS Identity and Access Management Access Analyzer の検出結果に関連付けられたすべての ASFF タイプの詳細を示しています。

ASFF 結果タイプ	Security Hub 検出結果のタイトル	説明
エフェクト/データエクスポート/外部アクセス許可	<resource ARN> パブリックアクセスを許可	リソースにアタッチされたリソースベースのポリシーは、リソースに対するすべての外部プリンシパルへのパブリックアクセスを許可します。
ソフトウェアと構成のチェック/AWS セキュリティのベストプラクティス/外部アクセスの許可	<resource ARN> クロスアカウントアクセスを許可	リソースにアタッチされたリソースベースのポリシーは、アナライザーの信頼ゾーン外の外部プリンシパルへのクロスアカウントアクセスを許可します。
ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用のアクセス許可	<リソース ARN> には、未使用のアクセス許可が含まれません	ユーザーまたはロールには、未使用のサービスおよびアクションのアクセス許可が含まれません。
ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ロール	<リソース ARN> には、未使用の IAM ロールが含まれます	ユーザーまたはロールには、未使用の IAM ロールが含まれます。
ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ユーザーパスワード	<リソース ARN> には、未使用の IAM ユーザーパスワードが含まれます	ユーザーまたはロールには、未使用の IAM ユーザーパスワードが含まれます。
ソフトウェアと設定のチェック/AWS セキュリティのベストプラクティス/未使用の IAM ユーザーアクセスキー	<リソース ARN> には、未使用の IAM ユーザーアクセスキーが含まれます	ユーザーまたはロールには、未使用の IAM ユーザーアクセスキーが含まれます。

## IAM Access Analyzer からの一般的な検出結果

IAM Access Analyzer は、[AWS Security Finding Format \(ASFF\)](#) を使用して検出結果を Security Hub に送信します。

以下に、外部アクセスの検出結果に関する IAM Access Analyzer からの一般的な検出結果の例を示します。

```
{
  "SchemaVersion": "2018-10-08",
  "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/my-analyzer/
arn:aws:s3::my-bucket",
  "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
  "GeneratorId": "aws/access-analyzer",
  "AwsAccountId": "111122223333",
  "Types": ["Software and Configuration Checks/AWS Security Best Practices/External
Access Granted"],
  "CreatedAt": "2020-11-10T16:17:47Z",
  "UpdatedAt": "2020-11-10T16:43:49Z",
  "Severity": {
    "Product": 1,
    "Label": "LOW",
    "Normalized": 1
  },
  "Title": "AwsS3Bucket/arn:aws:s3::my-bucket/ allows cross-account access",
  "Description": "AWS::S3::Bucket/arn:aws:s3::my-bucket/ allows cross-account access
from AWS 444455556666",
  "Remediation": {
    "Recommendation": {"Text": "If the access isn't intended, it indicates a
potential security risk. Use the console for the resource to modify or remove the
policy that grants the unintended access. You can use the Rescan button on the Finding
details page in the Access Analyzer console to confirm whether the change removed the
access. If the access is removed, the status changes to Resolved."}
  },
  "SourceUrl": "https://console.aws.amazon.com/access-analyzer/home?region=us-
west-2#/findings/details/dad90d5d-63b4-6575-b0fa-ef9c556ge798",
  "Resources": [
    {
      "Type": "AwsS3Bucket",
      "Id": "arn:aws:s3::my-bucket",
      "Details": {
        "Other": {
          "External Principal Type": "AWS",
```

```

        "Condition": "none",
        "Action Granted": "s3:GetObject,s3:GetObjectVersion",
        "External Principal": "444455556666"
    }
}
],
"WorkflowState": "NEW",
"Workflow": {"Status": "NEW"},
"RecordState": "ACTIVE"
}

```

以下に、未使用のアクセスの検出結果に関する IAM Access Analyzer からの一般的な検出結果の例を示します。

```

{
  "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-DO-NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
      "ProductName": "IAM Access Analyzer",
      "CompanyName": "AWS",
      "Region": "us-west-2",
      "GeneratorId": "aws/access-analyzer",
      "AwsAccountId": "111122223333",
      "Types": [
        "Software and Configuration Checks/AWS Security Best Practices/Unused Permission"
      ],
      "CreatedAt": "2023-09-18T16:29:09.657Z",
      "UpdatedAt": "2023-09-21T20:39:16.651Z",
      "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
      },
      "Title": "AwsIamRole/arn:aws:iam::111122223333:role/IsengardRole-DO-NOT-DELETE/contains unused permissions",
      "Description": "AWS::IAM::Role/arn:aws:iam::111122223333:role/IsengardRole-DO-NOT-DELETE/ contains unused service and action-level permissions",
      "Remediation": {

```

```
    "Recommendation": {
      "Text": "If the unused permissions aren't required, delete the permissions to
refine access to your account. Use the IAM console to modify or remove the policy that
grants the unused permissions. If all the unused permissions are removed, the status
of the finding changes to Resolved."
    }
  },
  "SourceUrl": "https://us-west-2.console.aws.amazon.com/access-analyzer/
home?region=us-west-2#/unused-access-findings?resource=arn%3Aaws%3Aiam%3A
%3A903798373645%3Arole%2FTestRole",
  "ProductFields": {
    "numberOfUnusedActions": "256",
    "numberOfUnusedServices": "15",
    "resourceOwnerAccount": "111122223333",
    "findingId": "DEM024d8d-0d3f-4d3d-99f4-299fc8a62ee7",
    "findingType": "UnusedPermission",
    "aws/securityhub/FindingId": "arn:aws:securityhub:us-west-2::product/aws/access-
analyzer/arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-D0-
NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
    "aws/securityhub/ProductName": "AM Access Analyzer",
    "aws/securityhub/CompanyName": "AWS"
  },
  "Resources": [
    {
      "Type": "AwsIamRole",
      "Id": "arn:aws:iam::111122223333:role/TestRole"
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ARCHIVED",
  "FindingProviderFields": {
    "Severity": {
      "Label": "LOW"
    },
    "Types": [
      "Software and Configuration Checks/AWS Security Best Practices/Unused Permission"
    ]
  }
}
```

```
}
```

## 統合の有効化と構成

Security Hub との統合を利用するには、Security Hub を有効にする必要があります。Security Hub を有効にする方法については、[AWS Security Hub ユーザーガイド](#)の「Security Hub の設定」を参照してください。

IAM Access Analyzer と Security Hub の両方を有効にすると、自動的に統合が有効になります。IAM Access Analyzer は、Security Hub への検出結果の送信を即時に開始します。

## 検出結果の送信を停止する方法

Security Hub への結果の送信を停止するには、Security Hub コンソールまたは API を使用できません。

「[統合からの検出結果のフローの無効化と有効化 \(コンソール\)](#)」または[AWS Security Hub ユーザーガイド](#)の「[統合からの検出結果のフローの無効化 \(Security Hub API、AWS CLI\)](#)」を参照してください。

## AWS CloudTrail を使用した IAM Access Analyzer API コールのログ記録

IAM Access Analyzer は、IAM Access Analyzer のユーザー、ロール、または AWS CloudTrail のサービスによって実行されたアクションをレコードするサービスである AWS と統合されています。CloudTrail は、IAM Access Analyzer のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、IAM Access Analyzer コンソールからの呼び出しと、IAM Access Analyzer の API オペレーションへのコードの呼び出しが含まれます。

証跡を作成する場合は、IAM Access Analyzer のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの Event history (イベント履歴)で最新のイベントを表示できます。

CloudTrail で収集された情報を使用して、IAM Access Analyzer に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、[AWS CloudTrail ユーザーガイド](#)を参照してください。

## CloudTrail での IAM Access Analyzer 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。サポートされているイベントアクティビティが IAM Access Analyzer で発生すると、そのアクティビティは

[Event history] (イベント履歴) の他の AWS サービスのイベントとともに、CloudTrail イベントにレコードされます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「[CloudTrail イベント履歴でのイベントの表示](#)」を参照してください。

AWS アカウントのイベント (IAM Access Analyzer のイベントを含む) を継続的に記録するには、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべての AWS リージョンに適用されます。追跡は、AWSパーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- [追跡を作成するための概要](#)
- [CloudTrail のサポート対象サービスと統合](#)
- [Amazon SNS の CloudTrail の通知の設定](#)
- 「[複数のリージョンから CloudTrail ログファイルを受け取る](#)」および「[複数のアカウントから CloudTrail ログファイルを受け取る](#)」

すべての IAM Access Analyzer アクションは CloudTrail によってログに記録され、[IAM Access Analyzer API リファレンス](#)に記録されます。例えば、CreateAnalyzer、CreateArchiveRule、ListFindingsの各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーティッドユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、[\[CloudTrail userIdentity Element\]](#) (CloudTrail ユーザーアイデンティティ要素) を参照してください。

## IAM Access Analyzer ログファイルエントリの概要

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、「2021年6月14日」に「Alice-tempcreds」という引き受けたロールのセッションによって実行された CreateAnalyzer オペレーションを示す CloudTrail ログエントリです。ロールセッションは、admin-tempcreds というロールによって発行されました。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIIBKEVSQ6C2EXAMPLE:Alice-tempcreds",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin-tempcreds/Alice-tempcreds",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2021-06-14T22:54:20Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin-tempcreds",
        "accountId": "111122223333",
        "userName": "admin-tempcreds"
      },
      "webIdFederationData": {}
    }
  },
  "eventTime": "2021-06-14T22:57:36Z",
  "eventSource": "access-analyzer.amazonaws.com",
  "eventName": "CreateAnalyzer",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.179",
```



```

"userAgent": "aws-sdk-java/1.12.79 Linux/5.4.141-78.230 OpenJDK_64-
Bit_Server_VM/25.302-b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/
standard",
"requestParameters": {
  "analyzerName": "test",
  "type": "ACCOUNT",
  "clientToken": "11111111-abcd-2222-abcd-222222222222",
  "tags": {
    "tagkey1": "tagvalue1"
  }
},
"responseElements": {
  "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/test"
},
"requestID": "22222222-dcba-4444-dcba-333333333333",
"eventID": "33333333-bcde-5555-bcde-444444444444",
"readOnly": false,
"eventType": "AwsApiCall",,
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

## IAM Access Analyzer フィルターキーにアクセスする










以下のフィルタキーを使用して、アーカイブ規則の定義 ([CreateArchiveRule](#))、アーカイブ規則の更新 ([UpdateArchiveRule](#))、検出結果の一覧の取得 ([ListFindings](#) および [ListFindingsV2](#))、またはリソースのアクセスプレビューの一覧の取得 ([ListAccessPreviewFindings](#)) を行うことができます。アーカイブ規則を構成するための IAM API と AWS CloudFormation の使用に違いはありません。

Criterion	説明	[Type] (タイプ)	アーカイブ規則	結果の一覧表示	アクセスプレビューの結果を一覧表示
リソース	外部プリンシパルがアクセスできるリソースを一意に識別する ARN。詳細については、「 <a href="#">Amazon</a>	文字列	 はい	 はい	 はい










Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
	<a href="#">リソースネーム (ARN)</a> 」 を参照してください。				

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
resourceType  AWS::IAM: :Role   AWS::KMS: :Key   AWS::Lambda: Function   AWS::Lambda: LayerVersion   AWS::S3:: Bucket   AWS::S3Ex press::Di rectoryBu cket   AWS::SQS: :Queue   AWS::Secr etsManage r::Secret   AWS::EFS: :FileSyst em   AWS::EC2: :Snapshot 	外部プリンシパルがアクセスできるリソースのタイプ。	文字列	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
AWS::ECR: :Repository   AWS::RDS: :DBSnapshot   AWS::RDS: :DBClusterSnapshot   AWS::SNS: :Topic   AWS::Dyna moDB::Str eam   AWS::Dyna moDB::Tab le					
resourceO wnerAccou nt	リソースを所有する 12 桁の AWS アカウント ID。詳細については、「 <a href="#">AWS アカウント識別子</a> 」を参照してください。	文字列	 はい	 はい	 はい
isPublic	パブリックアクセスを許可するポリシーを持つリソースが結果によって報告されるかどうかを示します。	ブール値	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
findingType  UnusedIAM Role   UnusedIAM UserAccess Key   UnusedIAM UserPassw ord   UnusedPer mission	結果のタイプ。未使用の アクセスの検出結果につ いては、検出結果タイプ でのフィルターのみ可能 です。	文字列	 はい	 はい	 はい
status  ACTIVE   ARCHIVED   RESOLVED	結果の現在のステータ ス。	文字列	 いいえ	 はい	 はい
error	結果に対して報告された エラーを示します。	文字列	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
principal .AWS	検出結果の Principal フィールドのリソースへのアクセスが付与されたアカウント。外部の AWS ユーザーまたはロールの 12 桁の AWS アカウント ID または ARN を入力します。詳細については、「 <a href="#">AWS アカウント識別子</a> 」を参照してください。	文字列	 はい	 はい	 はい
principal .Federated	結果でリソースにアクセスできるフェデレーション ID の ARN。詳細については、「 <a href="#">ID プロバイダーとフェデレーション</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .aws:PrincipalArn	リソースアクセスの条件として示されたプリンシパル (IAM ユーザー、ロール、またはグループ) の ARN。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
condition .aws:Prin cipalOrgID	リソースアクセスの条件として示されるプリンシパルの組織 ID。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .aws:Prin cipalOrgP aths	リソースアクセスの条件として示される組織または組織単位 (OU) ID。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .aws:Sour celp	指定した IP アドレスを使用するときに、リソースへのプリンシパルアクセスを許可する IP アドレス。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	IP ア ドレス	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
condition .aws:SourceVpc	指定された VPC を使用するときにはリソースへのプリンシパルアクセスを許可する VPC ID。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .aws:UserId	リソースへのアクセス条件として示された外部アカウントからの IAM ユーザーのユーザー ID。詳細については、「 <a href="#">AWS グローバル条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .cognito-identity. amazonaws.com:aud	検索で IAM ロールアクセスの条件として指定された Amazon Cognito ID プールの ID。詳細については、「 <a href="#">IAM および AWS STS の条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい



Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
condition .graph.facebook.com:app_id	[Login with Facebook (Facebook でログイン)] フェデレーションが結果の IAM ロールにアクセスできるようにするための条件として指定された Facebook アプリケーション ID (またはサイト ID)。詳細については、「 <a href="#">IAM および AWS STS の条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .accounts.google.com:aud	IAM ロールへのアクセス条件として指定された Google アプリケーション ID。詳細については、「 <a href="#">IAM および AWS STS の条件コンテキストキー</a> 」を参照してください。	文字列	 はい	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
condition .kms:Call erAccount	AWS を呼び出すサービスにより使用される呼び出し元エンティティ (IAM ユーザー、ロール、またはアカウントルートユーザー) を所有する AWS KMS アカウント ID。詳細については、「 <a href="#">AWS Key Management Service の条件キー</a> 」を参照してください。	文字列	 はい	 はい	 はい
condition .www.amaz on.com:ap p_id	[Login with Amazon (Amazon でログイン)] フェデレーションにロールへのアクセスを許可するための条件として指定された Amazon アプリケーション ID (またはサイト ID)。詳細については、次を参照してください。	文字列	 はい	 はい	 はい
id	結果の ID。	文字列	 いいえ	 はい	 はい

Criterion	説明	[Type] (タイプ)	アーカイブ ルール	結果の一覧 表示	アクセスプ レビューの 結果を一覧 表示
cchangeType	アクセスプレビューの結果と IAM Access Analyzer で識別された既存のアクセスとの比較に関するコンテキストを提供します。	文字列	 いいえ	 いいえ	 はい
既存の検索 Id	IAM Access Analyzer での結果の既存の ID。アクセスプレビューの既存の結果に対してのみ提供されます。	文字列	 いいえ	 いいえ	 はい
既存の検索 ステータス	アクセスプレビューの既存の調査結果に対してのみ提供される、検索結果の既存のステータス。	文字列	 いいえ	 いいえ	 はい

## AWS Identity and Access Management Access Analyzer のサービスにリンクされたロールの使用

AWS Identity and Access Management Access Analyzer は IAM [サービスにリンクされたロール](#)を使用します。サービスリンクロールは、IAM Access Analyzer に直接リンクされた特殊なタイプの IAM ロールです。サービスリンクロールは、IAM Access Analyzer によって事前定義されており、ユーザーの代わりに機能が他の AWS サービスを呼び出す必要のあるアクセス許可がすべて含まれています。

サービスリンクロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、IAM Access Analyzer の設定が簡単になります。IAM Access Analyzer は、サービスリンクロールのアクセス許可を定義します。特に定義されている場合を除き、IAM Access Analyzer のみがそのロールを引き受けることができます。定義されたアクセス許可には、信頼ポリシーとアクセス許可

ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティに添付することはできません。

サービスリンクロールをサポートする他のサービスについては、「[IAM と連携する AWS のサービス](#)」を参照して、[Service-Linked Role] (サービスリンクロール) 列が [Yes] (はい) になっているサービスを探してください。そのサービスに関するサービスにリンクされたロールのドキュメントを表示するには、リンクが設定されている [Yes] (はい) を選択します。

## AWS Identity and Access Management Access Analyzer のサービスにリンクされたロールの許可

AWS Identity and Access Management Access Analyzer は、AWSServiceRoleForAccessAnalyzer という名前のサービスリンクロールを使用します。これにより、Access Analyzer が外部アクセスのリソースメタデータを分析したり、アクティビティを分析して未使用のアクセスを特定したりできます。

サービスにリンクされたロール AWSServiceRoleForAccessAnalyzer は、以下のサービスを信頼してロールを引き受けます。

- `access-analyzer.amazonaws.com`

[AccessAnalyzerServiceRolePolicy](#) という名前のロールアクセス許可ポリシーを使用すると、IAM Access Analyzer は特定のリソースに対するアクションを完了できます。

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。詳細については、「IAM ユーザーガイド」の「[サービスにリンクされたロールの許可](#)」を参照してください。

## IAM Access Analyzer のサービスリンクロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。AWS Management Console または AWS API で Access Analyzer を有効にすると、IAM Access Analyzer によって、サービスリンクロールが作成されます。IAM Access Analyzer を有効にしたすべてのリージョンで、同じサービスにリンクされたロールが使用されます。外部アクセスと未使用のアクセスの両方の結果で、同じサービスリンクロールが使用されます。

**Note**

IAM Access Analyzer はリージョンに基づきます。IAM Access Analyzer は、各リージョンで個別に有効にする必要があります。

このサービスにリンクされたロールを削除すると、次回のアナライザーの作成時に、このロールが IAM Access Analyzer によって再作成されます。

Access Analyzer ユースケースでサービスにリンクされたロールを作成するには、IAM コンソールを使用します。AWS CLI または AWS API で、`access-analyzer.amazonaws.com` サービス名を使用してサービスリンクロールを作成します。詳細については、IAM ユーザーガイドの「[サービスリンクロールの作成](#)」を参照してください。このサービスリンクロールを削除する場合、この同じプロセスを使用して、もう一度ロールを作成できます。

## IAM Access Analyzer のサービスリンクロールの編集

IAM Access Analyzer では、`AWSServiceRoleForAccessAnalyzer` サービスリンクロールを編集できません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「[サービスにリンクされたロールの編集](#)」を参照してください。

## IAM Access Analyzer のサービスリンクロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングまたはメンテナンスされることがなくなります。ただし、手動で削除する前に、サービスにリンクされたロールのリソースをクリーンアップする必要があります。

**Note**

リソースを削除する際に、IAM Access Analyzer でロールが使用されている場合、削除は失敗することがあります。失敗した場合は、数分待ってから操作を再試行してください。

`AWSServiceRoleForAccessAnalyzer` によって使用されている IAM Access Analyzer リソースを削除する方法

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. [Access reports (アクセスレポート)] セクションの [Access analyzer (アクセスアナライザー)] で、[Analyzer (アナライザー)] を選択します。
3. [Analyzers (アナライザー)] テーブルのアナライザーのリストの上にある左上のチェックボックスを選択して、すべてのアナライザーを選択します。
4. [Delete] (削除) をクリックします。
5. アナライザーを削除することを確定するには、「**delete**」と入力し、[Delete (削除)] を選択します。

IAM を使用してサービスリンクロールを手動で削除するには

IAM コンソール、AWS CLI、または AWS API を使用して、サービスにリンクされたロール `AWSServiceRoleForAccessAnalyzer` を削除します。詳細については、「IAM ユーザーガイド」の「[サービスにリンクされたロールの削除](#)」を参照してください。

## IAM Access Analyzer サービスリンクロールをサポートするリージョン

IAM Access Analyzer では、このサービスが利用可能なすべてのリージョンで、サービスリンクロールの使用をサポートしています。詳細については、「[AWS リージョンとエンドポイント](#)」を参照してください。

## アクセスのプレビュー

外部エンティティと共有されているリソースの識別に加えて、AWS IAM Access Analyzer は、リソースのアクセス許可をデプロイする前に IAM Access Analyzer の結果のプレビューを表示するので、ポリシーの変更により、リソースに対する意図したパブリックアクセスとクロスアカウントアクセスのみが許可されていることを確認できます。これは、リソースへの意図した外部アクセスからスタートするのに役立ちます。

[Amazon S3 コンソール](#)で、AmazonS3 バケットへのパブリックアクセスおよびクロスアカウントアクセスをプレビューして検証できます。また、IAM Access Analyzer APIを使用して、リソースに提案されたへの許可を提供することにより、Amazon S3 バケット、AWS KMS キー、IAMロール、Amazon SQSキュー、および Secrets Manager のシークレットのパブリックアクセスとクロスアカウントアクセスをプレビューすることもできます。

### トピック

- [Amazon S3 コンソールでのアクセスのプレビュー](#)

- [IAM Access Analyzer API を使用したアクセスのプレビュー](#)

## Amazon S3 コンソールでのアクセスのプレビュー

Amazon S3 コンソールでバケットポリシーを完了すると、Amazon S3 バケットへの公開アクセスとクロスアカウントアクセスをプレビューできます。変更の保存を選択する前に、ポリシーの変更によって意図した外部アクセスのみが許可されていることを検証できます。このオプションの手順により、バケットの AWS Identity and Access Management Access Analyzer の検出結果のプレビューを実行できます。ポリシーの変更によって新しい検出結果が導入されるかどうか、または外部アクセスの既存の検出結果を解決するかどうかを検証できます。この検証ステップをスキップして、Amazon S3 バケットポリシーをいつでも保存できます。

バケットへの外部アクセスをプレビューするには、バケットのリージョンに、そのアカウントを信頼ゾーンとしてアクティブなアカウントアナライザが必要です。IAM Access Analyzer の使用およびアクセスのプレビューに必要な許可を持っている必要があります。IAM Access Analyzer の有効化および必要な許可の詳細については、「[IAM Access Analyzer の有効化](#)」を参照してください。

バケットポリシーを作成または編集する際に Amazon S3 バケットへのアクセスをプレビューするには

1. バケットポリシーの作成または編集が完了したら、ポリシーが有効な Amazon S3 バケットポリシーであることを確認します。ポリシー ARN はバケット ARN と一致し、[ポリシー要素](#)は有効である必要があります。
2. ポリシーの下の[Preview external access] (外部アクセスのプレビュー) で、アクティブなアカウントアナライザを選択し、[Preview] (プレビュー) を選択します。IAM Access Analyzer の検出結果のプレビューがバケットに対して生成されます。プレビューでは、表示された Amazon S3 バケットポリシーと、既存のバケットアクセス権限が分析されます。これには、バケットとアカウント BPA 設定、バケット ACL、バケットにアタッチされた Amazon S3 アクセスポイントとマルチリージョンアクセスポイント、およびそれらのポリシーと BPA 設定が含まれます。
3. アクセスプレビューが完了すると、IAM Access Analyzer の検出結果のプレビューが表示されます。各結果では、ポリシーを保存した後、バケットへのアクセス権を持つアカウントの外部にあるプリンシパルのインスタンスが報告されます。各結果を確認することで、バケットへのアクセスを検証できます。検出結果のヘッダーではアクセスの概要を知ることができます。また、検出結果を展開して[検出結果の詳細](#)を確認できます。検出結果のバッジにより、バケットポリシーを保存することでバケットへのアクセスがどのように変更されるかについてのコンテキストを知ることができます。たとえば、ポリシーの変更によって新しい調査結果が導入されるのか、外部アクセスの既存の調査結果を解決するのかが検証するのに役立ちます。



- a. 新規 — ポリシーによって導入される新しい外部アクセスの検出を示します。
  - b. 解決済み — ポリシーによって削除される既存の外部アクセスの結果を示します。
  - c. アーカイブ済み — 分析結果を意図したとおりにマークするタイミングを定義する分析装置のアーカイブルールに基づいて、自動的にアーカイブされる新しい外部アクセスの結果を示します。
  - d. 既存 — 変更されない外部アクセスの既存の検出を示します。
  - e. 公開 — 検索結果がリソースへのパブリックアクセス用である場合、公開バッジ、上記のいずれかのバッジに加えて。
4. 導入または削除する予定のない外部アクセスを特定した場合は、ポリシーを修正し、プレビューを再度実行して、意図した外部アクセスを達成します。公開というラベルの付いた検索結果がある場合は、変更を保存を選択する前に、公開アクセスを削除するようにポリシーを改訂することをお勧めします。アクセス権のプレビューはオプションの手順で、変更の保存にいつでもアクセスできます。

## IAM Access Analyzer API を使用したアクセスのプレビュー

[IAM Access Analyzer API](#) を使用して、Amazon S3 バケット、AWS KMS キー、IAM ロール、Amazon SQS キュー、および Secrets Manager シークレットの公開アクセスとクロスアカウントアクセスをプレビューできます。所有している既存のリソースまたはデプロイする新しいリソースに対して提案されたアクセス許可を提供することで、アクセスをプレビューできます。

リソースへの外部アクセスをプレビューするには、リソースのアカウントとリージョンに対してアクティブなアカウントアナライザが必要です。IAM Access Analyzer の使用およびアクセスのプレビューに必要な許可を持っている必要があります。IAM Access Analyzer の有効化および必要な許可の詳細については、「[IAM Access Analyzer の有効化](#)」を参照してください。

リソースのアクセスをプレビューするには、CreateAccessPreview オペレーションを実行し、アナライザー ARN とリソースのアクセス制御設定を提供します。このサービスは、アクセスプレビューの一意の ID を返します。これを使用して、GetAccessPreview オペレーションでアクセスプレビューのステータスをチェックできます。ステータスが Completed の場合、ListAccessPreviewFindings オペレーションを使用して、アクセスプレビュー用に生成された検出結果を取得できます。GetAccessPreview および ListAccessPreviewFindings オペレーションは、約 24 時間以内に作成されたアクセスプレビューと検出結果を取得します。

取得された各検出結果には、アクセスを説明する[検出結果の詳細](#)が含まれています。検出結果が Active、Archived、であるかまたは Resolved と、権限のデプロイ後であるか、および



changeType であるかを説明する検出結果のプレビューステータス。changeType では、アクセスプレビューが IAM Access Analyzer で識別された既存のアクセスと比較されるコンテキストを提供します。

- 新規 — 検出結果は、新しく導入されたアクセスのためのものです。
- 変更なし — プレビュー検出結果は変更されない既存の検出結果です。
- 変更あり — プレビュー結果は、ステータスが変更された既存の検出結果です。

status と changeType は、リソース構成によって既存のリソースへのアクセスがどのように変化するかを理解するのに役立ちます。changeType が Unchanged または変更された場合、検索結果には、IAM Access Analyzer の検出結果の既存の ID とステータスも含まれます。たとえば、プレビューステータス Changed と既存のステータス Resolved のある Active 検出結果は、提案されたアクセス許可の変更の結果として、リソースの既存の Active 結果が Resolved になることを示します。

ListAccessPreviews オペレーションを使用して、指定したアナライザーのアクセスプレビューのリストを取得します。このオペレーションは、約 1 時間以内に作成されたアクセスプレビューに関する情報を取得します。

一般に、アクセスプレビューが既存のリソースに対するものであり、設定オプションを指定しない場合、アクセスプレビューはデフォルトで既存のリソース設定を使用します。アクセスプレビューが新しいリソース用で、設定オプションを指定しない場合、アクセスプレビューはリソースタイプに応じてデフォルト値を使用します。各リソースタイプの設定ケースについては、次を参照してください。

## Amazon S3 バケットへのアクセスプレビュー機能

新しい Amazon S3 バケットまたは所有している既存の Amazon S3 バケットのアクセスプレビューを作成するには、バケットに添付された Amazon S3 バケットポリシー、バケット ACL、バケット BPA 設定、Amazon S3 アクセスポイント（複数リージョンアクセスポイントを含む）を指定して、バケット設定を提案します。

### Note

新しいバケットのアクセスプレビューを作成する前に、Amazon S3 [HeadBucket](#) オペレーションを使用して、指定されたバケットがすでに存在するかどうかを確認することをお勧めします。このオペレーションは、バケットが存在し、そのバケットにアクセスする許可があるかどうかを判断するのに役立ちます。

バケットポリシー — 設定が既存の Amazon S3 バケット用で、Amazon S3 バケットポリシーを指定しない場合、アクセスプレビューではバケットに添付された既存のポリシーが使用されます。アクセスプレビューが新しいリソースに対するもので、Amazon S3 バケットポリシーを指定しない場合、アクセスプレビューではポリシーのないバケットが想定されます。既存のバケットポリシーの削除を提案するには、空の文字列を指定します。サポートされるバケットポリシーの制限の詳細については、[バケットポリシーの例](#)を参照してください。。

バケットの ACL 付与機能 — バケットごとに最大 100 個の ACL 許可を提案できます。提案された許可設定が既存のバケット用である場合、アクセスプレビューでは、既存の許可の代わりに提案された許可設定のリストが使用されます。それ以外の場合、アクセスプレビューはバケットの既存の許可を使用します。

バケットアクセスポイント — この分析では、バケットごとに最大 100 個のアクセスポイント (マルチリージョンアクセスポイントを含む) がサポートされます。これには、バケットごとに提案できる新しいアクセスポイントが 10 個まで含まれます。提案された Amazon S3 アクセスポイント設定が既存のバケット用である場合、アクセスプレビューでは、既存のアクセスポイントの代わりに提案されたアクセスポイント設定が使用されます。ポリシーのないアクセスポイントを提案するには、アクセスポイントポリシーとして空の文字列を指定します。アクセスポイントポリシーの制限事項の詳細については、[アクセスポイントの制約と制限](#)を参照してください。。

公開アクセスブロック設定 — 提案された設定が既存の Amazon S3 バケット用であり、設定を指定しない場合、アクセスプレビューでは既存の設定が使用されます。提案された設定が新しいバケット用で、バケット BPA 設定を指定しない場合、アクセスプレビューは `false` を使用します。提案された設定が新しいアクセスポイントまたはマルチリージョンアクセスポイント用で、アクセスポイントの BPA 設定を指定しない場合、アクセスプレビューは `true` を使用します。

## AWS KMS キーへのアクセスプレビュー

新しい AWS KMS キーまたは所有している既存の AWS KMS キーのアクセスプレビューを作成するには、キーポリシーと AWS KMS 許可設定を指定して、AWS KMS キー設定を提案できます。

AWS KMS キーポリシー — 既存のキーの設定で、キーポリシーを指定しない場合、アクセスプレビューではキーに既存のポリシーが使用されます。アクセスプレビューが新しいリソースに対するもので、キーポリシーを指定しない場合、アクセスプレビューはデフォルトのキーポリシーを使用します。提案されたキーポリシーを空の文字列にすることはできません。

AWS KMS 許可 — 分析では、構成ごとに最大 100 の KMS 許可がサポートされます\*。\* 提案された許可設定が既存のキーに対するものである場合、アクセスプレビューでは、既存の許可の代わりに提案された許可設定のリストが使用されます。それ以外の場合、アクセスプレビューはキーの既存の許可を使用します。

## IAM ロールへのアクセスのプレビュー

新しい IAM ロールまたは所有している既存の IAM ロールのアクセスプレビューを作成するには、信頼ポリシーを指定して IAM ロール設定を提案できます。

ロールの信頼ポリシー — 設定が新しい IAM ロールの場合は、信頼ポリシーを指定する必要があります。所有している既存の IAM ロールの設定で、信頼ポリシーを提案しない場合、アクセスプレビューではロールに既存の信頼ポリシーが使用されます。提示された信頼ポリシーを空の文字列にすることはできません。

## Amazon SQS キューへのアクセスのプレビュー

新しい Amazon SQS キューまたはユーザーが所有する既存の Amazon SQS キューのアクセスプレビューを作成するには、キューの Amazon SQS ポリシーを指定して Amazon SQS キューの設定を提案します。

Amazon SQS キューポリシー — 設定が既存の Amazon SQS キュー用で、Amazon SQS ポリシーを指定しない場合、アクセスプレビューでは既存の Amazon SQS ポリシーがキューに使用されます。アクセスプレビューが新しいリソースに対するもので、ポリシーを指定しない場合、アクセスプレビューはポリシーなしの Amazon SQS キューを想定します。既存の Amazon SQS キューポリシーの削除を提案するには、Amazon SQS ポリシーに空の文字列を指定します。

## Secrets Manager のシークレットへのアクセスプレビュー

新しい Secrets Manager のシークレットまたは所有する既存の Secrets Manager のシークレットのアクセスプレビューを作成するには、シークレットポリシーとオプションの AWS KMS 暗号化キーを指定して、Secrets Manager のシークレット構成を提案できます。。

シークレットポリシー — 設定が既存のシークレット用であり、シークレットポリシーを指定しない場合、アクセスプレビューはシークレットに既存のポリシーを使用します。アクセスプレビューが新しいリソースに対するもので、ポリシーを指定しない場合、アクセスプレビューはポリシーのないシークレットを想定します。既存のポリシーの削除を提案するには、空の文字列を指定します。

AWS KMS暗号化キー — 提案された構成が新しいシークレット用であり、AWS KMS キー ID を指定しない場合、アクセスプレビューは AWS アカウントのデフォルトの KMS キーを使用します。AWS KMS キー ID に空の文字列を指定すると、アクセスプレビューは AWS アカウントのデフォルトの KMS キーを使用します。

## ポリシーを検証するためのチェック

IAM Access Analyzer には、IAM ポリシーをエンティティにアタッチする前に検証するのに役立つポリシーチェックが備わっています。これには、[ポリシーの文法](#)や [AWS ベストプラクティス](#)に照らしてポリシーを検証するためにポリシー検証によって提供されるポリシーチェックが含まれます。セキュリティ警告、エラー、一般的な警告、ポリシーの提案を含むポリシー検証チェックの結果を表示できます。

カスタムポリシーチェックを使用して、セキュリティ標準に基づいて新しいアクセスをチェックできます。料金は、新しいアクセスに対する各チェックに関連付けられます。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

### トピック

- [IAM Access Analyzer ポリシーの検証](#)
- [IAM Access Analyzer カスタムポリシーチェック](#)

## IAM Access Analyzer ポリシーの検証

AWS Identity and Access Management Access Analyzer ポリシー検証を使用してポリシーを検証できます。IAM コンソールの AWS CLI、AWS API、または JSON ポリシーエディタを使用して、ポリシーを作成または編集できます。IAM Access Analyzer は、IAM [ポリシーの文法](#)および [AWS ベストプラクティス](#)に照らしてポリシーを検証します。セキュリティ警告、エラー、一般的な警告、ポリシーの提案を含むポリシー検証チェックの結果を表示できます。これらの結果により、機能的でセキュリティのベストプラクティスに準拠したポリシーの作成に役立つ実用的な推奨事項が示されます。IAM Access Analyzer によって実行される基本ポリシーチェックのリストを表示するには、「[Access Analyzer ポリシーチェックリファレンス](#)」を参照してください。

### IAM (コンソール) でのポリシーの検証

IAM コンソールで管理ポリシーを作成または編集するときに、IAM Access Analyzer ポリシー検証によって生成された検出結果を表示できます。また、インラインユーザーまたはロールポリシーのこれらの結果を表示することもできます。IAM Access Analyzer では、インライングループポリシーのこれらの検出結果は生成されません。

IAM JSON ポリシーのポリシーチェックによって生成された結果を表示するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. 以下のいずれかの方法を使用して、ポリシーの作成または編集を開始します。
  - a. 新しい管理ポリシーを作成するには、[Policies] (ポリシー) ページを開き、新しいポリシーを作成します。詳細については、「[JSON エディターを使用したポリシーの作成](#)」を参照してください。
  - b. 既存のカスタマー管理ポリシーのポリシーチェックを表示するには、[ポリシー] ページで、ポリシーの名前を選択し、[編集] を選択します。詳細については、「[カスタマー管理ポリシーの編集 \(コンソール\)](#)」を参照してください。
  - c. ユーザーまたはロールのインラインポリシーについてのポリシーチェックを表示するには、[ユーザー] または [ロール] ページに移動して、ユーザーまたはロールの名前を選択し、[アクセス許可] タブでポリシーの名前を選択して、[編集] を選択します。詳細については、「[カスタマー管理ポリシーの編集 \(コンソール\)](#)」を参照してください。
3. ポリシーエディターで、[JSON] タブを選択します。
4. ポリシーの下にあるポリシー検証ペインで、次の 1 つ以上のタブを選択します。タブ名は、ポリシーの各結果タイプの番号も示します。
  - [Security] (セキュリティ) — アクセスが過剰に許容されているために、AWS がセキュリティリスクを考慮するアクセスをポリシーで許可している場合は、警告を表示します。
  - [Errors] (エラー) - ポリシーにポリシーが機能しない行が含まれている場合は、エラーを表示します。
  - [警告] - ポリシーがベストプラクティスに準拠していないが、セキュリティリスクではない場合は、警告を表示します。
  - [Suggestions] (提案) — AWS がポリシーのアクセス許可に影響しない改善を推奨する場合は、提案を表示します。
5. IAM Access Analyzer ポリシーチェックによって提供される結果の詳細を確認します。各結果には、報告された問題の場所が示されます。問題の原因とその解決方法の詳細については、結果の横にある [Learn more] (詳細) リンクを選択してください。また、[Access Analyzer ポリシーチェック](#) リファレンスページで、各結果に関連付けられているポリシーチェックを検索することもできます。
6. オプションです。既存のポリシーを編集する場合、カスタムポリシーチェックを実行して、更新したポリシーで新しいアクセス権限が付与されるかどうかを、既存のバージョンとの比較で判別できます。ポリシーの下にあるポリシー検証ペインで、[新しいアクセス権限の確認] タブを選択し、[ポリシーの確認] を選択します。変更したアクセス許可によって新しいアクセス権限が付与されると、ポリシー検証ペインでそのステートメントが強調表示されます。新しいアクセス権限を付与する予定がない場合は、ポリシーステートメントを更新し、新しいアクセスが検出されな

くなるまで [ポリシーの確認] を選択します。詳細については、「[IAM Access Analyzer カスタムポリシーチェック](#)」を参照してください。

**Note**

料金は、新しいアクセスに対する各チェックに関連付けられます。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

7. ポリシーを更新して結果を解決します。

**Important**

新しいポリシーや編集したポリシーを本番ワークフローに実装する前に、徹底的にテストします。

8. 完了したら、[Next] を選択します。[Policy validator](#) は、IAM Access Analyzer によって報告されない構文エラーを報告します。

**Note**

いつでも [ビジュアル] タブと [JSON] タブを切り替えることができます。ただし、[ビジュアル] タブで変更を加えるか [次へ] を選択した場合、IAM はポリシーを再構成し、ビジュアルエディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

9. 新しいポリシーの場合、[確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。[このポリシーで定義されている許可] を確認して、ポリシーによって付与されるアクセス許可を確認します。次に、[Create policy] (ポリシーの作成) を選択して作業を保存します。

既存のポリシーの場合、[確認と保存] ページで [このポリシーで定義されている許可] を確認して、ポリシーによって付与されるアクセス許可を確認します。[この新しいバージョンをデフォルトとして設定する。] チェックボックスを選択して、更新したバージョンをポリシーのデフォルトバージョンとして保存します。次に、[変更の保存] を選択して作業を保存します。



## IAM Access Analyzer (AWS CLI または AWS API) を使用したポリシーの検証

AWS Command Line Interface (AWS CLI) から、IAM Access Analyzer ポリシー検証によって生成された検出結果を表示できます。

IAM Access Analyzer ポリシー検証 (AWS CLI または AWS API) によって生成された検出結果を表示する方法

以下のいずれかを使用します。

- AWS CLI: [aws accessanalyzer ポリシーの検証](#)
- AWS API: [ポリシーの検証](#)

## Access Analyzer ポリシーチェックリファレンス

AWS Identity and Access Management Access Analyzer ポリシー検証を使用してポリシーを検証できます。IAM コンソールの AWS CLI、AWS API、または JSON ポリシーエディタを使用して、ポリシーを作成または編集できます。IAM Access Analyzer は、IAM [ポリシーの文法](#) および [AWS ベストプラクティス](#) に照らしてポリシーを検証します。セキュリティ警告、エラー、一般的な警告、ポリシーの提案を含むポリシー検証チェックの結果を表示できます。これらの結果により、機能的でセキュリティのベストプラクティスに準拠したポリシーの作成に役立つ実用的な推奨事項が示されます。IAM Access Analyzer が提供する基本ポリシーチェックのリストを以下に示します。ポリシー検証チェックの実行に関連する追加料金は発生しません。ポリシー検証を使用したポリシーの検証の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

エラー — ARN アカウントは許可されていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
ARN account not allowed: The service {{service}} does not support specifying an account ID in the resource ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The service {{service}} does not support specifying an account ID in the resource ARN."
```

エラーの解決

リソース ARN からアカウント ID を削除します。一部の AWS サービスのリソース ARN は、アカウント ID の指定をサポートしていません。

たとえば、Amazon S3 では、アカウント ID をバケット ARN の名前空間としてサポートしていません。Amazon S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。Amazon S3 で利用可能なすべてのリソースタイプを表示するには、「[サービス認証リファレンス](#)」の「Amazon S3 で定義されたリソースタイプ」を参照してください。

## 関連用語

- [ポリシーリソース](#)
- [アカウント ID](#)
- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

エラー — ARN リージョンは許可されていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
ARN Region not allowed: The service {{service}} does not support specifying a Region in the resource ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The service {{service}} does not support specifying a Region in the resource ARN."
```

## エラーの解決

リソース ARN からリージョンを削除します。一部の AWS サービスのリソース ARN は、リージョンの指定をサポートしていません。

たとえば、IAM はグローバルサービスです。IAM リソース ARN のリージョン部分は常に空白のままです。AWS アカウントが今日あるように、IAM リソースはグローバルです。たとえば、IAM ユーザーとしてサインインすると、あらゆるリージョンの AWS サービスにアクセスできます。



- [ポリシーリソース](#)
- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

## エラー — データ型の不一致

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Data type mismatch: The text does not match the expected JSON data type {{data_type}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The text does not match the expected JSON data type {{data_type}}."
```

## エラーの解決

サポートされているデータ型を使用するようにテキストを更新します。

たとえば、Version グローバル条件キーには String データ型が必要です。日付または整数を指定すると、データ型が一致しません。

## 関連用語

- [グローバル条件キー](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

## エラー - 大文字と小文字が違う重複したキー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Duplicate keys with different case: The condition key {{key}} appears more than once with different capitalization in the same condition block. Remove the duplicate condition keys.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key {{key}} appears more than once with different capitalization in the same condition block. Remove the duplicate condition keys."
```

## エラーの解決

同じ条件ブロック内の同様の条件キーを確認し、すべてのインスタンスで同じ大文字を使用します。

条件ブロックは、ポリシーステートメントの Condition 要素内のテキストです。条件キーの名前は、大文字小文字を区別しません。条件キーの値の大文字と小文字の区別は、使用する条件演算子によって異なります。条件キーでの大文字と小文字の区別に関する詳細については、[IAM JSON ポリシー要素 Condition](#) を参照してください。

## 関連用語

- [条件](#)
- [条件ブロック](#)
- [グローバル条件キー](#)
- [AWS サービス条件キー](#)

## エラー — 無効なアクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid action: The action {{action}} does not exist. Did you mean {{valid_action}}?
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The action {{action}} does not exist. Did you mean {{valid_action}}?"
```

## エラーの解決

指定されたアクションが無効です。これは、サービスプレフィックスまたはアクション名を誤って入力した場合に発生します。一般的な問題については、ポリシーチェックによって推奨されるアクションが返されます。

## 関連用語

- [ポリシーアクション](#)
- [AWS のサービスアクション](#)

このエラーが発生した AWS 管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

次の AWS 管理ポリシーには、ポリシーステートメントに無効なアクションが含まれています。無効なアクションは、ポリシーで付与されるアクセス権限には影響しません。AWS 管理ポリシーを参照として使用して、管理ポリシーを作成する場合、AWS がポリシーから無効なアクションを削除することをお勧めします。

- [AmazonEMRFullAccessPolicy\\_v2](#)
- [CloudWatchSyntheticsFullAccess](#)

エラー — 無効な ARN アカウント

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid ARN account: The resource ARN account ID {{account}} is not valid. Provide a 12-digit account ID.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The resource ARN account ID {{account}} is not valid. Provide a 12-digit account ID."
```

エラーの解決

リソース ARN のアカウント ID を更新します。アカウント ID は 12 桁の整数です。アカウント ID を表示する方法については、「[AWS アカウント ID の検索](#)」を参照してください。

関連用語

- [ポリシーリソース](#)
- [アカウント ID](#)

- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

### エラー — 無効な ARN プレフィックス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid ARN prefix: Add the required prefix (arn) to the resource ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add the required prefix (arn) to the resource ARN."
```

### エラーの解決

AWS リソースARNには、必要な arn: プレフィックスを含める必要があります。

### 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

### エラー — 無効な ARN リージョン

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid ARN Region: The Region {{region}} is not valid for this resource. Update the resource ARN to include a supported Region.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Region {{region}} is not valid for this resource. Update the resource ARN to include a supported Region."
```

### エラーの解決

リソースタイプは、指定されたリージョンではサポートされていません。各リージョンでサポートされている AWS サービスの表については、[リージョン表リージョン表](#)を参照してください。

## 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [リージョン名とコード](#)

## エラー — 無効な ARN リソース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid ARN resource: Resource ARN does not match the expected ARN format. Update the resource portion of the ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Resource ARN does not match the expected ARN format. Update the resource portion of the ARN."
```

## エラーの解決

リソース ARN は、既知のリソースタイプの仕様と一致する必要があります。サービスに期待される ARN 形式を表示するには、「[AWS サービスのアクション、リソース、および条件キー](#)」を参照してください。サービスの名前を選択して、そのリソースタイプと ARN 形式を表示します。

## 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

## エラー — 無効な ARN サービスのケース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid ARN service case: Update the service name ${service} in the resource ARN to use all lowercase letters.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Update the service name ${service} in the resource ARN to use all lowercase letters."
```

## エラーの解決

リソース ARN 内のサービスは、サービスプレフィックスの仕様 (大文字化を含む) と一致する必要があります。サービスのプレフィックスについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。サービスの名前を選択し、最初の文でそのプレフィックスを見つけてください。

## 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [ARN 形式の AWS サービスリソース](#)

## エラー - 無効な条件データ型

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid condition data type: The condition value data types do not match. Use condition values of the same JSON data type.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition value data types do not match. Use condition values of the same JSON data type."
```

## エラーの解決

条件キーと値のペアの値は、条件キーと条件演算子のデータ型と一致する必要があります。サービスの条件キーのデータ型を表示するには、「[AWS サービスのアクション、リソース、および条件](#)

[キー](#)」を参照してください。サービスの名前を選択すると、そのサービスの条件キーが表示されます。

たとえば、[CurrentTime](#) グローバル条件キーは Date 条件演算子を使用します。条件ブロックの値に文字列または整数を指定すると、データ型は一致しません。

#### 関連用語

- [条件](#)
- [条件ブロック](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [グローバル条件キー](#)
- [AWS サービス条件キー](#)

#### エラー - 無効な条件キーの形式

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid condition key format: The condition key format is not valid. Use the format service:keyname.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key format is not valid. Use the format service:keyname."
```

#### エラーの解決

条件のキーと値のペアのキーは、サービスの仕様と一致する必要があります。サービスの条件キーを表示するには、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。サービスの名前を選択すると、そのサービスの条件キーが表示されます。

#### 関連用語

- [条件](#)
- [グローバル条件キー](#)
- [AWS サービス条件キー](#)

## エラー - 無効な条件の複数ブール値

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid condition multiple Boolean: The condition key does not support multiple Boolean values. Use a single Boolean value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key does not support multiple Boolean values. Use a single Boolean value."
```

## エラーの解決

条件キーと値のペアのキーは、単一のブール値を想定しています。複数のブール値を指定すると、条件一致によって期待どおりの結果が返されないことがあります。

サービスの条件キーを表示するには、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。サービスの名前を選択すると、そのサービスの条件キーが表示されます。

- [条件](#)
- [グローバル条件キー](#)
- [AWS サービス条件キー](#)

## エラー - 無効な条件演算子

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid condition operator: The condition operator {{operator}} is not valid. Use a valid condition operator.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition operator {{operator}} is not valid. Use a valid condition operator."
```



## エラーの解決

条件を更新して、サポートされている条件演算子を使用します。

### 関連用語

- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

## エラー — 無効な効果

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid effect: The effect {{effect}} is not valid. Use Allow or Deny.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The effect {{effect}} is not valid. Use Allow or Deny."
```

## エラーの解決

Effect 要素を更新して、有効な効果を使用します。Effect の有効値は、**Allow** と **Deny** です。

### 関連用語

- [Effect エレメント](#)
- [JSON ポリシー概要](#)

## エラー — 無効なグローバル条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid global condition key: The condition key {{key}} does not exist. Use a valid condition key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key {{key}} does not exist. Use a valid condition key."
```

## エラーの解決

条件のキーバリューのペアの条件キーを更新して、サポートされているグローバル条件キーを使用します。

グローバル条件キーは、aws: プレフィックスが付いた条件キーです。AWS のサービスは、グローバル条件キーをサポートするか、サービスプレフィックスを含むサービス固有のキーを提供できます。たとえば、IAM 条件キーには iam: プレフィックスが含まれます。詳細については、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照し、キーを表示するサービスを選択します。

## 関連用語

- [グローバル条件キー](#)

## エラー — 無効なパーティション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid partition: The resource ARN for the service {{service}} does not support the partition {{partition}}. Use the supported values: {{partitions}}
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The resource ARN for the service {{service}} does not support the partition {{partition}}. Use the supported values: {{partitions}}"
```

## エラーの解決

リソース ARN を更新して、サポートされているパーティションを含めます。サポートされているパーティションを含めると、サービスまたはリソースは含めたパーティションをサポートしていない可能性があります。

パーティションは AWS リージョンのグループです。各 AWS アカウントのスコープは 1 つのパーティションです。クラシックリージョンでは、aws パーティションを使用します。中国リージョンでは、aws-cn を使用します。

## 関連用語

- [Amazon リソースネーム \(ARN\) - パーティション](#)

## エラー — 無効なポリシー要素

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid policy element: The policy element {{element}} is not valid.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy element {{element}} is not valid."
```

## エラーの解決

サポートされている JSON ポリシーの要素のみを含めるようにポリシーを更新します。

## 関連用語

- [JSON ポリシーの要素](#)

## エラー - 無効なプリンシパル形式

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid principal format: The Principal element contents are not valid. Specify a key-value pair in the Principal element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Principal element contents are not valid. Specify a key-value pair in the Principal element."
```

## エラーの解決

プリンシパルを更新して、サポートされているキーと値のペアの形式を使用します。

リソースベースのポリシーでは、プリンシパルを指定できますが、アイデンティティベースのポリシーでは指定できません。

たとえば、AWS アカウント内のすべてのユーザーのアクセスを定義するには、ポリシーで次のプリンシパルを使用します。

```
"Principal": { "AWS": "123456789012" }
```

## 関連用語

- [JSON ポリシーの要素: プリンシパル](#)
- [アイデンティティベースおよびリソースベースのポリシー](#)

## エラー — 無効なプリンシパルキー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid principal key: The principal key {{principal-key}} is not valid.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The principal key {{principal-key}} is not valid."
```

## エラーの解決

プリンシパルキーバリューのペアのキーを更新して、サポートされているプリンシパルキーを使用します。サポートされているプリンシパルキーは次のとおりです。

- AWS
- CanonicalUser
- Federated
- サービス

## 関連用語

- [プリンシパル要素](#)

## エラー — 無効なリージョン

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid Region: The Region {{region}} is not valid. Update the condition value to a supported Region.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Region {{region}} is not valid. Update the condition value to a supported Region."
```

### エラーの解決

条件キーと値のペアの値を更新し、サポートされているリージョンを含めます。各リージョンでサポートされている AWS サービスの表については、[リージョン表リージョン表](#)を参照してください。

### 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [リージョン名とコード](#)

## エラー — 無効なサービス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid service: The service {{service}} does not exist. Use a valid service name.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The service {{service}} does not exist. Use a valid service name."
```

### エラーの解決

アクションまたは条件キーのサービスプレフィックスは、サービスプレフィックスの仕様 (大文字化を含む) と一致する必要があります。サービスのプレフィックスを表示するには、「[AWS サービス](#)

[のアクション、リソース、条件キー](#)」を参照してください。サービスの名前を選択し、最初の文でそのプレフィックスを見つけます。

## 関連用語

- [既知のサービスとそのアクション、リソース、条件キー](#)

### エラー — 無効なサービス条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid service condition key: The condition key {{key}} does not exist in the service {{service}}. Use a valid condition key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key {{key}} does not exist in the service {{service}}. Use a valid condition key."
```

## エラーの解決

サービスの既知の条件キーを使用するには、条件キーと値のペアのキーを更新します。グローバル条件キーは、aws プレフィックスを持つ条件キーです。AWS のサービスは、サービスプレフィックスを含むサービス固有のキーを提供できます。サービスのプレフィックスを表示するには、「[AWS サービスのアクション、リソース、条件キー](#)」を参照してください。

## 関連用語

- [グローバル条件キー](#)
- [既知のサービスとそのアクション、リソース、条件キー](#)

### エラー — 無効なサービスが動作しています

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid service in action: The service {{service}} specified in the action does not exist. Did you mean {{service2}}?
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The service {{service}} specified in the action does not exist. Did you mean {{service2}}?"
```

## エラーの解決

アクションのサービスプレフィクスは、サービスプレフィクスの仕様 (大文字化を含む) と一致する必要があります。サービスのプレフィックスを表示するには、「[AWS サービスのアクション、リソース、条件キー](#)」を参照してください。サービスの名前を選択し、最初の文でそのプレフィックスを見つけます。

## 関連用語

- [アクション要素](#)
- [既知のサービスとそのアクション](#)

## エラー — 無効な演算子の変数

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid variable for operator: Policy variables can only be used with String and ARN operators.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Policy variables can only be used with String and ARN operators."
```

## エラーの解決

Resource 要素のポリシー変数、および Condition 要素の文字列比較を活用できます。条件では、文字列演算子または ARN 演算子を使用するときに変数がサポートされます。文字列演算子には StringEquals、StringLike、StringNotLike が含まれます。ARN 演算子には ArnEquals と ArnLike が含まれます。ポリシー変数は、数値、日付、ブール値、バイナリ、IPアドレス、Null 演算子などの他の演算子では使用できません。

## 関連用語

- [条件要素でのポリシー変数の使用](#)
- [条件の要素](#)

## エラー — 無効なバージョン

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid version: The version ${version} is not valid. Use one of the following versions: ${versions}
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The version ${version} is not valid. Use one of the following versions: ${versions}"
```

## エラーの解決

Version ポリシー要素は、AWS このポリシーを処理するために使用される言語構文ルールを指定します。ポリシーの使用可能なすべての機能を使用するには、すべてのポリシーの Statement 要素の前に最新の Version 要素を含めます。

```
"Version": "2012-10-17"
```

## 関連用語

- [バージョン要素](#)

## エラー - Json 構文エラー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Json syntax error: Fix the JSON syntax error at index {{index}} line {{line}} column {{column}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。



```
"findingDetails": "Fix the JSON syntax error at index {{index}} line {{line}} column {{column}}."
```

## エラーの解決

ポリシーに構文エラーがあります。JSON 構文を確認してください。

### 関連用語

- [JSON バリデータ](#)
- [IAM JSON ポリシーエレメントのリファレンス](#)
- [JSON ポリシー概要](#)

## エラー - Json 構文エラー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Json syntax error: Fix the JSON syntax error.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Fix the JSON syntax error."
```

## エラーの解決

ポリシーに構文エラーがあります。JSON 構文を確認してください。

### 関連用語

- [JSON バリデータ](#)
- [IAM JSON ポリシーエレメントのリファレンス](#)
- [JSON ポリシー概要](#)

## エラー — アクションが見つかりません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing action: Add an Action or NotAction element to the policy statement.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add an Action or NotAction element to the policy statement."
```

## エラーの解決

AWSJSON ポリシーには Action または NotAction 要素を含める必要があります。

## 関連用語

- [アクション要素](#)
- [NotAction エレメント](#)
- [JSON ポリシー概要](#)

## エラー — ARN フィールドがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing ARN field: Resource ARNs must include at least {{fields}} fields in the following structure: arn:partition:service:region:account:resource
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Resource ARNs must include at least {{fields}} fields in the following structure: arn:partition:service:region:account:resource"
```

## エラーの解決

リソース ARN のすべてのフィールドは、既知のリソースタイプの仕様と一致する必要があります。サービスに期待される ARN 形式を表示するには、「[AWS サービスのアクション、リソース、および条件キー](#)」を参照してください。サービスの名前を選択して、そのリソースタイプと ARN 形式を表示します。

## 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)

- [ARN 形式の AWS サービスリソース](#)

### エラー — ARN リージョンがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing ARN Region: Add a Region to the {{service}} resource ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a Region to the {{service}} resource ARN."
```

### エラーの解決

ほとんどの AWS サービスのリソース ARN では、リージョンを指定する必要があります。各リージョンでサポートされている AWS サービスの表については、[リージョン表リージョン表](#)を参照してください。

### 関連用語

- [ポリシーリソース](#)
- [リソース ARN](#)
- [リージョン名とコード](#)

### エラー — 効果がありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing effect: Add an Effect element to the policy statement with a value of Allow or Deny.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add an Effect element to the policy statement with a value of Allow or Deny."
```

## エラーの解決

AWS JSON ポリシーには Effect および **Allow** の値を持つ **Deny** 要素を含める必要があります。

### 関連用語

- [Effect エレメント](#)
- [JSON ポリシー概要](#)

## エラー — プリンシパルが見つかりません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing principal: Add a Principal element to the policy statement.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a Principal element to the policy statement."
```

## エラーの解決

リソースベースのポリシーには Principal 要素を含める必要があります。

たとえば、AWS アカウント内のすべてのユーザーのアクセスを定義するには、ポリシーで次のプリンシパルを使用します。

```
"Principal": { "AWS": "123456789012" }
```

### 関連用語

- [プリンシパル要素](#)
- [アイデンティティベースおよびリソースベースのポリシー](#)

## エラー - 修飾子がありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing qualifier: The request context key ${key} has multiple values. Use the
ForAllValues or ForAnyValue condition key qualifiers in your policy.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The request context key ${key} has multiple values. Use the
ForAllValues or ForAnyValue condition key qualifiers in your policy."
```

## エラーの解決

Condition 要素に、条件演算子 (等しい、より小さい、など) を使用して、ポリシーの条件キーと値をリクエストコンテキストのキーと値に一致させる式を構築します。1つの条件キーに複数の値が含まれるリクエストの場合、配列のように条件を角括弧で囲む必要があります ("Key2":["Value2A", "Value2B"])。また、ForAllValues または ForAnyValue set 演算子を StringLike 条件演算子とともに使用する必要もあります。これらの限定子によって条件演算子にセット演算機能が追加されるため、複数のリクエスト値を複数の条件値と照合できます。

## 関連用語

- [複数值のコンテキストキー](#)
- [条件の要素](#)

このエラーが発生した AWS 管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

次の AWS 管理ポリシーでは、ポリシーステートメントに条件キーの修飾子がありません。AWS 管理ポリシーを参照として使用してカスタマー管理ポリシーを作成する場合、AWS は ForAllValues または ForAnyValue 条件キー修飾子を Condition 要素に追加することをお勧めします。

- [AWSGlueConsoleSageMakerNotebookFullAccess](#)

## エラー — リソースがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing resource: Add a Resource or NotResource element to the policy statement.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a Resource or NotResource element to the policy statement."
```

## エラーの解決

ロール信頼ポリシーを除くすべてのポリシーには、Resource または NotResource 要素を含める必要があります。

## 関連用語

- [リソースの要素](#)
- [非リソースの要素](#)
- [アイデンティティベースおよびリソースベースのポリシー](#)
- [JSON ポリシー概要](#)

## エラー — ステートメントが見つかりません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing statement: Add a statement to the policy
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a statement to the policy"
```

## エラーの解決

JSON ポリシーには、ステートメントを含める必要があります。

## 関連用語

- [JSON ポリシーの要素](#)

## エラー — 存在する場合はNULL

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Null with if exists: The Null condition operator cannot be used with the IfExists suffix. Update the operator or the suffix.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Null condition operator cannot be used with the IfExists suffix. Update the operator or the suffix."
```

## エラーの解決

IfExists 条件演算子を除く任意の条件演算子名の末尾に Null を追加できます。条件キーが承認時に存在するかどうかを確認するには、Null 条件演算子を使用します。...ifExists を使用して、「ポリシーキーがリクエストのコンテキストで存在する場合、ポリシーで指定されたとおりにキーを処理します。キーが存在しない場合、条件要素は true と評価されます。」

## 関連用語

- [...条件演算子が存在する場合](#)
- [Null 条件演算子](#)
- [条件の要素](#)

## エラー — SCP 構文エラー処理のワイルドカード

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error action wildcard: SCP actions can include wildcards (*) only at the end of a string. Update {{action}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCP actions can include wildcards (*) only at the end of a string. Update {{action}}."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) では、Action または NotAction 要素での値の指定をサポートします。ただし、これらの値には、文字列の末尾にのみワイルドカード (\*) を含めることができます。つまり、iam:Get\* は指定できますが、iam:\*role は指定できません。

複数のアクションを指定するには、AWSは、個別にリストすることをお勧めします。

### 関連用語

- [SCP アクションおよび NotAction 要素](#)
- [SCP 評価](#)
- [AWS Organizations サービスコントロールポリシー](#)
- [IAM JSON ポリシーエレメント: アクション](#)

## エラー — SCP 構文エラー 条件を許可

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error allow condition: SCPs do not support the Condition element with effect Allow. Update the element Condition or the effect.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCPs do not support the Condition element with effect Allow. Update the element Condition or the effect."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) では、Condition を使用する場合にのみ "Effect": "Deny" 要素に値を指定することをサポートします。

単一のアクションのみを許可するために、...NotEquals バージョンの条件演算子を使用して指定した条件以外のすべてへのアクセスを拒否できます。これは、演算子によって行われた比較を無効にします。

### 関連用語

- [SCP 条件の要素](#)



- [SCP 評価](#)
- [AWS Organizations サービスコントロールポリシー](#)
- [ポリシーの例：リクエストされたリージョンに基づいて AWS へのアクセスを拒否する](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## エラー — SCP 構文エラー NotAction を許可

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error allow NotAction: SCPs do not support NotAction with effect Allow.
Update the element NotAction or the effect.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCPs do not support NotAction with effect Allow. Update the element
NotAction or the effect."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) は、NotAction での "Effect": "Allow" 要素の使用をサポートしていません。

アクションのリストを許可するか、またはリストされていないすべてのアクションを拒否するには、ロジックを書き直す必要があります。

## 関連用語

- [SCP アクションおよび NotAction 要素](#)
- [SCP 評価](#)
- [AWS Organizations サービスコントロールポリシー](#)
- [IAM JSON ポリシーエレメント: アクション](#)

## エラー — SCP 構文エラー リソースを許可

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error allow resource: SCPs do not support Resource with effect Allow. Update the element Resource or the effect.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCPs do not support Resource with effect Allow. Update the element Resource or the effect."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) では、Resource を使用する場合にのみ "Effect": "Deny" 要素に値を指定することをサポートします。

すべてのリソースを許可するか、リストされているすべてのリソースを拒否するには、ロジックを書き直す必要があります。

## 関連用語

- [SCP リソースの要素](#)
- [SCP 評価](#)
- [AWS Organizations サービスコントロールポリシー](#)
- [IAM JSON ポリシーエレメント: リソース](#)

## エラー — SCP 構文エラー NotResource

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error NotResource: SCPs do not support the NotResource element. Update the policy to use Resource instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCPs do not support the NotResource element. Update the policy to use Resource instead."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) は、NotResource 要素をサポートしていません。

すべてのリソースを許可するか、リストされているすべてのリソースを拒否するには、ロジックを書き直す必要があります。

#### 関連用語

- [SCP リソースの要素](#)
- [SCP 評価](#)
- [AWS Organizations サービスコントロールポリシー](#)
- [IAM JSON ポリシーエレメント: リソース](#)

#### エラー — SCP 構文エラー プリンシパル

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
SCP syntax error principal: SCPs do not support specifying principals. Remove the Principal or NotPrincipal element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "SCPs do not support specifying principals. Remove the Principal or NotPrincipal element."
```

#### エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) は、Principal または NotPrincipal 要素をサポートしていません。

aws:PrincipalArn 要素の Condition グローバル条件キーを使用して Amazon リソースネーム (ARN) を指定できます。

#### 関連用語

- [SCP 構文](#)
- [プリンシパルのグローバル条件キー](#)

## エラー — 一意の Sid が必要です

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unique Sids required: Duplicate statement IDs are not supported for this policy type.
Update the Sid value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Duplicate statement IDs are not supported for this policy type.
Update the Sid value."
```

## エラーの解決

一部のポリシータイプでは、ステートメント ID は一意である必要があります。Sid (ステートメント ID) 要素を使用すると、ポリシーステートメントに指定するオプションの識別子を入力できません。SID 要素を使用して、ステートメント配列内の各ステートメントにステートメントID値を割り当てることができます。SQS や SNS などの エレメントを特定するサービスでは、Sid 値はポリシードキュメント ID のサブ ID に過ぎません。たとえば、IAM では、Sid 値は JSON ポリシー内で固有のものである必要があります。

## 関連用語

- [IAM JSON ポリシーエレメント: Sid](#)

## エラー — ポリシーでサポートされていないアクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
attached to the resource type {{resourceType}}."
```

## エラーの解決

一部のアクションは、異なるリソースタイプに添付された、リソースベースのポリシーの Action エlement ではサポートされていません。例えば、AWS Key Management Service アクションは、Amazon S3 バケットのポリシーではサポートされていません。リソースベースのポリシーに添付のリソースタイプでサポートされているアクションを指定してください。

### 関連用語

- [JSON ポリシーエレメント: Action](#)

## エラー - サポートされていない要素の組み合わせ

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported element combination: The policy elements ${element1} and ${element2} can not be used in the same statement. Remove one of these elements.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy elements ${element1} and ${element2} can not be used in the same statement. Remove one of these elements."
```

## エラーの解決

JSON ポリシーの要素の組み合わせによっては、一緒に使用できないものもあります。たとえば、Action と NotAction を同じポリシーステートメントで使用することはできません。相互に排他的な他のペアには、Principal/NotPrincipal と Resource/NotResource が含まれます。

### 関連用語

- [IAM JSON ポリシーエレメントのリファレンス](#)
- [JSON ポリシー概要](#)

## エラー — サポートされていないグローバル条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported global condition key: The condition key aws:ARN is not supported. Use aws:PrincipalArn or aws:SourceArn instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key aws:ARN is not supported. Use aws:PrincipalArn or aws:SourceArn instead."
```

## エラーの解決

AWS は、指定されたグローバル条件キーの使用をサポートしていません。ユースケースに応じて、aws:PrincipalArn または aws:SourceArn グローバル条件キーを使用できます。たとえば、aws:ARN の代わりに aws:PrincipalArn を使用して、リクエストを行ったプリンシパルの Amazon リソースネーム (ARN) をポリシーで指定した ARN と比較します。代わりに、aws:SourceArn グローバル条件キーを使用して、サービス間リクエストを行っているリソースの Amazon リソースネーム (ARN) を、ポリシーで指定した ARN と比較します。

## 関連用語

- [AWS グローバル条件コンテキストキー](#)

## エラー — サポートされていないプリンシパル

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported principal: The policy type ${policy_type} does not support the Principal element. Remove the Principal element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy type ${policy_type} does not support the Principal element. Remove the Principal element."
```

## エラーの解決

Principal 要素は、リソースへのアクセスを許可または拒否するプリンシパルを指定します。Principal エレメントを IAM アイデンティティベースのポリシーで使用することはできません。

ん。これは、IAM ロール用の信頼ポリシーおよびリソースベースのポリシーで使用することができます。リソースベースのポリシーは、リソースに直接埋め込むポリシーです。たとえば、Amazon S3 バケットあるいは AWS KMS キーにポリシーを埋め込むことができます。

### 関連用語

- [AWS JSON ポリシーエレメント: プリンシパル](#)
- [IAM でのクロスアカウントのリソースへのアクセス](#)

### エラー — ポリシーでサポートされていないリソース ARN

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}."
```

### エラーの解決

ポリシーが異なるリソースタイプに添付されている場合、一部のリソース ARN は、リソースベースのポリシーの Resource エレメントではサポートされていません。例えば、AWS KMS ARN は Amazon S3 バケットの Resource ポリシーではサポートされていません。リソースベースのポリシーに添付のリソースタイプでサポートされているリソース ARN を指定してください。

### 関連用語

- [JSON ポリシーエレメント: Action](#)

### エラー — サポートされていない Sid

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported Sid: Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]"
```

## エラーの解決

Sid 要素では、大文字、小文字、および数字をサポートしています。

## 関連用語

- [IAM JSON ポリシーエレメント: Sid](#)

## エラー - プリンシパルでサポートされていないワイルドカード

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported wildcard in principal: Wildcards (*, ?) are not supported with the principal key {{principal_key}}. Replace the wildcard with a valid principal value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Wildcards (*, ?) are not supported with the principal key {{principal_key}}. Replace the wildcard with a valid principal value."
```

## エラーの解決

Principal 要素構造体は、キーバリューのペアの使用をサポートしています。ポリシーで指定されたプリンシパル値には、ワイルドカード (\*) が含まれます。指定したプリンシパルキーにワイルドカードを含めることはできません。たとえば、Principal エレメント内でユーザーを指定する際に、「すべてのユーザー」の意味でワイルドカード () を使用することはできません。特定のユーザーに名前を付ける必要があります。同様に、要素で引き受けたロールセッションを指定する場合、ワイルドカードを使用して「すべてのセッション」を意味することはできません。特定のセッションに名前を付ける必要があります。また、ワイルドカードとして使用して、名前または ARN の一部に一致させることはできません。

この結果を解決するには、ワイルドカードを削除し、より具体的なプリンシパルを指定します。



## 関連用語

- [AWS JSON ポリシーエレメント: プリンシパル](#)

### エラー - 変数に中括弧がありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing brace in variable: The policy variable is missing a closing curly brace. Add } after the variable text.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy variable is missing a closing curly brace. Add } after the variable text."
```

### エラーの解決

ポリシー変数構造は、\$ プレフィックスとそれに続く中括弧 ({ }) のペアの使用をサポートします。\${ } 文字の中に、ポリシーで使用するリクエストの値の名前を含めます。

この結果を解決するには、欠落している中括弧を追加して、中括弧の完全な開閉セットが存在することを確認します。

## 関連用語

- [IAM ポリシーエレメント: 変数](#)

### エラー - 変数に引用符がありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing quote in variable: The policy variable default value must begin and end with a single quote. Add the missing quote.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy variable default value must begin and end with a single quote. Add the missing quote."
```

## エラーの解決

ポリシーに変数を追加するときに、変数のデフォルト値を指定できます。変数が存在しない場合、AWS では、指定されたデフォルトのテキストが使用されます。

変数にデフォルト値を追加するには、デフォルト値を一重引用符 ( ' ' ) で囲み、変数テキストとデフォルト値をコンマとスペース ( , ) で区切ります。

例えば、プリンシパルが `team=yellow` でタグ付けされている場合、`DOC-EXAMPLE-BUCKET-yellow` という名前で `DOC-EXAMPLE-BUCKET` Amazon S3 バケットにアクセスできます。このリソースを使用するポリシーでは、チームメンバーが自分のリソースにアクセスすることは許可されますが、他のチームのリソースにアクセスすることはできません。チームタグのないユーザーには、`company-wide` のデフォルト値を設定できます。これらのユーザーは `DOC-EXAMPLE-BUCKET-company-wide` バケットにのみアクセスでき、チームへの参加方法などの一般的な情報を確認できます。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

## 関連用語

- [IAM ポリシーエレメント: 変数](#)

### エラー — 変数でサポートされていないスペース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported space in variable: A space is not supported within the policy variable text. Remove the space.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "A space is not supported within the policy variable text. Remove the space."
```

## エラーの解決

ポリシー変数構造は、\$ プレフィックスとそれに続く中括弧 ({ }) のペアの使用をサポートします。\${ } 文字の中に、ポリシーで使用するリクエストの値の名前を含めます。デフォルトの変数を指定するときにスペースを含めることはできますが、変数名にスペースを含めることはできません。

### 関連用語

- [IAM ポリシーエレメント: 変数](#)

## エラー — 空の変数

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty variable: Empty policy variable. Remove the ${ } variable structure or provide a variable within the structure.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Empty policy variable. Remove the ${ } variable structure or provide a variable within the structure."
```

## エラーの解決

ポリシー変数構造は、\$ プレフィックスとそれに続く中括弧 ({ }) のペアの使用をサポートします。\${ } 文字の中に、ポリシーで使用するリクエストの値の名前を含めます。

### 関連用語

- [IAM ポリシーエレメント: 変数](#)

## エラー - 要素で変数がサポートされていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Variable unsupported in element: Policy variables are supported in the Resource and Condition elements. Remove the policy variable {{variable}} from this element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Policy variables are supported in the Resource and Condition elements. Remove the policy variable {{variable}} from this element."
```

## エラーの解決

Resource 要素のポリシー変数、および Condition 要素の文字列比較を活用できます。

## 関連用語

- [IAM ポリシーエレメント: 変数](#)

エラー — 変数はバージョンでサポートされていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Variable unsupported in version: To include variables in your policy, use the policy version 2012-10-17 or later.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "To include variables in your policy, use the policy version 2012-10-17 or later."
```

## エラーの解決

ポリシー変数を使用するには、ステートメントに Version エレメントを含めて、ポリシー変数をサポートするバージョンに設定する必要があります。変数はバージョン 2012-10-17 から導入されました。旧バージョンでのポリシー言語では、ポリシー変数をサポートしていません。Version を 2012-10-17 以降に設定しない場合、`${aws:username}` などの変数はポリシーでリテラル文字列として扱われます。

Version ポリシー要素は、ポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。ポリシーバージョンは、IAM でカスタマーマネージド型ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリ

シーが上書きされることはありません。代わりに、IAM はマネージド型ポリシーの新しいバージョンを作成します。

## 関連用語

- [IAM ポリシーエレメント: 変数](#)
- [IAM JSON ポリシーエレメント: バージョン](#)

## エラー — プライベート IP アドレス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Private IP address: aws:SourceIp works only for public IP address ranges. The values for condition key aws:SourceIp include only private IP addresses and will not have the desired effect. Update the value to include only public IP addresses.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "aws:SourceIp works only for public IP address ranges. The values for condition key aws:SourceIp include only private IP addresses and will not have the desired effect. Update the value to include only public IP addresses."
```

## エラーの解決

グローバル条件キーは、aws:SourceIp パブリック IP アドレス範囲に対してのみ機能します。このエラーは、ポリシーでプライベート IP アドレスのみが許可されている場合に表示されます。この場合、条件は一致しません。

- [aws:SourceIp グローバル条件キー](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## エラー — プライベート通知アドレス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Private NotIpAddress: The values for condition key aws:SourceIp include only private IP addresses and has no effect. aws:SourceIp works only for public IP address ranges. Update the value to include only public IP addresses.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The values for condition key aws:SourceIp include only private IP addresses and has no effect. aws:SourceIp works only for public IP address ranges. Update the value to include only public IP addresses."
```

## エラーの解決

グローバル条件キーは、aws:SourceIp パブリック IP アドレス範囲に対してのみ機能します。NotIpAddress 条件演算子を使用し、プライベートIPアドレスのみをリストすると、このエラーが発生します。この場合、条件は常に一致し、効果がありません。

- [aws:SourceIp グローバル条件キー](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## エラー — ポリシーのサイズが SCP クォータを超えています

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Policy size exceeds SCP quota: The {{policySize}} characters in the service control policy (SCP) exceed the {{policySizeQuota}} character maximum for SCPs. We recommend that you use multiple granular policies.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{policySize}} characters in the service control policy (SCP) exceed the {{policySizeQuota}} character maximum for SCPs. We recommend that you use multiple granular policies."
```

## エラーの解決

AWS Organizations サービスコントロールポリシー (SCP) では、Action または NotAction 要素での値の指定をサポートします。ただし、これらの値には、文字列の末尾にのみワイルドカード (\*) を含めることができます。つまり、iam:Get\* は指定できますが、iam:\*role は指定できません。

複数のアクションを指定するには、AWSは、個別にリストすることをお勧めします。

## 関連用語

- [AWS Organizationsのクォータ](#)
- [AWS Organizations サービスコントロールポリシー](#)

## エラー — 無効なサービスプリンシパルの形式

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid service principal format: The service principal does not match the expected format. Use the format {{expectedFormat}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The service principal does not match the expected format. Use the format {{expectedFormat}}."
```

## エラーの解決

条件のキーバリューのペアの値は、定義されたサービスプリンシパルの形式と一致する必要があります。

サービスプリンシパルは、サービスにアクセス許可を付与するために使用される識別子です。Principal エlementが、一部のグローバル条件キーとサービス固有のキーの値で、サービスプリンシパルを指定できます。サービスプリンシパルは各サービスによって定義されます。

サービスプリンシパルの識別子にはサービス名が含まれ、通常はすべて小文字で、次の形式になります。

*service-name*.amazonaws.com

サービス固有のキーによっては、サービスプリンシパルの形式が異なる場合があります。例えば、kms:ViaService 条件キーには、サービスプリンシパルは小文字で、次の形式になっている必要があります。

*service-name.AWS\_region*.amazonaws.com

## 関連用語

- [サービスプリンシパル](#)
- [AWS グローバル条件キー](#)
- [kms:ViaService 条件キー](#)

エラー — 条件にタグキーがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing tag key in condition: The condition key {{conditionKeyName}} must include a tag key to control access based on tags. Use the format {{conditionKeyName}}tag-key and specify a key name for tag-key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key {{conditionKeyName}} must include a tag key to control access based on tags. Use the format {{conditionKeyName}}tag-key and specify a key name for tag-key."
```

## エラーの解決

タグに基づいてアクセスを制御するには、ポリシーの[条件要素](#)でタグ情報を提供します。

たとえば、[AWS リソースへのアクセスをコントロール](#)するには、aws:ResourceTag 条件キーを含めます。このキーには、形式 aws:ResourceTag/*tag-key* が必要です。条件でタグキー owner とタグ値 JaneDoe を指定するには、次の形式を使用します。

```
"Condition": {
  "StringEquals": {"aws:ResourceTag/owner": "JaneDoe"}
}
```

## 関連用語

- [タグを使用したアクセス制御](#)
- [条件](#)
- [グローバル条件キー](#)
- [AWS サービス条件キー](#)



## エラー — VPC 形式が無効です

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid vpc format: The VPC identifier in the condition key value is not valid. Use the prefix 'vpc-' followed by 8 or 17 alphanumeric characters.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The VPC identifier in the condition key value is not valid. Use the prefix 'vpc-' followed by 8 or 17 alphanumeric characters."
```

### エラーの解決

aws:SourceVpc 条件キーにはプレフィックスとして vpc- を使用し、続けて 8 文字または 17 文字の英数字を使用する必要があります。例えば、vpc-11223344556677889 または vpc-12345678 のようになります。

### 関連用語

- [AWS グローバル条件キー: aws:SourceVpc](#)

## エラー — VPCE 形式が無効です

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid vpce format: The VPCE identifier in the condition key value is not valid. Use the prefix 'vpce-' followed by 8 or 17 alphanumeric characters.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The VPCE identifier in the condition key value is not valid. Use the prefix 'vpce-' followed by 8 or 17 alphanumeric characters."
```

### エラーの解決

aws:SourceVpce 条件キーにはプレフィックスとして vpce- を使用し、続けて 8 文字または 17 文字の英数字を使用する必要があります。例えば、vpce-11223344556677889 または vpce-12345678 のようになります。

## 関連用語

- [AWS グローバル条件キー: aws:SourceVpce](#)

エラー — フェデレーティッドプリンシパルはサポートされていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Federated principal not supported: The policy type does not support a federated identity provider in the principal element. Use a supported principal.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The policy type does not support a federated identity provider in the principal element. Use a supported principal."
```

## エラーの解決

Principal エlement は、IAM ロールに添付された信頼ポリシーにはフェデレーティッドプリンシパルを使用し、ID フェデレーションを通じてアクセスを提供します。ID ポリシー、およびその他のリソースベースのポリシーは、Principal Element でフェデレーティッド ID プロバイダーをサポートしていません。例えば、Amazon S3 バケットポリシーでは、SAML プリンシパルを使用できません。Principal Element を、サポートされているプリンシパルタイプに変更します。

## 関連用語

- [ID フェデレーション用のロールの作成](#)
- [JSON ポリシーの要素: プリンシパル](#)

エラー — 条件キーでサポートされていないアクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported action for condition key: The following actions: {{actions}} are not supported by the condition key {{key}}. The condition will not be evaluated for these actions. We recommend that you move these actions to a different statement without this condition key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The following actions: {{actions}} are not supported by the condition key {{key}}. The condition will not be evaluated for these actions. We recommend that you move these actions to a different statement without this condition key."
```

## エラーの解決

ポリーステートメントの Condition エlementにある条件キーが、Action エlementのそれぞれのアクションに適用されていることを確認してください。指定したアクションが、ポリシーによってしっかりと許可または拒否されるようにするには、サポートされないアクションを、条件キーが含まれない別のステートメントに移動する必要があります。

### Note

このエラーのために、Action エlementにワイルドカード付きのアクションがある場合は、IAM Access Analyzer がこれらのアクションを評価しません。

## 関連用語

- [JSON ポリシーElement: Action](#)

## エラー — ポリシーでサポートされていないアクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported action in policy: The action {{action}} is not supported for the resource-based policy attached to the resource type {{resourceType}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy attached to the resource type {{resourceType}}."
```

## エラーの解決

一部のアクションは、異なるリソースタイプに添付された、リソースベースのポリシーの Action エlement ではサポートされていません。例えば、AWS Key Management Service アクションは、Amazon S3 バケットのポリシーではサポートされていません。リソースベースのポリシーに添付のリソースタイプでサポートされているアクションを指定してください。

## 関連用語

- [JSON ポリシーエレメント: Action](#)

エラー — ポリシーでサポートされていないリソース ARN

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}."
```

## エラーの解決

ポリシーが異なるリソースタイプに添付されている場合、一部のリソース ARN は、リソースベースのポリシーの Resource Element ではサポートされていません。例えば、AWS KMS ARN は Amazon S3 バケットの Resource ポリシーではサポートされていません。リソースベースのポリシーに添付のリソースタイプでサポートされているリソース ARN を指定してください。

## 関連用語

- [JSON ポリシーエレメント: Action](#)

エラー — サービスプリンシパルでサポートされていない条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unsupported condition key for service principal: The following condition keys are not supported when used with the service principal: {{conditionKeys}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The following condition keys are not supported when used with the service principal: {{conditionKeys}}."
```

## エラーの解決

サービスの識別子であるサービスプリンシパルを使用して、リソースベースのポリシーの AWS のサービス エlement で Principal を指定できます。一部の条件キーは、特定のサービスプリンシパルでは使用できません。例えば、サービスプリンシパル `cloudfront.amazonaws.com` では、`aws:PrincipalOrgID` 条件キーは使用できません。Principal Element のサービスプリンシパルに適用されない条件キーは、削除する必要があります。

## 関連用語

- [サービスプリンシパル](#)
- [JSON ポリシーの要素: プリンシパル](#)

## エラー — ロール信頼ポリシー構文エラー notprincipal

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Role trust policy syntax error notprincipal: Role trust policies do not support NotPrincipal. Update the policy to use a Principal element instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Role trust policies do not support NotPrincipal. Update the policy to use a Principal element instead."
```

## エラーの解決

ロール信頼ポリシーは、IAM ロールにアタッチされているリソースベースのポリシーです。信頼ポリシーでは、ロールを引き受けることができるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーテッドユーザー) を定義します。ロール信頼ポリシーは、NotPrincipal をサポートしていません。ポリシーを更新して、代わりに Principal 要素を使用します。

## 関連用語

- [JSON ポリシーの要素: プリンシパル](#)
- [JSON ポリシーエレメント: NotPrincipal](#)

エラー - ロール信頼ポリシーはプリンシパルでワイルドカードをサポートしていません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Role trust policy unsupported wildcard in principal: "Principal:" "*" is not supported in the principal element of a role trust policy. Replace the wildcard with a valid principal value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "\"Principal:\" \"*\" is not supported in the principal element of a role trust policy. Replace the wildcard with a valid principal value."
```

## エラーの解決

ロール信頼ポリシーは、IAM ロールにアタッチされているリソースベースのポリシーです。信頼ポリシーは、ロールを引き受けることができるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーテッドユーザー) を定義します。"Principal:" "\*" は、ロール信頼ポリシーの Principal 要素ではサポートされていません。ワイルドカードを有効なプリンシパル値に置き換えてください。

## 関連用語

- [JSON ポリシーの要素: プリンシパル](#)

エラー — ロール信頼ポリシー構文エラー リソース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Role trust policy syntax error resource: Role trust policies apply to the role that they are attached to. You cannot specify a resource. Remove the Resource or NotResource element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Role trust policies apply to the role that they are attached to. You cannot specify a resource. Remove the Resource or NotResource element."
```

## エラーの解決

ロール信頼ポリシーは、IAM ロールにアタッチされているリソースベースのポリシーです。信頼ポリシーでは、ロールを引き受けることができるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーテッドユーザー) を定義します。ロール信頼ポリシーは、アタッチされているロールに適用されます。ロール信頼ポリシーで、Resource または NotResource 要素を指定することはできません。Resource または NotResource 要素を削除します。

- [JSON ポリシーエレメント: Resource](#)
- [JSON ポリシーエレメント: NotResource](#)

## エラー – IP 範囲のタイプが一致しません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch IP range: The condition operator {{operator}} is used with an invalid IP range value. Specify the IP range in standard CIDR format.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition operator {{operator}} is used with an invalid IP range value. Specify the IP range in standard CIDR format."
```

## エラーの解決

CIDR 形式の IP アドレス条件演算子のデータタイプを使用するようにテキストを更新します。

## 関連用語

- [IP アドレス条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

## エラー — 条件キーのアクションがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing action for condition key: The {{actionName}} action must be in the action block to allow setting values for the condition key {{keyName}}. Add {{actionName}} to the action block.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{actionName}} action must be in the action block to allow setting values for the condition key {{keyName}}. Add {{actionName}} to the action block."
```

## エラーの解決

ポリシーステートメントの Condition 要素の条件キーは、指定されたアクションが Action 要素内にはない限り、評価されません。指定した条件キーが、ポリシーによってしっかりと許可または拒否されるようにするには、Action 要素にアクションを追加します。

## 関連用語

- [JSON ポリシーエレメント: Action](#)

## エラー — ロール信頼ポリシー内のフェデレーションプリンシパル構文が無効です

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid federated principal syntax in role trust policy: The principal value specifies a federated principal that does not match the expected format. Update the federated principal to a domain name or a SAML metadata ARN.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The principal value specifies a federated principal that does not match the expected format. Update the federated principal to a domain name or a SAML metadata ARN."
```



## エラーの解決

プリンシパル値はが、期待される形式と一致しないフェデレーションプリンシパルを指定しています。フェデレーションプリンシパルの形式を有効なドメイン名または SAML メタデータ ARN に更新します。

### 関連用語

- [フェデレーテッドユーザーとロール](#)

## エラー — プリンシパルのアクションが一致しません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Mismatched action for principal: The {{actionName}} action is invalid with the following principal(s): {{principalNames}}. Use a SAML provider principal with the sts:AssumeRoleWithSAML action or use an OIDC provider principal with the sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use either of the two options.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{actionName}} action is invalid with the following principal(s): {{principalNames}}. Use a SAML provider principal with the sts:AssumeRoleWithSAML action or use an OIDC provider principal with the sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use either of the two options."
```

## エラーの解決

ポリシーステートメントの Action 要素で指定されたアクションが、Principal 要素で指定されたプリンシパルで無効です。例えば、SAML プロバイダーのプリンシパルを sts:AssumeRoleWithWebIdentity アクションで使用することはできません。sts:AssumeRoleWithSAML アクションで SAML プロバイダー プリンシパルを使用するか、sts:AssumeRoleWithWebIdentity アクションで OIDC プロバイダー プリンシパルを使用する必要があります。

### 関連用語

- [AssumeRoleWithSAML](#)

- [AssumeRoleWithWebIdentity](#)

エラー — ロールエニウェア信頼ポリシーのアクションが見つかりません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing action for roles anywhere trust policy: The rolesanywhere.amazonaws.com service principal requires the sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession permissions to assume a role. Add the missing permissions to the policy.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The rolesanywhere.amazonaws.com service principal requires the sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession permissions to assume a role. Add the missing permissions to the policy."
```

## エラーの解決

IAM Roles Anywhere がロールを引き受けて一時的に AWS 認証情報を配信できるようにするには、ロールが IAM Roles Anywhere のサービスプリンシパルを信頼する必要があります。IAM Roles Anywhere のサービスプリンシパルに `sts:AssumeRole`、`sts:SetSourceIdentity`、およびロールを引き受けるための `sts:TagSession` 許可が必要です。いずれかの許可が見つからない場合、ポリシーに追加する必要があります。

## 関連用語

- [AWS Identity and Access Management Roles Anywhere の信頼モデル](#)

一般的な警告 — NotResource で SLR を作成する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with NotResource: Using the iam:CreateServiceLinkedRole action with NotResource can allow creation of unintended service-linked roles for multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with NotResource can allow creation of unintended service-linked roles for multiple resources. We recommend that you specify resource ARNs instead."
```

### 一般的な警告の解決

アクション `iam:CreateServiceLinkedRole` は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。ポリシーで `iam:CreateServiceLinkedRole` 要素を使用して `NotResource` を使用すると、複数のリソースに対して意図しないサービスリンクロールを作成できる可能性があります。代わりに、AWS では、`Resource` 要素で許可されたARNを指定することをお勧めします。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

### 一般的な警告 — スターインアクションで `NotResource` の SLR を作成する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with star in action and NotResource: Using an action with a wildcard(*) and NotResource can allow creation of unintended service-linked roles because it can allow iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using an action with a wildcard(*) and NotResource can allow creation of unintended service-linked roles because it can allow iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

### 一般的な警告の解決

アクション `iam:CreateServiceLinkedRole` は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。Action にワイルドカード (\*) が含まれ、`NotResource` 要素を含むポリシーにより、複数のリソースに対して意図しないサービ

スリンクロールを作成できる場合があります。AWSでは、代わりに Resource 要素で許可された ARN を指定することをお勧めします。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

一般的な警告 — NotAction および NotResource を使用した SLR の作成

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with NotAction and NotResource: Using NotAction with NotResource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using NotAction with NotResource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

一般的な警告の解決

アクション iam:CreateServiceLinkedRole は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。NotAction 要素を NotResource 要素とともに使用すると、複数のリソースに対して意図しないサービスリンクロールを作成できる可能性があります。AWS では、代わりに iam:CreateServiceLinkedRole 要素の ARN の限定リストで Resource を許可するようにポリシーを書き直すことをお勧めします。iam:CreateServiceLinkedRole 要素に NotAction を追加することもできます。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: 非アクション](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

## 一般的な警告 — リソースにスター付き SLR を作成する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with star in resource: Using the iam:CreateServiceLinkedRole action with wildcards (*) in the resource can allow creation of unintended service-linked roles. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with wildcards (*) in the resource can allow creation of unintended service-linked roles. We recommend that you specify resource ARNs instead."
```

### 一般的な警告の解決

アクション `iam:CreateServiceLinkedRole` は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。 `iam:CreateServiceLinkedRole` 要素にワイルドカード (\*) を使用してポリシーで `Resource` を使用すると、複数のリソースに対して意図しないサービスリンクロールを作成できる可能性があります。AWSでは、代わりに `Resource` 要素で許可されたARNを指定することをお勧めします。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: リソース](#)

### AWSこの一般的な警告を伴う管理ポリシー

[AWS 管理ポリシー](#) を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

これらのユースケースには、アカウント内のパワーユーザー向けのものが含まれています。次の AWS 管理ポリシーは、パワーユーザーアクセスを提供し、任意の [AWS サービスのサービスリンクロール](#) を作成するためのアクセス許可を付与します。AWSでは、パワーユーザーと見なす IAM ID のみ、次の AWS 管理対象ポリシーをアタッチすることをお勧めします。

- [PowerUserAccess](#)
- [AlexaForBusinessFullAccess](#)

- [AWSOrganizationsServiceTrustPolicy](#) – この AWS 管理ポリシーは、AWS Organizations サービスにリンクされたロールで使用するための許可を提供します。このロールを使用すると、Organizationsは、AWS 組織内の他のサービスに対してサービスにリンクされた追加のロールを作成できます。

#### 一般的な警告 — アクションとリソースにスター付き SLR を作成する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with star in action and resource: Using wildcards (*) in the action and the resource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using wildcards (*) in the action and the resource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead."
```

#### 一般的な警告の解決

アクション `iam:CreateServiceLinkedRole` は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。Action 要素とResource 要素にワイルドカード (\*) が含まれるポリシーでは、複数のリソースに対して意図しないサービスリンクロールを作成できます。これにより "Action": "\*"、"Action": "iam:\*"、"Action": "iam:Create\*"、またはを指定したときにサービスにリンクされたロールを作成できます。AWS では、代わりに Resource 要素で許可されたARNを指定することをお勧めします。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: リソース](#)

#### AWSこの一般的な警告を伴う管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

これらのユースケースの中には、アカウント内の管理者向けのものがあります。次の AWS 管理ポリシーは、管理者アクセスを提供し、任意の AWS サービスの [サービスリンクロール](#) を作成するためのアクセス許可を付与します。AWS では、次の AWS 管理ポリシーを、管理者と見なす IAM ID にのみアタッチすることをお勧めします。

- [AdministratorAccess](#)
- [IAMFullAccess](#)

一般的な警告 — リソースにスターがあり、NotAction で SLR を作成する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Create SLR with star in resource and NotAction: Using a resource with wildcards (*) and NotAction can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead."
```

一般的な警告の解決

アクション `iam:CreateServiceLinkedRole` は AWS のサービスがユーザーに代わってアクションを実行できるようにする IAM ロールを作成する許可を付与します。ポリシーで NotAction 要素を使用し、Resource 要素にワイルドカード (\*) を使用すると、複数のリソースに対して意図しないサービスリンクロールを作成できるようになります。AWS では、代わりに Resource 要素で許可された ARN を指定することをお勧めします。iam:CreateServiceLinkedRole 要素に NotAction を追加することもできます。

- [CreateServiceLinkedRole オペレーション](#)
- [IAM JSON ポリシーエレメント: 非アクション](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: リソース](#)

## 一般的な警告 — 非推奨のグローバル条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Deprecated global condition key: We recommend that you update aws:ARN to use the newer condition key aws:PrincipalArn.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "We recommend that you update aws:ARN to use the newer condition key aws:PrincipalArn."
```

### 一般的な警告の解決

ポリシーには、非推奨のグローバル条件キーが含まれています。条件キーと値のペアの条件キーを更新して、サポートされているグローバル条件キーを使用します。

- [グローバル条件キー](#)

### 一般的な警告 - 無効な日付値

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid date value: The date {{date}} might not resolve as expected. We recommend that you use the YYYY-MM-DD format.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The date {{date}} might not resolve as expected. We recommend that you use the YYYY-MM-DD format."
```

### 一般的な警告の解決

Unix エポック時間は、1970 年 1 月 1 日から経過した時点からうるう秒を引いた時点を記述します。エポック時間は、予想される正確な時間に解決されない場合があります。AWS では、日付と時刻の書式に W3C 標準を使用することをお勧めします。たとえば、YYYY-MM-DD (1997-07-16) のように



完全な日付を指定したり、YYYY-MM-DDThh:mm:ssTZD (1997-07-16T19:20:30+01:00) のように 2 番目の日付に時刻を追加することもできます。

- [W3C 日付と時刻の形式](#)
- [IAM JSON ポリシーエレメント: バージョン](#)
- [aws:CurrentTime グローバル条件キー](#)

#### 一般的な警告 - 無効なロールリファレンス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid role reference: The Principal element includes the IAM role ID {{roleid}}. We recommend that you use a role ARN instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Principal element includes the IAM role ID {{roleid}}. We recommend that you use a role ARN instead."
```

#### 一般的な警告の解決

AWSでは、IAM ロールには、プリンシパル ID ではなく Amazon リソースネーム (ARN) を指定することをお勧めします。IAM がポリシーを保存すると、ARN が既存のロールのプリンシパル ID に変換されます。AWS には、安全上の注意事項が含まれています。ロールを削除して再作成すると、そのロールには新しい ID が割り当てられ、ポリシーは新しいロールの ID と一致しません。

- [プリンシパルの指定:IAM ロール](#)
- [IAM ARN](#)
- [IAM 一意の ID](#)

#### 一般的な警告 — 無効なユーザーリファレンス

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Invalid user reference: The Principal element includes the IAM user ID {{userid}}. We recommend that you use a user ARN instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The Principal element includes the IAM user ID {{userid}}. We recommend that you use a user ARN instead."
```

### 一般的な警告の解決

AWSでは、IAM ユーザーに対して、プリンシパルIDではなく Amazon リソースネーム (ARN) を指定することをお勧めします。IAM がポリシーを保存すると、ARN が既存のユーザーのプリンシパル ID に変換されます。AWS には、安全上の注意事項が含まれています。誰かがユーザーを削除して再作成すると、そのユーザーには新しい ID が割り当てられ、ポリシーは新しいユーザーの ID と一致しません。

- [プリンシパルの指定:IAM ユーザー](#)
- [IAM ARN](#)
- [IAM 一意の ID](#)

### 一般的な警告-バージョンがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing version: We recommend that you specify the Version element to help you with debugging permission issues.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "We recommend that you specify the Version element to help you with debugging permission issues."
```

### 一般的な警告の解決

AWS ではポリシーにオプションの Version パラメータを含めることをお勧めします。Version 要素を含めない場合、デフォルト値は 2012-10-17 ですが、ポリシー変数などの新しい機能はポリシーでは機能しません。たとえば、`${aws:username}` などの変数は変数として認識されず、代わりにポリシー内のリテラル文字列として扱われます。

- [IAM JSON ポリシーエレメント: バージョン](#)

#### 一般的な警告 — 固有の Sid を推奨

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Unique Sids recommended: We recommend that you use statement IDs that are unique to your policy. Update the Sid value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "We recommend that you use statement IDs that are unique to your policy. Update the Sid value."
```

#### 一般的な警告の解決

AWS では、一意のステートメント ID を使用することをお勧めします。Sid (ステートメントID) 要素を使用すると、ポリシーステートメントに指定するオプションの識別子を入力できます。SID 要素を使用して、ステートメント配列内の各ステートメントにステートメントID値を割り当てることができます。

#### 関連用語

- [IAM JSON ポリシーエレメント: Sid](#)

#### 一般的な警告 – 演算子のようなものがないワイルドカード

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Wildcard without like operator: Your condition value includes a * or ? character. If you meant to use a wildcard (*, ?), update the condition operator to include Like.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Your condition value includes a * or ? character. If you meant to use a wildcard (*, ?), update the condition operator to include Like."
```

## 一般的な警告の解決

Condition 要素構造では、条件演算子とキーと値のペアを使用する必要があります。ワイルドカード (\*、?) を使用する条件値を指定する場合は、Like バージョンの条件演算子を使用する必要があります。たとえば、StringEquals 文字列条件演算子の代わりに、StringLike を使用します。

```
"Condition": {"StringLike": {"aws:PrincipalTag/job-category": "admin-*"}}
```

- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## AWSこの一般的な警告を伴う管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS ユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

次の AWS 管理ポリシーでは、パターンマッチング用の Like を含む条件演算子を使用せずに、条件値にワイルドカードが含まれています。AWS 管理ポリシーを参照として使用して顧客管理ポリシーを作成する場合、AWS では、StringLike などのワイルドカード (\*、?) とのパターンマッチングをサポートする条件演算子を使用することをお勧めします。

- [AWSGlueConsoleSageMakerNotebookFullAccess](#)

## 一般的な警告 — ポリシーのサイズが ID ポリシーのクォータを超えています

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Policy size exceeds identity policy quota: The {{policySize}} characters in the identity policy, excluding whitespace, exceed the {{policySizeQuota}} character maximum for inline and managed policies. We recommend that you use multiple granular policies.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{policySize}} characters in the identity policy, excluding whitespace, exceed the {{policySizeQuota}} character maximum for inline and managed policies. We recommend that you use multiple granular policies."
```

## 一般的な警告の解決

最大 10 のマネージド型ポリシーを IAM ID (ユーザー、ユーザーのグループ、ロール) にアタッチできます。ただし、各管理ポリシーのサイズは、デフォルトのクォータである 6,144 文字を超えることはできません。IAM では、このクォータに対するポリシーのサイズを計算する際にスペースはカウントしません。AWS のクォータ (制限とも呼ばれます) は、AWS アカウントのリソース、アクション、および制限の最大値です。

さらに、IAM アイデンティティに必要な数のインラインポリシーを追加できます。ただし、アイデンティティごとのすべてのインラインポリシーの合計サイズは、指定されたクォータを超えることはできません。

ポリシーがクォータより大きい場合は、ポリシーを複数のステートメントに整理し、ステートメントを複数のポリシーにグループ化できます。

### 関連用語

- [IAM および AWS STS 文字クォータ](#)
- [複数のステートメントと複数のポリシー](#)
- [IAM カスタマーマネージド型ポリシー](#)
- [JSON ポリシー概要](#)
- [IAM JSON ポリシー文法](#)

### AWSこの一般的な警告を伴う管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

次の AWS 管理ポリシーは、多くの AWS サービスにわたるアクションにアクセス許可を付与し、最大ポリシーサイズを超えています。管理ポリシーを作成するための参照として AWS 管理ポリシーを使用する場合は、ポリシーを複数のポリシーに分割する必要があります。

- [ReadOnlyAccess](#)
- [AWSSupportServiceRolePolicy](#)

一般的な警告 — ポリシーのサイズが リソースポリシーのクォータを超えています

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Policy size exceeds resource policy quota: The {{policySize}} characters in the resource policy exceed the {{policySizeQuota}} character maximum for resource policies. We recommend that you use multiple granular policies.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{policySize}} characters in the resource policy exceed the {{policySizeQuota}} character maximum for resource policies. We recommend that you use multiple granular policies."
```

### 一般的な警告の解決

リソースベースのポリシーは、Amazon S3 バケットなどのリソースにアタッチする JSON ポリシードキュメントです。これらのポリシーでは、そのリソースに対して特定のアクションを実行するために指定されたプリンシパルのアクセス許可を付与するとともに、このアクセス許可が適用される条件を定義します。リソースベースのポリシーのサイズは、そのリソースに設定されたクォータを超えることができません。AWS のクォータ (制限とも呼ばれます) は、AWS アカウントのリソース、アクション、および制限の最大値です。

ポリシーがクォータより大きい場合は、ポリシーを複数のステートメントに整理し、ステートメントを複数のポリシーにグループ化できます。

### 関連用語

- [リソースベースのポリシー](#)
- [Amazon S3 バケットポリシー](#)
- [複数のステートメントと複数のポリシー](#)
- [JSON ポリシー概要](#)
- [IAM JSON ポリシー文法](#)

### 一般的な警告-タイプの不一致

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch: Use the operator type {{allowed}} instead of operator {{operator}} for the condition key {{key}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Use the operator type {{allowed}} instead of operator {{operator}} for the condition key {{key}}."
```

### 一般的な警告の解決

サポートされている条件演算子データ型を使用するようにテキストを更新します。

たとえば、aws:MultiFactorAuthPresent グローバル条件キーには、Boolean データ型の条件演算子が必要です。日付または整数を指定すると、データ型が一致しません。

### 関連用語

- [グローバル条件キー](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

### 一般的な警告 — 型の不一致ブール値

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch Boolean: Add a valid Boolean value (true or false) for the condition operator {{operator}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a valid Boolean value (true or false) for the condition operator {{operator}}."
```

### 一般的な警告の解決

true または false などのブール条件演算子データ型を使用するようにテキストを更新します。

たとえば、aws:MultiFactorAuthPresent グローバル条件キーには、Boolean データ型の条件演算子が必要です。日付または整数を指定すると、データ型が一致しません。

### 関連用語

- [ブール条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

#### 一般的な警告-タイプが一致しない日付

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch date: The date condition operator is used with an invalid value. Specify a valid date using YYYY-MM-DD or other ISO 8601 date/time format.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The date condition operator is used with an invalid value. Specify a valid date using YYYY-MM-DD or other ISO 8601 date/time format."
```

#### 一般的な警告の解決

YYYY-MM-DD または他のISO8601 日時形式で、日付条件演算子のデータ型を使用するようにテキストを更新します。

#### 関連用語

- [日付条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

#### 一般的な警告 - タイプ不一致番号

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch number: Add a valid numeric value for the condition operator {{operator}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a valid numeric value for the condition operator {{operator}}."
```

#### 一般的な警告の解決



数値条件演算子のデータ型を使用するようにテキストを更新します。

#### 関連用語

- [数値条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

一般的な警告-型が一致しない文字列

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Type mismatch string: Add a valid base64-encoded string value for the condition operator {{operator}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a valid base64-encoded string value for the condition operator {{operator}}."
```

#### 一般的な警告の解決

文字列条件演算子のデータ型を使用するようにテキストを更新します。

#### 関連用語

- [文字列条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)

一般的な警告 — 特定の github リポジトリとブランチを推奨

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Specific github repo and branch recommended: Using a wildcard (*) in token.actions.githubusercontent.com:sub can allow requests from more sources than you intended. Specify the value of token.actions.githubusercontent.com:sub with the repository and branch name.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using a wildcard (*) in token.actions.githubusercontent.com:sub can allow requests from more sources than you intended. Specify the value of token.actions.githubusercontent.com:sub with the repository and branch name."
```

### 一般的な警告の解決

GitHub を OIDC IdP として使用する場合、ベストプラクティスは、IAM IdP に関連付けられたロールを引き受けることができるエンティティを制限することです。ロール信頼ポリシーに Condition ステートメントを含めると、ロールを特定の GitHub Organization、リポジトリ、またはブランチに制限できます。条件キー `token.actions.githubusercontent.com:sub` を使用してアクセスを制限できます。条件を特定のリポジトリまたはブランチのセットに制限することをお勧めします。`token.actions.githubusercontent.com:sub` でワイルドカード (\*) を使用する場合、管理外の Organization またはリポジトリの GitHub Actions は、AWS アカウントで GitHub IAM IdP に関連付けられたロールを引き受けることができます。

### 関連用語

- [GitHub OIDC ID プロバイダーのロールの設定](#)

一般的な警告 — ポリシーのサイズが信頼ポリシーのクォータを超えています

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Policy size exceeds role trust policy quota: The characters in the role trust policy, excluding whitespace, exceed the character maximum. We recommend that you request a role trust policy length quota increase using Service Quotas and AWS Support Center. If the quotas have already been increased, then you can ignore this warning.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The characters in the role trust policy, excluding whitespace, exceed the character maximum. We recommend that you request a role trust policy length quota increase using Service Quotas and AWS Support Center. If the quotas have already been increased, then you can ignore this warning."
```

### 一般的な警告の解決

IAM と AWS STS に、ロール信頼ポリシーのサイズを制限するクォータがあります。ロール信頼ポリシー内の文字数 (空白を除く) が最大文字数を超えています。Service Quotas と AWS Support

Center Console を使用して、ロール信頼ポリシーの長さのクォータの増加をリクエストすることをお勧めします。

## 関連用語

- [IAM クォータおよび AWS STS クォータ、名前の要件、および文字の制限](#)

## セキュリティ警告 — NotPrincipal で許可する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Allow with NotPrincipal: Using Allow with NotPrincipal can be overly permissive. We recommend that you use Principal instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend that you use Principal instead."
```

## セキュリティ警告を解決する

"Effect": "Allow" で NotPrincipal を使用することは過度に許容される可能性があります。たとえば、これにより匿名プリンシパルにアクセス許可を付与できます。AWS では、Principal 要素を使用してアクセスする必要があるプリンシパルを指定することをお勧めします。または、広範なアクセスを許可し、NotPrincipal および要素 "Effect": "Deny" 使用して別のステートメントを追加することもできます。

- [AWS JSON ポリシーエレメント: プリンシパル](#)
- [AWS JSON ポリシーエレメント: 非プリンシパル](#)

## セキュリティ警告 - 単一の値キーを持つ ForAllValues

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
ForAllValues with single valued key: Using ForAllValues qualifier with the single-valued condition key {{key}} can be overly permissive. We recommend that you remove ForAllValues:.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using ForAllValues qualifier with the single-valued condition key {{key}} can be overly permissive. We recommend that you remove ForAllValues:."
```

### セキュリティ警告を解決する

AWS では、複数値の条件でのみ ForAllValues を使用することをお勧めします。ForAllValues 集合演算子は、リクエストセットのすべてのメンバーの値が条件キーセットのサブセットであるかどうかをテストします。リクエストのすべてのキーバリューがポリシーの 1 つ以上の値と一致する場合、条件は true を返します。また、リクエストにキーがない場合、またはキーバリューが空の文字列などの null データセットに解決される場合は true を返します。

条件が単一の値または複数の値をサポートしているかどうかを確認するには、サービスの[アクション、リソース、および条件キー](#)のページをご確認ください。ArrayOf データタイプのプレフィックスを持つ条件キーは、複数の値を持つ条件キーです。例えば、Amazon SES は単一の値 (String) を持つキーと、複数の値を持つ ArrayOfString のデータタイプをサポートしています。

- [複数値のコンテキストキー](#)

### セキュリティ警告 — NotResource でロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Pass role with NotResource: Using the iam:PassRole action with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the iam:PassRole action with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

### セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、ID (ユーザー、ユーザーのグループ、またはロール) に iam:PassRole アクセス許

可を付与する必要があります。ポリシーで `iam:PassRole` 要素を使用して `NotResource` を使用すると、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようになります。AWS では、代わりに、`Resource` 要素で許可されたARNを指定することをお勧めします。さらに、`iam:PassedToService` 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

### セキュリティ警告 — スター付きアクションで `NotResource` でロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Pass role with star in action and NotResource: Using an action with a wildcard (*) and NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using an action with a wildcard (*) and NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

### セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、`iam:PassRole` アクセス許可をアイデンティティ (ユーザー、ユーザーのグループ、ロール) に送信するには、アクセス許可が必要です。Action にワイルドカード (\*) があり、`NotResource` 要素を含むポリシーにより、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようになります。AWS では、代わりに `Resource` 要素で許可されたARNを指定することをお勧めします。さらに、`iam:PassedToService` 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)

- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

### セキュリティ警告 — NotAction および NotResource でロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Pass role with NotAction and NotResource: Using NotAction with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources.. We recommend that you specify resource ARNs instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using NotAction with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources.. We recommend that you specify resource ARNs instead."
```

### セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、iam:PassRole アクセス許可をアイデンティティ (ユーザー、ユーザーのグループ、ロール) に送信するには、アクセス許可が必要です。NotAction 要素を使用し、NotResource 要素にいくつかのリソースをリストすると、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようになります。AWS では、代わりに、Resource 要素で許可されたARNを指定することをお勧めします。さらに、iam:PassedToService 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: リソース](#)

### セキュリティ警告 — リソースにスターでロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

Pass role with star in resource: Using the iam:PassRole action with wildcards (\*) in the resource can be overly permissive because it allows iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the iam:PassRole action with wildcards (*) in the resource can be overly permissive because it allows iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

## セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、ID (ユーザー、ユーザーのグループ、またはロール) に iam:PassRole アクセス許可を付与する必要があります。iam:PassRole を許可し、Resource 要素にワイルドカード (\*) を含むポリシーにより、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようになります。AWS では、代わりに、Resource 要素で許可された ARN を指定することをお勧めします。さらに、iam:PassedToService 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

一部の AWS サービスには、役割の名前にサービス名前空間が含まれています。このポリシーチェックでは、ポリシーを分析して結果を生成する際に、これらの規則を考慮に入れます。たとえば、次のリソース ARN は結果を生成しない可能性があります。

```
arn:aws:iam::*:role/Service*
```

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)
- [IAM JSON ポリシーエレメント: リソース](#)

## AWS マネージド型ポリシーとこのセキュリティ警告

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

これらのユースケースの 1 つは、アカウント内の管理者向けです。次の AWS 管理ポリシーは、管理者アクセスを提供し、任意の IAM ロールを任意のサービスに渡すためのアクセス許可を付与します。AWSでは、次の AWS 管理対象ポリシーは、管理者と見なすIAMIDにのみアタッチすることをお勧めします。

- [AdministratorAccess-Amplify](#)

次の AWS管理ポリシーには、リソースにワイルドカード (\*) を使用した iam:PassRole へのアクセス許可が含まれており、[非推奨パス](#)上にあります。これらのポリシーごとに、権限ガイダンスが更新されました。たとえば、新しい AWS マネージド型ポリシーで、ユースケースをサポートします。これらのポリシーの代替案については、[各サービス](#) のガイドを参照してください。

- AWSElasticBeanstalkFullAccess
- AWSElasticBeanstalkService
- AWSLambdaFullAccess
- AWSLambdaReadOnlyアクセス
- AWSOpsWorksFullAccess
- AWSOpsWorksRole
- AWSDataPipelineRole
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonElasticMapReduceFullAccess
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonEC2ContainerServiceFullAccess

次の AWS 管理ポリシーは、[サービスにリンクされたロール](#)にのみアクセス許可を提供します。これにより、AWS サービスはユーザーに代わってアクションを実行できます。これらのポリシーを IAM ID にアタッチすることはできません。

- [AWSServiceRoleForAmazonEKSNodegroup](#)

セキュリティ警告 — スターインアクションとリソースでロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。



Pass role with star in action and resource: Using wildcards (\*) in the action and the resource can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using wildcards (*) in the action and the resource can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

## セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、ID (ユーザー、ユーザーのグループ、またはロール) に iam:PassRole アクセス許可を付与する必要があります。Action 要素と Resource 要素にワイルドカード (\*) が含まれるポリシーでは、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようにすることができます。AWS では、代わりに Resource 要素で許可された ARN を指定することをお勧めします。さらに、iam:PassedToService 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: リソース](#)

## AWS マネージド型ポリシーとこのセキュリティ警告

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

これらのユースケースの中には、アカウント内の管理者向けのものがあります。次の AWS 管理ポリシーは、管理者アクセスを提供し、任意の IAM ロールを任意の AWS サービスに渡すためのアクセス許可を付与します。AWS では、次の管理ポリシーを、AWS 管理者と見なす IAM ID にのみアタッチすることをお勧めします。

- [AdministratorAccess](#)

- [IAMFullAccess](#)

セキュリティ警告 — リソースにスターと NotAction でロールを渡す

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Pass role with star in resource and NotAction: Using a resource with wildcards (*) and NotAction can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

セキュリティ警告を解決する

多くの AWS サービスを設定するには、そのサービスに IAM ロールを渡す必要があります。これを許可するには、ID (ユーザー、ユーザーのグループ、またはロール) に iam:PassRole アクセス許可を付与する必要があります。ポリシーで NotAction 要素を使用し、Resource 要素にワイルドカード (\*) を使用すると、プリンシパルが意図したよりも多くのサービスまたは機能にアクセスできるようになります。AWS では、代わりに Resource 要素で許可された ARN を指定することをお勧めします。さらに、iam:PassedToService 条件キーを使用して、単一のサービスへのアクセス許可を減らすことができます。

- [サービスにロールを渡す](#)
- [iam:PassedToService](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)
- [IAM JSON ポリシーエレメント: アクション](#)
- [IAM JSON ポリシーエレメント: リソース](#)

セキュリティ警告 — ペアリングされた条件キーがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing paired condition keys: Using the condition key {{conditionKeyName}}
can be overly permissive without also using the following condition keys:
{{recommendedKeys}}. Condition keys like this one are more secure when paired with
a related key. We recommend that you add the related condition keys to the same
condition block.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the condition key {{conditionKeyName}} can be overly
permissive without also using the following condition keys: {{recommendedKeys}}.
Condition keys like this one are more secure when paired with a related key. We
recommend that you add the related condition keys to the same condition block."
```

## セキュリティ警告を解決する

一部の条件キーは、他の関連する条件キーと組み合わせると、より安全になります。AWSでは、関連する条件キーを既存の条件キーと同じ条件ブロックに含めることをお勧めします。これにより、ポリシーを通じて付与されるアクセス許可がより安全になります。

たとえば、`aws:VpcSourceIp` 条件キーを使用して、要求が行われたIPアドレスをポリシーで指定したIPアドレスと比較できます。AWSでは、関連する `aws:SourceVPC` 条件キーを追加することをお勧めします。これは、リクエストがポリシーで指定したVPCと指定したIPアドレスからのものであるかどうかをチェックします。

## 関連用語

- [\(aws:VpcSourceIp グローバル条件キー\)](#)
- [\(aws:SourceVPC グローバル条件キー\)](#)
- [グローバル条件キー](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

セキュリティ警告 — サービスに対してサポートされていないタグ条件キーで拒否

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Deny with unsupported tag condition key for service: Using the effect Deny with the
tag condition key {{conditionKeyName}} and actions for services with the following
```

```
prefixes can be overly permissive: {{serviceNames}}. Actions for the listed services are not denied by this statement. We recommend that you move these actions to a different statement without this condition key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the effect Deny with the tag condition key {{conditionKeyName}} and actions for services with the following prefixes can be overly permissive: {{serviceNames}}. Actions for the listed services are not denied by this statement. We recommend that you move these actions to a different statement without this condition key."
```

## セキュリティ警告を解決する

ポリシーの Condition 要素でサポートされていないタグ条件キーを "Effect": "Deny" とともに使用すると、そのサービスでは条件が無視されるため、過度に許容される可能性があります。AWS では、条件キーをサポートしないサービスアクションを削除し、それらのアクションの特定のリソースへのアクセスを拒否する別のステートメントを作成することをお勧めします。

aws:ResourceTag 条件キーを使用していて、サービスアクションでサポートされていない場合、そのキーは要求コンテキストに含まれません。この場合、Deny ステートメントの条件は常に false を返し、アクションが拒否されることはありません。これは、リソースが正しくタグ付けされている場合でも発生します。

サービスが aws:ResourceTag 条件キーをサポートしている場合、タグを使用してそのサービスのリソースへのアクセスを制御できます。これは、[属性ベースのアクセス制御 \(ABAC\)](#) と呼ばれます。これらのキーをサポートしないサービスでは、[リソースベースのアクセス制御 \(RBAC\)](#) を使用してリソースへのアクセスを制御する必要があります。

### Note

一部のサービスでは、リソースとアクションのサブセットの aws:ResourceTag 条件キーのサポートが許可されています。IAM Access Analyzer は、サポートされていないサービスアクションの結果を返します。たとえば、Amazon S3はそのリソースのサブセットに対して aws:ResourceTag をサポートしています。aws:ResourceTag 条件キーをサポートする Amazon S3で利用可能なすべてのリソースタイプを表示するには、『サービス認証リファレンス』の「[Amazon S3 で定義されたリソースタイプ](#)」を参照してください。

たとえば、キーと値のペア `status=Confidential` でタグ付けされた特定のリソースのタグ付け解除へのアクセスを拒否するとします。また、AWS Lambda ではリソースのタグ付けとタグ付け解除が可能です。ただし、`aws:ResourceTag` 条件キーはサポートされていないと仮定します。このタグが存在する場合に AWS App Mesh および AWS Backup の削除アクションを拒否するには、`aws:ResourceTag` 条件キーを使用します。Lambda の場合は、`"Confidential"` プレフィックスを含むリソース命名規則を使用します。次に、その命名規則でリソースの削除を防止する別のステートメントを含めます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteSupported",
      "Effect": "Deny",
      "Action": [
        "appmesh:DeleteMesh",
        "backup:DeleteBackupPlan"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/status": "Confidential"
        }
      }
    },
    {
      "Sid": "DenyDeleteUnsupported",
      "Effect": "Deny",
      "Action": "lambda:DeleteFunction",
      "Resource": "arn:aws:lambda:*:123456789012:function:status-Confidential*"
    }
  ]
}
```

#### Warning

この結果の回避策として、条件演算子の...[IfExists](#) バージョンを使用しないでください。これは、キーがリクエストコンテキストに存在し、値が一致する場合、アクションを拒否することを意味します。それ以外の場合は、アクションを拒否します。前の例では、`lambda:DeleteFunction` 演算子を使用して `DenyDeleteSupported` ステートメントに `StringEqualsIfExists` アクションを含めると、常にアクションが拒否されます。

そのアクションでは、キーはコンテキストに存在せず、リソースがタグ付けされているかどうかにかかわらず、そのリソースタイプを削除しようとするすべての試みは拒否されます。

## 関連用語

- [グローバル条件キー](#)
- [ABAC と RBAC の比較](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

セキュリティ警告 — サービスに対してサポートされていないタグ条件キーで NotAction を拒否する AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Deny NotAction with unsupported tag condition key for service: Using the effect Deny with NotAction and the tag condition key {{conditionKeyName}} can be overly permissive because some service actions are not denied by this statement. This is because the condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the effect Deny with NotAction and the tag condition key {{conditionKeyName}} can be overly permissive because some service actions are not denied by this statement. This is because the condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction."
```

## セキュリティ警告を解決する

ポリシーの Condition 要素で要素 NotAction および "Effect": "Deny" を使用してタグ条件キーを使用すると、過度に許容される可能性があります。条件キーをサポートしていないサービスアクションでは、条件は無視されます。AWS では、ロジックを書き換えてアクションのリストを拒否することをお勧めします。

aws:ResourceTag 条件キーを NotAction とともに使用する場合、キーをサポートしない新規または既存のサービスアクションは拒否されません。AWS は、拒否するアクションを明示的にリスト

することをお勧めします。IAM Access Analyzer は、aws:ResourceTag 条件キーをサポートしないリストされたアクションに対して個別の結果を返します。詳細については、「[セキュリティ警告 — サービスに対してサポートされていないタグ条件キーで拒否](#)」を参照してください。

サービスが aws:ResourceTag 条件キーをサポートしている場合、タグを使用してそのサービスのリソースへのアクセスを制御できます。これは、[属性ベースのアクセス制御 \(ABAC\)](#) と呼ばれます。これらのキーをサポートしないサービスでは、[リソースベースのアクセス制御 \(RBAC\)](#) を使用してリソースへのアクセスを制御する必要があります。

## 関連用語

- [グローバル条件キー](#)
- [ABAC と RBAC の比較](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

## セキュリティ警告 — サービスプリンシパルへのアクセスを制限する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Restrict access to service principal: Granting access to a service principal without specifying a source is overly permissive. Use aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths condition key to grant fine-grained access.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Granting access to a service principal without specifying a source is overly permissive. Use aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths condition key to grant fine-grained access."
```

## セキュリティ警告を解決する

サービスの識別子であるサービスプリンシパルを使用して、リソースベースのポリシーの Principal エレメントで AWS のサービスを指定できます。あなたの代わりに実行できるアクセス権をサービスプリンシパルに付与する場合は、アクセス権を制限してください。aws:SourceArn、aws:SourceAccount、aws:SourceOrgID または

`aws:SourceOrgPaths` などの条件キーを使い、特定のリソース ARN、AWS アカウント、組織 ID、組織パスなど、特定のソースへのアクセスを制限することで、ポリシーに対し過剰にアクセス許可が付与されないようにすることができます。アクセスを制限することで、混乱した代理と呼ばれるセキュリティ上の問題を防ぐことができます。

## 関連用語

- [AWS のサービス プリンシパル](#)
- [AWS グローバル条件キー: `aws:SourceAccount`](#)
- [AWS グローバル条件キー: `aws:SourceArn`](#)
- [AWS グローバル条件キー: `aws:SourceOrgId`](#)
- [AWS グローバル条件キー: `aws:SourceOrgPaths`](#)
- [混乱する代理問題](#)

## セキュリティ警告 — oidc プリンシパルの条件キーがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing condition key for oidc principal: Using an Open ID Connect principal without a condition can be overly permissive. Add condition keys with a prefix that matches your federated OIDC principals to ensure that only the intended identity provider assumes the role.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using an Open ID Connect principal without a condition can be overly permissive. Add condition keys with a prefix that matches your federated OIDC principals to ensure that only the intended identity provider assumes the role."
```

## セキュリティ警告を解決する

Open ID Connect プリンシパルを条件なしで使用すると、過度に許容されることがあります。フェデレーション OIDC プリンシパルと一致するプレフィックスを含む条件キーを追加して、対象の ID プロバイダーのみがロールを引き受けるようにします。

## 関連用語

- [ウェブ ID または OpenID Connect フェデレーション用のロールの作成 \(コンソール\)](#)



## セキュリティ警告 — github リポジトリの条件キーがありません

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Missing github repo condition key: Granting a federated GitHub principal permissions without a condition key can allow more sources to assume the role than you intended. Add the token.actions.githubusercontent.com:sub condition key and specify the branch and repository name in the value.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Granting a federated GitHub principal permissions without a condition key can allow more sources to assume the role than you intended. Add the token.actions.githubusercontent.com:sub condition key and specify the branch and repository name in the value."
```

### セキュリティ警告を解決する

GitHub を OIDC IdP として使用する場合、ベストプラクティスは、IAM IdP に関連付けられたロールを引き受けることができるエンティティを制限することです。ロール信頼ポリシーに Condition ステートメントを含めると、ロールを特定の GitHub Organization、リポジトリ、またはブランチに制限できます。条件キー `token.actions.githubusercontent.com:sub` を使用してアクセスを制限できます。条件を特定のリポジトリまたはブランチのセットに制限することをお勧めします。この条件を含めない場合、管理外の Organization またはリポジトリの GitHub Actions は、AWS アカウントで GitHub IAM IdP に関連付けられたロールを引き受けることができます。

### 関連用語

- [GitHub OIDC ID プロバイダーのロールの設定](#)

### 提案 - 空の配列アクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array action: This statement includes no actions and does not affect the policy. Specify actions.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "This statement includes no actions and does not affect the policy. Specify actions."
```

## 提案の解決

ステートメントには Action または NotAction アクションのセットを含む要素を含める必要があります。エレメントが空の場合、ポリシーステートメントはパーミッションを提供しません。Action 要素にアクションを指定します。

- [IAM JSON ポリシーエレメント: アクション](#)

## 提案 - 空の配列条件

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array condition: There are no values for the condition key {{key}} and it does not affect the policy. Specify conditions.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "There are no values for the condition key {{key}} and it does not affect the policy. Specify conditions."
```

## 提案の解決

オプションの Condition 要素構造体を使用するには、条件演算子とキーと値のペアを使用する必要があります。条件値が空の場合、条件は true に設定され、ポリシーステートメントにはアクセス許可がありません。条件値を指定します。

- [IAM JSON ポリシーエレメント: 条件](#)

## 提案 – ForAllValues の空の配列条件

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array condition ForAllValues: The ForAllValues prefix with an empty condition key matches only if the key {{key}} is missing from the request context. To determine if
```

```
the request context is empty, we recommend that you use the Null condition operator with the value of true instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The ForAllValues prefix with an empty condition key matches only if the key {{key}} is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead."
```

## 提案の解決

Condition 要素構造体を使用するには、条件演算子とキーバリューのペアを使用する必要があります。ForAllValues セット演算子は、要求セットのすべてのメンバーの値が条件キーセットのサブセットであるかどうかをテストします。

空の条件キーで ForAllValues を使用すると、要求にキーがない場合にのみ条件が一致します。AWS では、要求コンテキストが空であるかどうかをテストする場合は、代わりに Null 条件演算子を使用することをお勧めします。

- [複数値のコンテキストキー](#)
- [条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## 提案 - 空の配列条件 ForAnyValue

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array condition ForAnyValue: The ForAnyValue prefix with an empty condition key {{key}} never matches the request context and it does not affect the policy. Specify conditions.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The ForAnyValue prefix with an empty condition key {{key}} never matches the request context and it does not affect the policy. Specify conditions."
```

## 提案の解決

Condition 要素構造体を使用するには、条件演算子とキーバリューのペアを使用する必要があります。ForAnyValues セット演算子は、要求値のセットの少なくとも1つのメンバーが条件キー値のセットの少なくとも1つのメンバーと一致するかどうかをテストします。

空の条件キーで ForAnyValues を使用すると、条件が一致することはありません。これは、ステートメントがポリシーに影響を与えないことを意味します。AWSでは、条件を書き換えることをお勧めします。

- [複数値のコンテキストキー](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## 提案 - 空の条件 IfExists

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array condition IfExists: The IfExists suffix with an empty condition key matches only if the key {{key}} is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The IfExists suffix with an empty condition key matches only if the key {{key}} is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead."
```

## 提案の解決

...IfExists 接尾辞は、条件演算子を編集します。これは、ポリシーキーがリクエストのコンテキストで存在する場合、ポリシーで指定されたとおりにキーを処理することを意味します。キーが存在しない場合、条件要素は true と評価されます。

空の条件キーで ...IfExists を使用すると、要求にキーがない場合にのみ条件が一致します。AWS では、リクエストコンテキストが空かどうかをテストする場合は、代わりに Null 条件演算子を使用することをお勧めします。

- [...IfExists 条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

### 提案 – 空の配列プリンシパル

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array principal: This statement includes no principals and does not affect the policy. Specify principals.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "This statement includes no principals and does not affect the policy. Specify principals."
```

### 提案の解決

IAM ロールの信頼ポリシーおよびリソースベースのポリシーで Principal または NotPrincipal 要素を使用する必要があります。リソースベースのポリシーは、リソースに直接埋め込むポリシーです。

ステートメントの Principal 要素ではステートメントがポリシーに影響しません。AWS では、リソースにアクセスできるプリンシパルを指定することをお勧めします。

- [IAM JSON ポリシーエレメント: プリンシパル](#)
- [IAM JSON ポリシーエレメント: 非プリンシパル](#)

### 提案 — 空の配列リソース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty array resource: This statement includes no resources and does not affect the policy. Specify resources.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "This statement includes no resources and does not affect the policy. Specify resources."
```

## 提案の解決

ステートメントには、Resource または NotResource 要素を含める必要があります。

ステートメントの resource 要素に空の配列を指定すると、そのステートメントはポリシーに影響を与えません。AWS では、リソースに Amazon リソースネーム (ARN) を指定することをお勧めします。

- [IAM JSON ポリシーエレメント: リソース](#)
- [IAM JSON ポリシーエレメント: 非リソース](#)

## 提案 — 空のオブジェクト条件

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty object condition: This condition block is empty and it does not affect the policy. Specify conditions.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "This condition block is empty and it does not affect the policy. Specify conditions."
```

## 提案の解決

Condition 要素構造体を使用するには、条件演算子とキーと値のペアを使用する必要があります。

ステートメントの condition 要素に空のオブジェクトを指定すると、そのステートメントはポリシーに影響を与えません。オプションの要素を削除するか、条件を指定します。

- [IAM JSON ポリシーエレメント: 条件](#)

## 提案 — 空のオブジェクト主体

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

Empty object principal: This statement includes no principals and does not affect the policy. Specify principals.

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "This statement includes no principals and does not affect the policy. Specify principals."
```

### 提案の解決

IAMロールの信頼ポリシーおよびリソースベースのポリシーで Principal または NotPrincipal 要素を使用する必要があります。リソースベースのポリシーは、リソースに直接埋め込むポリシーです。

ステートメントの Principal 要素ではステートメントがポリシーに影響しません。AWS では、リソースにアクセスできるプリンシパルを指定することをお勧めします。

- [IAM JSON ポリシーエレメント: プリンシパル](#)
- [IAM JSON ポリシーエレメント: 非プリンシパル](#)

### 提案 — 空の Sid 値

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Empty Sid value: Add a value to the empty string in the Sid element.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Add a value to the empty string in the Sid element."
```

### 提案の解決

オプションの Sid (statement ID) 要素を使用すると、ポリシーステートメントに指定する識別子を入力できます。Sid 値は、ステートメント配列内の各ステートメントに割り当てることができます。使用することを選択した場合、Sid 要素では文字列値を指定する必要があります。

### 関連用語

- [IAM JSON ポリシーエレメント: Sid](#)

### 提案 — IP 範囲の改善

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Improve IP range: The non-zero bits in the IP address after the masked bits are ignored. Replace address with {{addr}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The non-zero bits in the IP address after the masked bits are ignored. Replace address with {{addr}}."
```

### 提案の解決

IPアドレス条件は、203.0.113.0/24 または 2001:DB8:1234:5678::/64 などの標準的な CIDR 形式である必要があります。マスクされたビットの後にゼロ以外のビットを含めると、条件では考慮されません。AWSでは、メッセージに含まれている新しいアドレスを使用することをお勧めします。

- [IP アドレス条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

### 提案 - 修飾子付き NULL

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Null with qualifier: Avoid using the Null condition operator with the ForAllValues or ForAnyValue qualifiers because they always return a true or false respectively.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Avoid using the Null condition operator with the ForAllValues or ForAnyValue qualifiers because they always return a true or false respectively."
```

### 提案の解決



Condition 要素では、条件演算子 (等しい、より小さい、など) を使用して、ポリシーの条件をリクエストコンテキストのキーと値と比較する式を作成します。単一の条件キーに複数の値を含むリクエストの場合は、ForAllValues または ForAnyValue の集合演算子を使用する必要があります。

Null で ForAllValues 条件演算子を使用すると、ステートメントは常に true を返します。Null で ForAnyValue 条件演算子を使用すると、ステートメントは常に false を返します。AWSでは、これらの集合演算子とともに StringLike 条件演算子を使用することをお勧めします。

## 関連用語

- [複数值のコンテキストキー](#)
- [Null 条件演算子](#)
- [条件の要素](#)

## 提案 — プライベート IP アドレスのサブセット

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Private IP address subset: The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses will not have the desired effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses will not have the desired effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp."
```

## 提案の解決

グローバル条件キーは、aws:SourceIp パブリック IP アドレス範囲に対してのみ機能します。

Condition 要素にプライベートIPアドレスとパブリックIPアドレスが混在している場合、ステートメントが目的の効果を発揮しない可能性があります。aws:VpcSourceIP を使用してプライベートIPアドレスを指定できます。

**Note**

グローバル条件キー `aws:VpcSourceIp` では、リクエストが指定された IP アドレスから送信され、VPC エンドポイントを経由する場合にのみキーが一致します。

- [aws:SourceIp グローバル条件キー](#)
- [aws:VpcSourceIp グローバル条件キー](#)
- [IP アドレス条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

**提案 — プライベート通知アドレスのサブセット**

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Private NotIpAddress subset: The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses have no effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses have no effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp."
```

**提案の解決**

グローバル条件キーは、`aws:SourceIp` パブリック IP アドレス範囲に対してのみ機能します。

Condition 要素に `NotIpAddress` 条件演算子が含まれ、プライベート IP アドレスとパブリック IP アドレスが混在している場合、ステートメントが目的の効果を発揮しない可能性があります。ポリシーで指定されていないパブリック IP アドレスはすべて一致します。プライベート IP アドレスは一致しません。この効果を実現するには、`NotIpAddress` と `aws:VpcSourceIp` を併用して、一致しないプライベート IP アドレスを指定します。

- [aws:SourceIp グローバル条件キー](#)

- [aws:VpcSourceIp グローバル条件キー](#)
- [IP アドレス条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

## 提案 — 冗長なアクション

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Redundant action: The {{redundantActionCount}} action(s) are redundant because they provide similar permissions. Update the policy to remove the redundant action such as: {{redundantAction}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{redundantActionCount}} action(s) are redundant because they provide similar permissions. Update the policy to remove the redundant action such as: {{redundantAction}}."
```

## 提案の解決

Action 要素でワイルドカード (\*) を使用する場合、冗長なアクセス許可を含めることができます。AWS では、ポリシーを確認し、必要なアクセス許可のみを含めることをお勧めします。これは、冗長なアクションを削除するのに役立ちます。

たとえば、次のアクションには2回の iam:GetCredentialReport アクションが含まれます。

```
"Action": [  
    "iam:Get*",  
    "iam:List*",  
    "iam:GetCredentialReport"  
],
```

この例では、Get または List で始まるすべての IAM アクションに対してアクセス許可が定義されています。IAM が追加の取得またはリスト操作を追加する場合、このポリシーによって許可されます。これらの読み取り専用アクションをすべて許可することをお勧めします。iam:GetCredentialReport アクションはすでに iam:Get\* の一部として含まれています。重複したアクセス許可を削除するには、iam:GetCredentialReport を削除します。

アクションの内容がすべて冗長である場合、このポリシーチェックの結果が表示されます。この例では、要素に `iam:*CredentialReport` が含まれている場合、冗長とは見なされません。これには、冗長な `iam:GetCredentialReport` とそうでない `iam:GenerateCredentialReport` が含まれます。`iam:Get*` または `iam:*CredentialReport` のいずれかを削除すると、ポリシーのアクセス許可が変更されます。

## • [IAM JSON ポリシーエレメント: アクション](#)

この提案を伴うAWS 管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

冗長アクションは、ポリシーで付与されるアクセス権限には影響しません。顧客管理ポリシーを作成するための参照として AWS 管理ポリシーを使用する場合、AWS では、ポリシーから冗長なアクションを削除することをお勧めします。

提案 — 冗長な条件値 num

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Redundant condition value num: Multiple values in {{operator}} are redundant. Replace with the {{greatest/least}} single value for {{key}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Multiple values in {{operator}} are redundant. Replace with the {{greatest/least}} single value for {{key}}."
```

提案の解決

条件キーで類似の値に数値条件演算子を使用すると、重複を作成して重複する権限を作成できます。

たとえば、次の Condition 要素には、年の重なりが1200秒である複数の `aws:MultiFactorAuthAge` 条件が含まれています。

```
"Condition": {
  "NumericLessThan": {
    "aws:MultiFactorAuthAge": [
```

```
        "2700",
        "3600"
    ]
}
}
```

この例では、多要素認証 (MFA) が 3600 秒 (1 時間) 前に完了した場合にアクセス許可が定義されます。冗長な 2700 値を削除できます。

- [数値条件演算子](#)
- [IAM JSON ポリシーエレメント: 条件](#)

### 提案 — 冗長リソース

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Redundant resource: The {{redundantResourceCount}} resource ARN(s) are redundant because they reference the same resource. Review the use of wildcards (*)
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The {{redundantResourceCount}} resource ARN(s) are redundant because they reference the same resource. Review the use of wildcards (*)"
```

### 提案の解決

Amazon リソースネーム (ARN) でワイルドカード (\*) を使用すると、冗長リソースアクセス権限を作成できます。

たとえば、次の Resource 要素には、冗長なアクセス許可を持つ複数の ARN が含まれます。

```
"Resource": [
    "arn:aws:iam::111122223333:role/jane-admin",
    "arn:aws:iam::111122223333:role/jane-s3only",
    "arn:aws:iam::111122223333:role/jane*"
],
```

この例では、名前が jane で始まるすべてのロールに対して権限が定義されています。結果の権限を変更せずに、冗長な jane-admin および jane-s3only ARN を削除できます。これにより、ポリ

シーはダイナミックになります。これは、で始まる将来のロールに対するパーミッションを定義します jane。ポリシーの意図が静的数のロールへのアクセスを許可する場合は、最後の ARN を削除し、定義する必要がある ARN のみをリストします。

- [IAM JSON ポリシーエレメント: リソース](#)

この提案を伴う AWS 管理ポリシー

[AWS 管理ポリシー](#)を使用すると、一般的な AWS のユースケースに基づいてアクセス許可を割り当てることにより、AWS の使用を開始できます。

冗長リソースは、ポリシーで付与されるアクセス許可には影響しません。AWS マネージドポリシーを参照として使用してカスターマネージドポリシーを作成する場合、AWS ではポリシーから冗長なリソースを削除することをお勧めします。

提案 — 冗長ステートメント

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Redundant statement: The statements are redundant because they provide identical permissions. Update the policy to remove the redundant statement.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The statements are redundant because they provide identical permissions. Update the policy to remove the redundant statement."
```

提案の解決

Statement 要素は、ポリシーの主要要素です。この要素は必須です。Statement 要素には、単一のステートメントまたは個々のステートメントの配列を含めることができます。

長いポリシーに同じステートメントを複数回含めると、ステートメントは冗長になります。ポリシーによって付与されるアクセス許可に影響を与えずに、ステートメントの 1 つを削除できます。ポリシーを編集したユーザーは、重複したステートメントを更新せずにステートメントの 1 つを変更することがあります。これにより、意図した権限よりも多くなる可能性があります。

- [IAM JSON ポリシーエレメント: ステートメント](#)

## 提案 — サービス名のワイルドカード

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Wildcard in service name: Avoid using wildcards (*, ?) in the service name because it might grant unintended access to other AWS services with similar names.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Avoid using wildcards (*, ?) in the service name because it might grant unintended access to other AWS services with similar names."
```

### 提案の解決

ポリシーに AWS サービスの名前を含める場合は、AWS では、ワイルドカード (\*、?) を含めないことをお勧めします。これにより、意図しない将来のサービスに対するアクセス許可が追加される可能性があります。たとえば、名前に単語 `*code*` が含まれている AWS サービスは12以上あります

```
"Resource": "arn:aws:*code*::111122223333:*"
```

- [IAM JSON ポリシーエレメント: リソース](#)

## 提案 — サービスに対してサポートされていないタグ条件キーで許可する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Allow with unsupported tag condition key for service: Using the effect Allow with the tag condition key {{conditionKeyName}} and actions for services with the following prefixes does not affect the policy: {{serviceNames}}. Actions for the listed service are not allowed by this statement. We recommend that you move these actions to a different statement without this condition key.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the effect Allow with the tag condition key {{conditionKeyName}} and actions for services with the following prefixes does not affect the policy: {{serviceNames}}. Actions for the listed service are not allowed"
```

by this statement. We recommend that you move these actions to a different statement without this condition key."

## 提案の解決

ポリシーの Condition 要素でサポートされていないタグ条件キーを "Effect": "Allow" で使用しても、そのサービスアクションでは条件が無視されるため、ポリシーによって付与されるアクセス許可には影響しません。AWS では、条件キーをサポートしていないサービスのアクションを削除し、そのサービス内の特定のリソースへのアクセスを許可する別のステートメントを作成することをお勧めします。

aws:ResourceTag 条件キーを使用していて、サービスアクションでサポートされていない場合、そのキーは要求コンテキストに含まれません。この場合は、Allow ステートメントは、常に false に設定され、アクションは決して許可されません。これは、リソースが正しくタグ付けされている場合でも発生します。

サービスが aws:ResourceTag 条件キーをサポートしている場合、タグを使用してそのサービスのリソースへのアクセスを制御できます。これは、[属性ベースのアクセス制御 \(ABAC\)](#) と呼ばれます。これらのキーをサポートしないサービスでは、[リソースベースのアクセス制御 \(RBAC\)](#) を使用してリソースへのアクセスを制御する必要があります。

### Note

一部のサービスでは、リソースとアクションのサブセットの aws:ResourceTag 条件キーのサポートが許可されています。IAM Access Analyzer は、サポートされていないサービスアクションの結果を返します。たとえば、Amazon S3はそのリソースのサブセットに対して aws:ResourceTag をサポートしています。aws:ResourceTag 条件キーをサポートする Amazon S3 で利用可能なすべてのリソースタイプを表示するには、『サービス認証リファレンス』の[Amazon S3 で定義されるリソースタイプ](#)を参照してください。

たとえば、チームメンバーがキーと値のペア team=BumbleBee でタグ付けされた特定のリソースの詳細を表示できるようにするとします。また、AWS Lambda でリソースにタグを付けることはできますが、aws:ResourceTag 条件キーをサポートしていないと仮定します。このタグが存在する場合に AWS App Mesh および AWS Backup のビューアクションを許可するには、aws:ResourceTag 条件キーを使用します。Lambda の場合は、チーム名をプレフィックスとして含むリソース命名規則を使用します。その後、その命名規則でリソースを表示することを許可する別のステートメントを含めます。



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewSupported",
      "Effect": "Allow",
      "Action": [
        "appmesh:DescribeMesh",
        "backup:GetBackupPlan"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/team": "BumbleBee"
        }
      }
    },
    {
      "Sid": "AllowViewUnsupported",
      "Effect": "Allow",
      "Action": "lambda:GetFunction",
      "Resource": "arn:aws:lambda:*:123456789012:function:team-BumbleBee*"
    }
  ]
}
```

### ⚠ Warning

この結果の回避策として、Not [バージョンの条件演算子](#)を "Effect": "Allow" と一緒に使用しないでください。これらの条件演算子は、否定的なマッチングを提供します。これは、条件が評価された後、結果が否定されることを意味します。前の例では、lambda:GetFunction 演算子を使用して AllowViewSupported ステートメントに StringNotEquals アクションを含めると、リソースがタグ付けされているかどうかに関係なく、常にアクションが許可されます。

この結果の回避策として、条件演算子の...[IfExists](#) バージョンを使用しないでください。これは、キーがリクエストコンテキストに存在し、値が一致する場合、アクションを許可することを意味します。それ以外の場合は、アクションを許可します。前の例では、演算子を使用して AllowViewSupported ステートメントに lambda:GetFunction アクションを含めると、常に StringEqualsIfExists アクションが許可されます。そのアクションでは、

キーはコンテキストに存在せず、リソースがタグ付けされているかどうかにかかわらず、そのリソースタイプを表示しようとするすべての試みが許可されます。

## 関連用語

- [グローバル条件キー](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

提案 — サービス用にサポートされていないタグ条件キーで NotAction を許可する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Allow NotAction with unsupported tag condition key for service: Using the effect Allow with NotAction and the tag condition key {{conditionKeyName}} allows only service actions that support the condition key. The condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "Using the effect Allow with NotAction and the tag condition key {{conditionKeyName}} allows only service actions that support the condition key. The condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction."
```

## 提案の解決

ポリシーの Condition 要素でサポートされていないタグ条件キーを要素 NotAction および "Effect": "Allow" と併用しても、ポリシーによって付与されるアクセス許可には影響しません。条件キーをサポートしていないサービスアクションでは、条件は無視されます。AWS では、アクションのリストを許可するようにロジックを書き直すことを推奨しています。

aws:ResourceTag 条件キーを NotAction と併用する場合、キーをサポートしない新規または既存のサービスアクションは許可されません。AWS では、許可するアクションを明示的にリストすることをお勧めします。IAM Access Analyzer は、aws:ResourceTag 条件キーをサポートしないリス

トされたアクションに対して個別の結果を返します。詳細については、「[提案 — サービスに対してサポートされていないタグ条件キーで許可する](#)」を参照してください。

サービスが `aws:ResourceTag` 条件キーをサポートしている場合、タグを使用してそのサービスのリソースへのアクセスを制御できます。これは、[属性ベースのアクセス制御 \(ABAC\)](#) と呼ばれます。これらのキーをサポートしないサービスでは、[リソースベースのアクセス制御 \(RBAC\)](#) を使用してリソースへのアクセスを制御する必要があります。

## 関連用語

- [グローバル条件キー](#)
- [ABAC と RBAC の比較](#)
- [IAM JSON ポリシーエレメント: 条件演算子](#)
- [条件の要素](#)
- [JSON ポリシー概要](#)

## 提案 — サービスプリンシパルで推奨される条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Recommended condition key for service principal: To restrict access to the service principal {{servicePrincipalPrefix}} operating on your behalf, we recommend aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of {{key}}.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "To restrict access to the service principal {{servicePrincipalPrefix}} operating on your behalf, we recommend aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of {{key}}."
```

## 提案の解決

サービスの識別子であるサービスプリンシパルを使用して、リソースベースのポリシーの Principal エレメントで AWS のサービスを指定できます。サービスプリンシパルへのアクセスを付与する場合は、`aws:Referer` のような条件キーの代わりに、`aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID`、`aws:SourceOrgPaths` な

どの条件キーを使用してください。これにより、混乱した代理と呼ばれるセキュリティ上の問題を防ぐことができます。

## 関連用語

- [AWS のサービス プリンシパル](#)
- [AWS グローバル条件キー: aws:SourceAccount](#)
- [AWS グローバル条件キー: aws:SourceArn](#)
- [AWS グローバル条件キー: aws:SourceOrgId](#)
- [AWS グローバル条件キー: aws:SourceOrgPaths](#)
- [混乱する代理問題](#)

## 提案 — ポリシー内の不適切な条件キー

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Irrelevant condition key in policy: The condition key {{condition-key}} is not relevant for the {{resource-type}} policy. Use this key in an identity-based policy to govern access to this resource.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The condition key {{condition-key}} is not relevant for the {{resource-type}} policy. Use this key in an identity-based policy to govern access to this resource."
```

## 提案の解決

一部の条件キーは、リソースベースのポリシーに適していない場合があります。例えば、s3:ResourceAccount 条件キーは Amazon S3 バケットや Amazon S3 アクセスポイントのリソースタイプに添付された、リソースベースのポリシーには適していません。

ID ベースのポリシーでは条件キーを使用して、リソースへのアクセスを制御する必要があります。

## 関連用語

- [アイデンティティベースおよびリソースベースのポリシー](#)

## 提案 — ロール信頼ポリシーにおける冗長なプリンシパル

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Redundant principal in role trust policy: The assumed-role principal
  {{redundant_principal}} is redundant with its parent role {{parent_role}}. Remove the
  assumed-role principal.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The assumed-role principal {{redundant_principal}} is redundant with
  its parent role {{parent_role}}. Remove the assumed-role principal."
```

### 提案の解決

ポリシーの Principal 要素で、継承されたロールのプリンシパルとその親の役ロールの両方を指定した場合、異なる許可は許可または拒否されません。例えば、次の形式を使用して Principal 要素を指定すると冗長になります。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::AWS-account-ID:role/rolename",
    "arn:aws:iam::AWS-account-ID:assumed-role/rolename/rolesessionname"
  ]
}
```

ロールを引き受けたプリンシパルを削除することをお勧めします。

### 関連用語

- [ロールセッションプリンシパル](#)

## 提案 — 対象者のクレームタイプを確認する

AWS Management Console では、このチェックの結果に次のメッセージが表示されます。

```
Confirm audience claim type: The 'aud' (audience) claim key identifies the recipients
  that the JSON web token is intended for. Audience claims can be multivalued or single-
  valued. If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If
  the claim is single-valued, do not use a qualifier.
```

AWS CLI または AWS API へのプログラムによる呼び出しでは、このチェックの結果に次のメッセージが表示されます。

```
"findingDetails": "The 'aud' (audience) claim key identifies the recipients that the JSON web token is intended for. Audience claims can be multivalued or single-valued. If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If the claim is single-valued, do not use a qualifier."
```

## 提案の解決

aud (対象者) クレーム キーは、アプリを IdP に登録するときに発行されるアプリの一意の識別子であり、JSON ウェブトークンの対象となる受信者を識別します。対象者のクレームは、複数值でも単一値でもかまいません。クレームが複数值の場合は、ForAllValues または ForAnyValue 条件セット演算子を使用します。クレームが単一値の場合は、条件セット演算子を使用しないでください。

## 関連用語

- [ウェブ ID または OpenID Connect フェデレーション用のロールの作成 \(コンソール\)](#)
- [複数值のコンテキストキー](#)
- [単一値と複数值の条件キー](#)

## IAM Access Analyzer カスタムポリシーチェック

AWS Identity and Access Management Access Analyzer カスタムポリシーチェックを使用して、指定したセキュリティ標準に照らしてポリシーを検証できます。次の種類のカスタムポリシーチェックを実行できます。

- 参照ポリシーに対するチェック: ポリシーを編集するときに、更新したポリシーで新しいアクセス権限が付与されるかどうかを、参照ポリシー (既存のバージョンのポリシーなど) との比較で確認できます。このチェックは、AWS Command Line Interface (AWS CLI)、IAM Access Analyzer API (API)、または IAM コンソールの JSON ポリシーエディタを使用してポリシーを編集するときに実行できます。
- IAM アクションまたはリソースのリストに対するチェック: 特定の IAM アクションまたはリソースがポリシーで許可されていないかどうかを確認できます。アクションのみを指定した場合、IAM Access Analyzer はポリシー内のすべてのリソースに対するアクションのアクセスをチェックします。リソースのみを指定した場合、IAM Access Analyzer は、どのアクションで指定したリソー

スにアクセスできるかをチェックします。アクションとリソースの両方を指定した場合、IAM Access Analyzer は、指定したアクションのどれで指定したリソースにアクセスできるかをチェックします。このチェックは、AWS CLI または API を使用してポリシーを作成または編集するときに実行できます。

- パブリックアクセスのチェック: リソースポリシーで指定したリソースタイプへのパブリックアクセスを許可できるかどうかを確認できます。このチェックは、AWS CLI または API を使用してポリシーを作成または編集するときに実行できます。この種類のカスタムポリシーチェックは、チェックにアカウントまたは外部アクセスアナライザーコンテキストを必要としないため、[アクセスのプレビュー](#)とは異なります。アクセスプレビューでは、リソースに対するアクセス許可をデプロイする前に IAM Access Analyzer の検出結果をプレビューできます。一方、カスタムチェックでは、ポリシーによってパブリックアクセスが許可されているかどうかを判別します。

料金は、各カスタムポリシーチェックに関連付けられます。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

## カスタムポリシーチェックの仕組み

カスタムポリシーチェックは、ID ベースおよびリソースベースのポリシーに対して実行できます。カスタムポリシーチェックでは、新しいアクセスまたは指定されたアクセスがポリシーによって許可されているかどうかを判別する際に、パターンマッチング技術やアクセスログの検証を使用しません。外部アクセスの検出結果と同様に、カスタムポリシーチェックは [Zelkova](#) 上に構築されています。Zelkova は、IAM ポリシーを同等の論理ステートメントに変換し、問題に対して汎用および特殊な論理ソルバー (充足可能性モジュロ理論) のスイートを実行します。新しいアクセスや指定されたアクセスをチェックするために、IAM Access Analyzer は Zelkova をポリシーに繰り返し適用します。クエリは、ポリシーの内容に基づいてポリシーが許可する動作のクラスを特徴付けるために、ますます具体的になります。充足可能性モジュロ理論の詳細については、「[Satisfiability Modulo Theories](#)」を参照してください。

まれに、IAM Access Analyzer は、ポリシーステートメントが新しいアクセスや指定されたアクセスを許可できるかどうかを完全に判別できない場合があります。このような場合は、カスタムポリシーチェックに失敗することによって、安全のために、誤りであっても陽性を宣言すること (偽陽性) を選択します。IAM Access Analyzer は、包括的なポリシー評価を提供するように設計されており、偽陰性が最小限に抑えられます。このアプローチにより、IAM Access Analyzer では、チェックに合格した場合はポリシーによってアクセスが許可されなかったことを意味するということが高度に保証されます。IAM Access Analyzer からのレスポンスで報告されたポリシーステートメントを確認することで、失敗したチェックを手動で調べることができます。



## 新しいアクセスをチェックするための参照ポリシーの例

GitHub の「[IAM Access Analyzer custom policy checks samples](#)」リポジトリで、参照ポリシーの例を入手し、新しいアクセス用のカスタムポリシーチェックをセットアップして実行する方法を学習できます。

### 📌 これらの例を使用する前に

これらのサンプルリファレンスポリシーを使用する前に、次の手順を実行してください。

- 固有要件に応じて、参照ポリシーを慎重にレビューしてカスタマイズします。
- 使用する AWS のサービスが含まれる環境で、参照ポリシーを十分にテストします。

参照ポリシーは、カスタムポリシーチェックの実装と使用を示します。これらの例は、公式な AWS 推奨事項やベストプラクティスとして解釈されることを意図したものではありません。お客様の環境のセキュリティ要件を解決するために適切であるかどうかについては、お客様の責任で参照ポリシーを慎重にテストしてください。

- カスタムポリシーチェックの分析は環境に依存しません。この分析では、入力ポリシーに含まれる情報のみが考慮されます。例えば、カスタムポリシーチェックでは、アカウントが特定の AWS 組織のメンバーであるかどうかは確認できません。そのため、カスタムポリシーチェックでは、[aws:PrincipalOrgId](#) 条件キーおよび [aws:PrincipalAccount](#) 条件キーの値に基づいて新しいアクセスを比較することはできません。

## 失敗したカスタムポリシーチェックの検査

カスタムポリシーチェックが失敗した場合、IAM Access Analyzer からのレスポンスには、チェックが失敗した原因となったポリシーステートメントの [ステートメント ID \(Sid\)](#) が含まれます。ステートメント ID はオプションのポリシー要素ですが、すべてのポリシーステートメントにステートメント ID を追加することが推奨されます。カスタムポリシーチェックでは、チェックが失敗した理由を特定するのに役立つステートメントインデックスも返されます。ステートメントインデックスはゼロベースの番号付けに従い、最初のステートメントは 0 として参照されます。チェックが失敗する原因となったステートメントが複数ある場合、チェックでは一度に 1 つのステートメント ID のみが返されます。理由の中で強調表示されているステートメントを修正し、合格するまでチェックを再実行することを推奨します。



## カスタムポリシーチェックによるポリシーの検証 (コンソール)

オプションのステップとして、IAM コンソールの JSON ポリシーエディタでポリシーを編集するときに、カスタムポリシーチェックを実行できます。更新したポリシーで新しいアクセス権限が付与されるかどうかを、既存のバージョンとの比較で確認できます。

### IAM JSON ポリシーの編集時に新しいアクセスをチェックする方法

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーのリストで、編集するポリシーのポリシー名を選択します。検索ボックスを使用して、ポリシーのリストをフィルタリングできます。
4. [アクセス許可] タブを選択し、[編集] を選択します。
5. [JSON] オプションを選択し、ポリシーを更新します。
6. ポリシーの下にあるポリシー検証ペインで、[新しいアクセス権限の確認] タブを選択し、[ポリシーの確認] を選択します。変更したアクセス許可によって新しいアクセス権限が付与されると、ポリシー検証ペインでそのステートメントが強調表示されます。
7. 新しいアクセス権限を付与する予定がない場合は、ポリシーステートメントを更新し、新しいアクセスが検出されなくなるまで [ポリシーの確認] を選択します。

#### Note

料金は、新しいアクセスに対する各チェックに関連付けられます。価格設定の詳細については、「[IAM Access Analyzer pricing](#)」を参照してください。

8. [Next] を選択します。
9. [確認と保存] ページで [このポリシーで定義されている許可] を確認して、[変更の保存] を選択します。

## カスタムポリシーチェックによるポリシーの検証 (AWS CLI または API)

AWS CLI または IAM Access Analyzer API から IAM Access Analyzer のカスタムポリシーチェックを実行できます。

## IAM Access Analyzer のカスタムポリシーチェックを実行する方法 (AWS CLI)

- 既存のポリシーと比較して、更新されたポリシーで新しいアクセスが許可されるかどうかを確認するには、[check-no-new-access](#) コマンドを実行します。
- 指定したアクセスがポリシーによって許可されていないかどうかを確認するには、[check-access-not-granted](#) コマンドを実行します。
- リソースポリシーで指定したリソースタイプへのパブリックアクセスを許可できるかどうかを確認するには、コマンド [check-no-public-access](#) を実行します。

## IAM Access Analyzer のカスタムポリシーチェックを実行する方法 (API)

- 既存のポリシーと比較して、更新されたポリシーで新しいアクセスが許可されるかどうかを確認するには、[CheckNoNewAccess](#) API オペレーションを使用します。
- 指定したアクセスがポリシーによって許可されていないかどうかを確認するには、[CheckAccessNotGranted](#) API オペレーションを使用します。
- リソースポリシーで指定したリソースタイプへのパブリックアクセスを許可できるかどうかを確認するには、[CheckNoPublicAccess](#) API オペレーションを使用します。

## IAM Access Analyzer ポリシーの生成

管理者またはデベロッパーは、IAM エンティティ (ユーザーまたはロール) に必要な権限を超えるアクセス許可を付与することがあります。IAM には、付与するアクセス許可を絞り込むのに役立つオプションがいくつか用意されています。1つのオプションは、エンティティのアクセスアクティビティに基づく IAM ポリシーを生成することです。IAM Access Analyzer は AWS CloudTrail ログを確認し、指定した日付範囲内のロールが使用したアクセス許可を含むポリシーテンプレートを生成します。テンプレートを使用して、特定のユースケースをサポートするために必要なアクセス許可のみを付与する、きめ細かなアクセス許可を持つポリシーを作成できます。

### トピック

- [ポリシー生成の仕組み](#)
- [サービスレベルとアクションレベルの情報](#)
- [ポリシーの生成について知っておくべきこと](#)
- [ポリシーの生成に必要なアクセス許可](#)
- [CloudTrail アクティビティに基づくポリシーの生成 \(コンソール\)](#)
- [別のアカウントの AWS CloudTrail データを使用してポリシーを生成する](#)

- [CloudTrail アクティビティに基づくポリシーの生成 \(AWS CLI\)](#)
- [CloudTrail アクティビティに基づいたポリシーの生成 \(AWS API\)](#)
- [IAM Access Analyzer のポリシー生成サービス](#)

## ポリシー生成の仕組み

IAM Access Analyzer は CloudTrail イベントを分析して、IAM エンティティ (ユーザーまたはロール) が使用しているアクションとサービスを特定します。次に、そのアクティビティに基づく IAM ポリシーを生成します。エンティティにアタッチされていた広範なアクセス許可ポリシーを生成されたポリシーに置き換えると、エンティティのアクセス許可を絞り込むことができます。次に、ポリシー生成プロセスの概要を示します。

- **ポリシーテンプレート生成のための設定** – IAM Access Analyzer で AWS CloudTrail イベントの履歴を分析するために、最大 90 日間の期間を指定します。既存のサービスロールを指定するか、新しいサービスロールを作成する必要があります。サービスロールは、IAM Access Analyzer が CloudTrail 証跡とサービスの最終アクセス情報にアクセスして、使用されたサービスとアクションを識別できるようにします。ポリシーを生成する前に、アカウントのイベントをログに記録する CloudTrail 証跡を指定する必要があります。CloudTrail データの IAM Access Analyzer クォータの詳細については、「[IAM Access Analyzer クォータ](#)」を参照してください。
- **ポリシーの生成** – IAM Access Analyzer は CloudTrail イベントのアクセスアクティビティに基づいてポリシーを生成します。
- **ポリシーの確認とカスタマイズ** – ポリシーが生成された後、指定した日付範囲内にエンティティが使用したサービスとアクションを確認できます。ポリシーをさらにカスタマイズするには、アクセス許可を追加または削除し、リソースを指定し、ポリシーテンプレートに条件を追加します。
- **ポリシーの作成とアタッチ** – マネージドポリシーを作成して、生成されたポリシーを保存するオプションがあります。作成したポリシーは、ポリシーの生成に使用されたアクティビティを実行したユーザーまたはロールにアタッチできます。

## サービスレベルとアクションレベルの情報

IAM Access Analyzer が IAM ポリシーを生成したとき、ポリシーをさらにカスタマイズするための情報を返します。ポリシーが生成されたとき、次の 2 つのカテゴリの情報を返すことができます。

- **アクションレベルの情報を持つポリシー** – Amazon EC2 など一部の AWS サービスでは、IAM Access Analyzer によって、CloudTrail イベント内のアクションを識別し、生成されたポリシーで使用されるアクションを一覧表示することができます。サポートされているサービスのリストにつ

いては、「[IAM Access Analyzer のポリシー生成サービス](#)」を参照してください。一部のサービスでは、IAM Access Analyzer によって、生成されたポリシーにサービスのアクションを追加するように求められます。

- サービスレベル情報を持つポリシー – IAM Access Analyzer は[最終アクセス](#)情報を使用して、最近使用したすべてのサービスを含むポリシーテンプレートを作成します。AWS Management Console を使用する場合は、サービスを確認し、ポリシーを完了するためのアクションを追加するように求められます。

各サービスのアクションのリストについては、『サービス認証リファレンス』の「[AWS のアクション、リソース、条件キーサービス](#)」を参照してください。

## ポリシーの生成について知っておくべきこと

ポリシーを生成する前に、以下に示す重要な点を確認します。

- CloudTrail 証跡を有効にする – アクセスアクティビティに基づいてポリシーを生成するには、自分のアカウントで CloudTrail 証跡を有効にする必要があります。CloudTrail 証跡を作成すると、CloudTrail は証跡に関連するイベントを、指定した Amazon S3 バケットに送信します。CloudTrail 証跡の作成方法については、「AWS CloudTrail ユーザーガイド」の「[AWS アカウントの証跡の作成](#)」を参照してください。
- データイベントを使用できません – IAM Access Analyzer は、生成されたポリシーで、Amazon S3 データイベントなどのデータイベントのアクションレベルのアクティビティを識別しません。
- PassRole – iam:PassRole アクションは CloudTrail によって追跡されず、生成されたポリシーには含まれません。
- ポリシー生成時間を短縮する – ポリシーをより速く生成するには、ポリシー生成の設定時に指定する日付範囲を短くします。
- 監査に CloudTrail を使用する – 監査の目的には、ポリシー生成を使用せず、CloudTrail を使用してください。CloudTrail の使用に関する詳細については、「[AWS STS による IAM および AWS CloudTrail の API コールのログ記録](#)」を参照してください。
- 拒否されたアクション – ポリシー生成により、拒否されたアクションを含むすべての CloudTrail イベントが確認されます。
- IAM コンソールには 1 つのポリシー – IAM コンソールには、一度に 1 つのポリシーを生成できません。
- IAM コンソールに生成されたポリシーの可用性 – 生成されたポリシーは、生成後最大 7 日間 IAM コンソールで確認できます。7 日後に、新しいポリシーを生成する必要があります。

- ポリシー生成クォータ – IAM Access Analyzer ポリシー生成クォータの詳細については、「[IAM Access Analyzer クォータ](#)」を参照してください。
- Amazon S3 の標準料金が適用されます — ポリシー生成機能を使用する場合、IAM Access Analyzer は S3 バケット内の CloudTrail ログを確認します。ポリシー生成のために CloudTrail ログにアクセスするための追加ストレージ料金はかかりません。AWS は S3 バケットに保存されている CloudTrail ログのリクエストとデータ転送に対して、Amazon S3 の標準料金を課金します。
- AWS Control Tower サポート - ポリシー生成はポリシーを生成するための AWS Control Tower をサポートしていません。

## ポリシーの生成に必要なアクセス許可

初めてポリシーを生成するために必要なアクセス許可は、その後の使用のためにポリシーを生成するために必要なアクセス許可とは異なります。

### 初回の設定

ポリシーを初めて生成するときは、既存の適切な[サービスロール](#)をアカウントで選択するか、新しいサービスロールを作成する必要があります。サービスロールは、IAM Access Analyzer が CloudTrail とアカウント内で最後にアクセスされたサービスの情報にアクセスできるようにします。ロールの作成と設定に必要なアクセス許可は管理者のみに付与されていることが望ましい状態です。したがって、管理者が最初のセットアップ時にサービスロールを作成することをお勧めします。サービスロールの作成に必要なアクセス許可の詳細については、「[AWS サービスにアクセス許可を委任するためのロールの作成](#)」を参照してください。

### サービスロールの作成に必要なアクセス許可

サービスロールを作成するときは、ロールに対して 2 つのポリシーを設定します。ロールが実行できる操作を指定する IAM アクセス許可ポリシーをロールにアタッチします。また、ロールを使用できるプリンシパルを指定するロール信頼ポリシーをロールにアタッチします。

最初のポリシーの例は、ポリシーを生成するために必要なサービスロールのアクセス許可ポリシーを示しています。2 番目のポリシーの例は、サービスロールに必要なロール信頼ポリシーを示しています。これらのポリシーを使用すると、ポリシーを生成するために AWS API や AWS CLI を使用してサービスロールを作成できます。IAM コンソールを使用してポリシー生成プロセスの一部としてサービスロールを作成すると、これらのポリシーが生成されます。

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "cloudtrail:GetTrail",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:GetServiceLastAccessedDetails",
      "iam:GenerateServiceLastAccessedDetails"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    ]
  }
]
```

次のポリシーの例は、IAM Access Analyzer がロールを担うことができるアクセス許可を与えるロール信頼ポリシーを示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "access-analyzer.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

## その後の使用について

AWS Management Console でポリシーを生成するには、ポリシー生成に使用されるサービスロールを IAM Access Analyzer に渡すことができるアクセス許可ポリシーが IAM ユーザーに必要です。iam:PassRole には通常 iam:GetRole が付随します。これにより、ユーザーは渡されるロールの詳細を取得できます。この例でユーザーが渡すことができるのは、指定されたアカウントに存在し、AccessAnalyzerMonitorServiceRole\* で始まる名前を持つロールに限ります。IAM ロールの AWS サービスへの受け渡しの詳細については、「[AWS サービスにロールを渡すアクセス許可をユーザーに許可する](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-role/
AccessAnalyzerMonitorServiceRole*"
    }
  ]
}
```

AWS、AWS Management Console API、または AWS CLI でポリシーを生成するには、次のポリシーステートメントに示すように、次の IAM Access Analyzer のアクセス許可も必要です。

```
{
  "Sid": "AllowUserToGeneratePolicy",
  "Effect": "Allow",
  "Action": [
    "access-analyzer:CancelPolicyGeneration",
    "access-analyzer:GetGeneratedPolicy",
    "access-analyzer:ListPolicyGenerations",
    "access-analyzer:StartPolicyGeneration"
  ],
  "Resource": "*"
}
```



## 初回およびその後の使用の両方について

AWS Management Console を使用してポリシーを生成する場合、次のポリシーステートメントに示すように、アカウント内の CloudTrail 証跡を一覧表示する `cloudtrail:ListTrails` アクセス許可が必要です。

```
{
  "Sid": "AllowUserToListTrails",
  "Effect": "Allow",
  "Action": [
    "CloudTrail:ListTrails"
  ],
  "Resource": "*"
}
```

## CloudTrail アクティビティに基づくポリシーの生成 (コンソール)

IAM ユーザーまたはロールのポリシーを生成できます。

### ステップ 1: CloudTrail アクティビティに基づいてポリシーを生成する

次の手順で、AWS Management Console を使用してロールのポリシーを生成します。

#### IAM ロールのポリシーを生成する

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. 左側のナビゲーションペインで、[Roles] を選択します。

#### Note

IAM ユーザーのアクティビティに基づいてポリシーを生成する手順は、ほぼ同じです。そのためには、[Roles] (ロール) ではなく [Users] (ユーザー) を選択します。

3. アカウントのロールのリストで、ポリシーの生成に使用するアクティビティを行なったロールの名前を選択します。
4. [アクセス許可] タブの [CloudTrail イベントに基づいてポリシーを生成] セクションで、[ポリシーの生成] を選択します。



5. [ポリシーの生成] ページで、ロールで実行されたアクションについて IAM Access Analyzer が CloudTrail イベントを分析する期間を指定します。最大 90 日の範囲を選択できます。ポリシー生成時間を短縮するために、可能な限り最短の期間を選択することをお勧めします。
6. CloudTrail アクセスセクションで、適切な既存のロールを選択するか、適切なロールが存在しない場合は新しいロールを作成します。このロールは、ユーザーの代わりに CloudTrail データにアクセスするためのアクセス許可を IAM Access Analyzer に与え、アクセスを確認し、使用されているサービスとアクションを特定します。このロールに必要なアクセス許可の詳細については、「[ポリシーの生成に必要なアクセス許可](#)」を参照してください。
7. [分析対象の CloudTrail 証跡] セクションで、アカウントのイベントをログに記録する CloudTrail 証跡を指定します。

別のアカウントにログを保存する CloudTrail 証跡を選択すると、クロスアカウントアクセスに関する情報ボックスが表示されます。クロスアカウントアクセスでは、追加の設定が必要です。詳細については、このトピックの後半の「[Choose a role for cross-account access](#)」を参照してください。

8. [ポリシーの生成] を選択します。
9. ポリシー生成中は、[Permissions] (アクセス許可) タブの [Roles] (ロール) [Summary] (概要) ページに戻ります。[ポリシー要求の詳細] セクションの状態に [成功] が表示されるまで待ち、[生成されたポリシーを表示] を選択します。生成されたポリシーは、最大 7 日間表示できます。別のポリシーを生成すると、既存のポリシーは生成した新しいポリシーに置き換えられます。

## ステップ 2: アクセス許可を確認し、使用するサービスのアクションを追加する

ロールによって使用されたことを IAM Access Analyzer が特定したサービスとアクションを確認します。生成されたポリシーテンプレートに使用されたすべてのサービスに対するアクションを追加できます。

1. 以下のセクションを確認します。
  - [権限の確認] ページで、[生成されたポリシーに含まれるアクション] のリストを確認します。リストには、指定した日付範囲内でロールによって使用されたと IAM Access Analyzer が特定したサービスおよびアクションが表示されます。
  - [使用されたサービス] セクションには、指定された日付範囲内のロールによって使用されたと IAM Access Analyzer が特定した追加のサービスが表示されます。このセクションのサービスのリストからは、どのアクションが使用されたかの情報が得られない場合があります。リストされた各サービスのメニューを使用して、ポリシーに含めるアクションを手動で選択します。

## 2. アクションの追加が完了したら、[次へ] をクリックします。

### ステップ 3: 生成されたポリシーをさらにカスタマイズする

アクセス許可を追加または削除したり、リソースを指定したりして、ポリシーをさらにカスタマイズできます。

生成されたポリシーをカスタマイズするには

1. ポリシーテンプレートを更新します。次の図に示すように、ポリシーテンプレートには、リソースレベルのアクセス許可をサポートするアクション用のリソース ARN プレースホルダーが含まれています。リソースレベルのアクセス許可とは、ユーザーがアクションを実行できるリソースを指定できる機能を意味します。リソースレベルのアクセス許可をサポートするアクションについては、[ARN](#) を使用して、ポリシーで個々のリソースを指定することをお勧めします。プレースホルダーリソース ARN は、ユースケース用の有効なリソース ARN に置き換えることができます。

アクションがリソースレベルのアクセス許可をサポートしていない場合は、ワイルドカード (\*) を使用して、すべてのリソースがアクションの影響を受ける可能性があることを指定する必要があります。どの AWS サービスがリソースレベルのアクセス許可をサポートしているかを確認するには、「[IAM と連携する AWS サービス](#)」を参照してください。各サービスのアクションの一覧と、どのアクションがリソースレベルのアクセス許可をサポートしているかについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

Generated policy

1 2 3

Customize permissions

Review the following policy template. You must specify resources for actions that support resource-level permissions to continue creating the policy.

The screenshot displays the AWS IAM console interface for editing a policy. On the left, a code editor shows a JSON policy template. The following code is visible:

```

1 - {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "access-analyzer:ValidatePolicy",
8         "iam:GetAccountPasswordPolicy",
9         "iam:GetAccountSummary",
10        "iam:ListAccountAliases",
11        "iam:ListGroups",
12        "iam:ListPolicies",
13        "iam:ListRoles",
14        "iam:ListUsers"
15      ],
16      "Resource": "*"
17    },
18  ],
19  {
20    "Effect": "Allow",
21    "Action": [
22      "iam:GetRole",
23      "iam:ListAttachedRolePolicies",
24      "iam:ListInstanceProfilesForRole",
25      "iam:ListRolePolicies",
26      "iam:ListRoleTags"
27    ],
28    "Resource": "arn:aws:iam:${Account}:role/${RoleNameWithPath}"
29  },
30  {
31    "Effect": "Allow",
32    "Action": [
33      "iam:GetUser",
34      "iam:ListAccessKeys",
35      "iam:ListAttachedUserPolicies",
36      "iam:ListGroupsWithUser",
37      "iam:ListUserTags"
38    ],
39    "Resource": "arn:aws:iam:${Account}:user/${UserNameWithPath}"
40  }
41 ]

```

On the right, the 'Edit statement' panel is visible. It contains the text 'Select a statement' and 'Select an existing statement in the policy or add a new statement.' Below this text is a button labeled '+ Add new statement'.

2. (オプション) テンプレートの JSON ポリシーステートメントを追加、変更、または削除します。JSON ポリシーの記述の詳細については、「[IAM ポリシーの作成 \(コンソール\)](#)」を参照してください。
3. ポリシーテンプレートのカスタマイズ完了後、以下のオプションがあります。
  - (オプション) テンプレート内の JSON をコピーして、[生成されたポリシー] ページの外部で別途使用できます。たとえば、JSON を使用して別のアカウントでポリシーを作成する場合などです。テンプレート内のポリシーが JSON ポリシーの 6,144 文字の制限を超える場合、ポリシーは複数のポリシーに分割されます。
  - [次へ] をクリックして、同じアカウントでマネージドポリシーを確認して作成します。

## ステップ 4: マネージドポリシーを確認して作成する

IAM ポリシーを作成してアタッチするアクセス許可がある場合は、生成されたポリシーからマネージドポリシーを作成できます。その後、アカウントのユーザーまたはロールにポリシーをアタッチできます。

ポリシーを確認して作成するには

1. [マネージドポリシーの確認と作成] ページに作成するポリシーの [名前] と [説明] (オプション) を入力します。
2. (オプション) [概要] セクションでは、ポリシーに含まれるアクセス許可を確認できます。
3. (オプション) タグをキー - 値のペアとしてアタッチして、メタデータをポリシーに追加します。IAM でのタグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。
4. 完了したら、次のいずれかの操作を行います。
  - 新しいポリシーは、ポリシーの生成に使用されたロールに直接アタッチできます。これを行うには、ページの下部の [ポリシーを **YourRoleName** にアタッチする] の横にあるチェックボックスをオンにします。次に、[ポリシーを作成してアタッチ] を選択します。
  - あるいは、[ポリシーの作成] を選択します。作成したポリシーは、IAM コンソールの [ポリシー] ナビゲーションペインのポリシーの一覧に表示されます。
5. 作成したポリシーは、アカウントのエンティティにアタッチできます。ポリシーをアタッチした後、エンティティにアタッチされている可能性がある他の広範囲過ぎるポリシーを削除できます。マネージドポリシーをユーザーにアタッチする方法については、「[IAM ID アクセス許可の追加 \(コンソール\)](#)」を参照してください。

## 別のアカウントの AWS CloudTrail データを使用してポリシーを生成する

セントラルアカウントにデータを保存する CloudTrail トレイルを作成して、管理活動を合理化できます。たとえば、AWS Organizations を使用して、その組織内のすべての AWS アカウントの全イベントをログに記録する証跡を作成できます。証跡はセントラルアカウントに属します。CloudTrail ログデータが格納されているアカウントとは異なるアカウントでユーザーまたはロールのポリシーを生成する場合は、クロスアカウントアクセスを許可する必要があります。これを行うには、CloudTrail ログに対する IAM Access Analyzer のアクセス許可を付与するロールとバケットポリシーの両方が必要です。組織の証跡の作成についての詳細は、ユーザーガイドの「[組織の証跡の作成](#)」を参照してください。

この例では、アカウント A のユーザーまたはロールのポリシーを生成するとします。アカウント A の CloudTrail トレイルは、アカウント B のバケットに CloudTrail ログを格納します。ポリシーを生成する前に、次の更新を行う必要があります。

1. 既存のロールを選択するか、アカウント B ( CloudTrail ログが格納されている ) のバケットへの IAM Access Analyzer アクセスを許可する新しいサービスロールを作成します。
2. アカウント B で Amazon S3 バケットオブジェクトの所有権およびバケットのアクセス許可ポリシーを確認して、IAM Access Analyzer がバケット内のオブジェクトにアクセスできるようにします。

### ステップ 1: クロスアカウントアクセス用のロールを選択または作成する

- リポジトリの [ポリシーを生成する]画面では、既存のロールを使用するは、必要な権限を持つロールがアカウントに存在する場合、事前を選択されています。それ以外の場合は [新しいサービスロールの作成と使用] を選択します。新しいロールは、アカウント B の CloudTrail ログへの IAM アクセスアナライザのアクセス権限を付与するために使用されます。

### ステップ 2: アカウント B で Amazon S3 バケットの設定を確認または更新します

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. バケットリストで、CloudTrail トレイルログが保存されるバケットの名前を選択します。
3. [Permissions] (アクセス許可) タブを選択し、[Object Ownership] (オブジェクトの所有権) セクションに移動します。

Amazon S3 オブジェクトの所有権のバケット設定を使用して、バケットにアップロードするオブジェクトの所有権を制御します。デフォルトでは、他の AWS アカウント がバケットにオブジェクトをアップロードした場合、アップロードしたアカウントがそのオブジェクトを所有します。ポリシーを生成するには、バケットの所有者がバケット内のすべてのオブジェクトを所有している必要があります。ご利用の ACL ユースケースによっては、バケットの [Object Ownership] (オブジェクトの所有権) の設定を変更する必要があるかもしれません。[Object Ownership] (オブジェクトの所有権) を次のオプションのいずれかに設定します。

- [Bucket owner enforced] (バケット所有者の強制) (推奨)
- [Bucket owner preferred] (バケット所有者優先)

**⚠ Important**

ポリシーを正常に生成するには、バケット内のオブジェクトをバケット所有者が所有している必要があります。[Bucket owner preferred] (バケット所有者優先) を選択した場合、オブジェクトの所有権が変更された後の期間のみポリシーを生成することができます。

Amazon S3 におけるオブジェクトの所有権の詳細については、「Amazon S3 ユーザーガイド」の「[オブジェクトの所有権の制御およびバケットに対する ACL の無効化](#)」を参照してください。

4. アカウント B の Amazon S3 バケットポリシーにアクセス許可を追加して、アカウント A のロールへのアクセスを許可します。

次のポリシー例では、ListBucket という名前のバケットに対して GetObject と DOC-EXAMPLE-BUCKET を許可しています。バケットにアクセスするロールが組織内のアカウントに属し、名前が AccessAnalyzerMonitorServiceRole で始まる場合にアクセスを許可します。Resource 要素で [aws:PrincipalArn](#) を Condition として使用すると、ロールがアカウント A に属している場合にのみ、アカウントのアクティビティにアクセスできるようになります。DOC-EXAMPLE-BUCKET をバケット名に、optional-prefix をバケットのオプションプレフィックスに、organization-id を組織 ID に置き換えることができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

{
  "Sid": "PolicyGenerationBucketPolicy",
  "Effect": "Allow",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "s3:GetObject",
    "s3:ListBucket"
  ],
  "Resource": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET/optional-prefix/AWSLogs/organization-id/
    ${aws:PrincipalAccount}/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:PrincipalOrgID": "organization-id"
    },
    "StringLike": {
      "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
      role/AccessAnalyzerMonitorServiceRole*"
    }
  }
}

```

5. AWS KMS を使用してログを暗号化する場合は、次のポリシーの例に示すように、CloudTrail ログを保存するアカウントで AWS KMS キーポリシーを更新して、IAM AccessAnalyzer にキーを使用するためのアクセスを許可します。CROSS\_ACCOUNT\_ORG\_TRAIL\_FULL\_ARN を証跡の ARN に置き換え、organization-id を組織 ID に置き換えます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
    }
  ]
}

```

```
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:cloudtrail:arn":
"CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN",
        "aws:PrincipalOrgID": "organization-id"
      },
      "StringLike": {
        "kms:ViaService": [
          "access-analyzer.*.amazonaws.com",
          "s3.*.amazonaws.com"
        ]
        "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
role/AccessAnalyzerMonitorServiceRole*"
      }
    }
  }
}
```

## CloudTrail アクティビティに基づくポリシーの生成 (AWS CLI)

次のコマンドを使用して、AWS CLI を使用したポリシーを生成できます。

ポリシーを生成するには

- [aws accessanalyzer start-policy-generation](#)

生成されたポリシーを表示するには

- [aws accessanalyzer get-generated-policy](#)

ポリシー生成要求を取り消すには

- [aws accessanalyzer cancel-policy-generation](#)

ポリシー生成要求のリストを表示するには

- [aws accessanalyzer list-policy-generations](#)



## CloudTrail アクティビティに基づいたポリシーの生成 (AWS API)

以下のオペレーションを使用して、AWS API を使用したポリシーの生成ができます。

ポリシーを生成するには

- [StartPolicyGeneration](#)

生成されたポリシーを表示するには

- [GetGeneratedPolicy](#)

ポリシー生成要求を取り消すには

- [CancelPolicyGeneration](#)

ポリシー生成要求のリストを表示するには

- [ListPolicyGenerations](#)

## IAM Access Analyzer のポリシー生成サービス

次の表は、[IAM Access Analyzer](#) がアクションレベル情報を含むポリシーを生成する AWS サービスを一覧で示しています。各サービスのアクションのリストについては、「サービス認証リファレンス」の「[AWS サービスのアクション、リソース、条件キー](#)」を参照してください。

サービス	サービスプレフィックス
<a href="#">AWS Identity and Access Management Access Analyzer</a>	access-analyzer
<a href="#">AWS Account Management</a>	アカウント
<a href="#">AWS Certificate Manager</a>	acm
<a href="#">Amazon Managed Workflows for Apache Airflow</a>	airflow
<a href="#">AWS Amplify</a>	amplify



サービス	サービスプレフィックス
<a href="#">AWS Amplify UI Builder</a>	amplifyuibuilder
<a href="#">Amazon アプリインテグレーション</a>	app-integrations
<a href="#">AWS AppConfig</a>	appconfig
<a href="#">Amazon AppFlow</a>	appflow
<a href="#">AWS Application Cost Profiler</a>	application-cost-profiler
<a href="#">Amazon CloudWatch Application Insights</a>	applicationinsights
<a href="#">AWS App Mesh</a>	appmesh
<a href="#">Amazon AppStream 2.0</a>	appstream
<a href="#">AWS AppSync</a>	appsync
<a href="#">Amazon Managed Service for Prometheus</a>	aps
<a href="#">Amazon Athena</a>	athena
<a href="#">AWS Audit Manager</a>	auditmanager
<a href="#">AWS Auto Scaling</a>	オートスケーリング
<a href="#">AWS Marketplace</a>	aws-marketplace
<a href="#">AWS Backup</a>	バックアップ
<a href="#">AWS Batch</a>	バッチ
<a href="#">Amazon Braket</a>	braket
<a href="#">AWS Budgets</a>	予算
<a href="#">AWS Cloud9</a>	cloud9

サービス	サービスプレフィックス
<a href="#">AWS CloudFormation</a>	cloudformation
<a href="#">Amazon CloudFront</a>	cloudfront
<a href="#">AWS CloudHSM</a>	cloudhsm
<a href="#">Amazon CloudSearch</a>	cloudsearch
<a href="#">AWS CloudTrail</a>	CloudTrail
<a href="#">Amazon CloudWatch</a>	cloudwatch
<a href="#">AWS CodeArtifact</a>	codeartifact
<a href="#">AWS CodeDeploy</a>	codedeploy
<a href="#">Amazon CodeGuru Profiler</a>	codeguru-profiler
<a href="#">Amazon CodeGuru Reviewer</a>	codeguru-reviewer
<a href="#">AWS CodePipeline</a>	codepipeline
<a href="#">AWS CodeStar</a>	codestar
<a href="#">AWS CodeStar 通知</a>	codestar-notifications
<a href="#">Amazon Cognito ID</a>	cognito-identity
<a href="#">Amazon Cognito ユーザープール</a>	cognito-idp
<a href="#">Amazon Cognito Sync</a>	cognito-sync
<a href="#">Amazon Comprehend Medical</a>	comprehen dmedical
<a href="#">AWS Compute Optimizer</a>	compute-optimizer
<a href="#">AWS Config</a>	config

サービス	サービスプレフィックス
<a href="#">Amazon Connect</a>	接続
<a href="#">AWS Cost and Usage Report</a>	cur
<a href="#">AWS Glue DataBrew</a>	databrew
<a href="#">AWS Data Exchange</a>	dataexchange
<a href="#">AWS Data Pipeline</a>	datapipeline
<a href="#">DynamoDB Accelerator</a>	dax
<a href="#">AWS Device Farm</a>	devicefarm
<a href="#">Amazon DevOps Guru</a>	devops-guru
<a href="#">AWS Direct Connect</a>	directconnect
<a href="#">Amazon Data Lifecycle Manager</a>	dlm
<a href="#">AWS Database Migration Service</a>	dms
<a href="#">Amazon DocumentDB Elastic Clusters</a>	docdb-elastic
<a href="#">AWS Directory Service</a>	ds
<a href="#">Amazon DynamoDB</a>	dynamodb
<a href="#">Amazon Elastic Block Store</a>	ebs
<a href="#">Amazon Elastic Compute Cloud</a>	Ec2
<a href="#">Amazon Elastic Container Registry</a>	ecr
<a href="#">Amazon Elastic Container Registry Public</a>	ecr-public
<a href="#">Amazon Elastic Container Service</a>	Ecs
<a href="#">Amazon Elastic Kubernetes Service</a>	eks

サービス	サービスプレフィックス
<a href="#">Amazon Elastic Inference</a>	elastic-inference
<a href="#">Amazon ElastiCache</a>	elasticache
<a href="#">AWS Elastic Beanstalk</a>	elasticbeanstalk
<a href="#">Amazon Elastic File System</a>	elasticfilesystem
<a href="#">Elastic Load Balancing</a>	elasticloadbalancing
<a href="#">Amazon Elastic Transcoder</a>	elastictranscoder
<a href="#">Amazon EMR on EKS (EMR コンテナ)</a>	emr-containers
<a href="#">Amazon EMR Serverless</a>	emr-serverless
<a href="#">Amazon OpenSearch Service</a>	es
<a href="#">Amazon EventBridge</a>	events
<a href="#">Amazon CloudWatch Evidently</a>	evidently
<a href="#">Amazon FinSpace</a>	finspace
<a href="#">Amazon Data Firehose</a>	firehose
<a href="#">AWS Fault Injection Service</a>	fis
<a href="#">AWS Firewall Manager</a>	fms
<a href="#">Amazon Fraud Detector</a>	frauddetector
<a href="#">Amazon FSx</a>	fsx
<a href="#">Amazon GameLift</a>	gamelift
<a href="#">Amazon Location Service</a>	geo

サービス	サービスプレフィックス
<a href="#">Amazon S3 Glacier</a>	glacier
<a href="#">Amazon Managed Grafana</a>	grafana
<a href="#">AWS IoT Greengrass</a>	greengrass
<a href="#">AWS Ground Station</a>	groundstation
<a href="#">Amazon GuardDuty</a>	guardduty
<a href="#">AWS HealthLake</a>	healthlake
<a href="#">Amazon Honeycode</a>	honeycode
<a href="#">AWS Identity and Access Management</a>	iam
<a href="#">AWS Identity Store</a>	identitystore
<a href="#">EC2 Image Builder</a>	imagebuilder
<a href="#">Amazon Inspector Classic</a>	inspector
<a href="#">Amazon Inspector</a>	inspector2
<a href="#">AWS IoT</a>	iot
<a href="#">AWS IoT Analytics</a>	iotanalytics
<a href="#">AWS IoT Core Device Advisor</a>	iotdeviceadvisor
<a href="#">AWS IoT Events</a>	iotevents
<a href="#">AWS IoT Fleet Hub</a>	iotfleethub
<a href="#">AWS IoT SiteWise</a>	ioticsitewise
<a href="#">AWS IoT TwinMaker</a>	iotwinmaker
<a href="#">AWS IoT Wireless</a>	iotwireless

サービス	サービスプレフィックス
<a href="#">Amazon Interactive Video Service</a>	ivs
<a href="#">Amazon Interactive Video Service Chat</a>	ivschat
<a href="#">Amazon Managed Streaming for Apache Kafka</a>	kafka
<a href="#">Amazon Managed Streaming for Kafka Connect</a>	kafkaconnect
<a href="#">Amazon Kendra</a>	kendra
<a href="#">Amazon Kinesis</a>	kinesis
<a href="#">Amazon Kinesis Analytics V2</a>	kinesisanalytics
<a href="#">AWS Key Management Service</a>	kms
<a href="#">AWS Lambda</a>	lambda
<a href="#">Amazon Lex</a>	lex
<a href="#">AWS License Manager Linux サブスクリプションマネージャー</a>	license-manager-linux-subscriptions
<a href="#">Amazon Lightsail</a>	lightsail
<a href="#">Amazon CloudWatch Logs</a>	ログ
<a href="#">Amazon Lookout for Equipment</a>	lookoutequipment
<a href="#">Amazon Lookout for Metrics</a>	lookoutmetrics
<a href="#">Amazon Lookout for Vision</a>	lookoutvision
<a href="#">AWS Mainframe Modernization</a>	m2
<a href="#">Amazon Managed Blockchain</a>	managedblockchain
<a href="#">AWS Elemental MediaConnect</a>	mediaconnect

サービス	サービスプレフィックス
<a href="#">AWS Elemental MediaConvert</a>	mediaconvert
<a href="#">AWS Elemental MediaLive</a>	medialive
<a href="#">AWS Elemental MediaStore</a>	mediastore
<a href="#">AWS Elemental MediaTailor</a>	mediatailor
<a href="#">Amazon MemoryDB for Redis</a>	memorydb
<a href="#">AWS Application Migration Service</a>	mgn
<a href="#">AWS Migration Hub</a>	mgh
<a href="#">AWS Migration Hub 戦略レコメンデーション</a>	migration hub-strategy
<a href="#">Amazon Pinpoint</a>	mobiletargeting
<a href="#">Amazon MQ</a>	mq
<a href="#">AWS Network Manager</a>	networkmanager
<a href="#">Amazon Nimble Studio</a>	nimble
<a href="#">AWS HealthOmics</a>	omics
<a href="#">AWS OpsWorks</a>	opsworks
<a href="#">AWS OpsWorks CM</a>	opsworks-cm
<a href="#">AWS Outposts</a>	outposts
<a href="#">AWS Organizations</a>	組織
<a href="#">AWS Panorama</a>	panorama
<a href="#">AWS Performance Insights</a>	pi

サービス	サービスプレフィックス
<a href="#">Amazon EventBridge パイプ</a>	pipes
<a href="#">Amazon Polly</a>	polly
<a href="#">Amazon Connect Customer Profiles</a>	profile
<a href="#">Amazon QLDB</a>	qldb
<a href="#">AWS Resource Access Manager</a>	ram
<a href="#">AWS ごみ箱</a>	rbin
<a href="#">Amazon Relational Database Service</a>	rds
<a href="#">Amazon Redshift</a>	redshift
<a href="#">Amazon Redshift Data API</a>	redshift-data
<a href="#">AWS Migration Hub Refactor Spaces</a>	refactor-spaces
<a href="#">Amazon Rekognition</a>	rekognition
<a href="#">AWS Resilience Hub</a>	resiliencehub
<a href="#">AWS Resource Explorer</a>	resource-explorer-2
<a href="#">AWS Resource Groups</a>	resource-groups
<a href="#">AWS RoboMaker</a>	robomaker
<a href="#">AWS Identity and Access Management Roles Anywhere</a>	rolesanywhere
<a href="#">Amazon Route 53</a>	route53
<a href="#">Amazon Route 53 Recovery コントロール</a>	route53-recovery-control-config



サービス	サービスプレフィックス
<a href="#">Amazon Route 53 Recovery 準備状況</a>	route53-recovery-readiness
<a href="#">Amazon Route 53 Resolver</a>	route53resolver
<a href="#">AWS CloudWatch RUM</a>	rum
<a href="#">Amazon Simple Storage Service</a>	s3
<a href="#">Amazon S3 on Outposts</a>	s3-outposts
<a href="#">Amazon SageMaker 地理空間機能</a>	sagemaker-geospatial
<a href="#">Savings Plans</a>	savingsplans
<a href="#">Amazon EventBridge スキーマ</a>	schemas
<a href="#">Amazon SimpleDB</a>	sdb
<a href="#">AWS Secrets Manager</a>	secretsmanager
<a href="#">AWS Security Hub</a>	securityhub
<a href="#">Amazon Security Lake</a>	securitylake
<a href="#">AWS Serverless Application Repository</a>	serverlessrepo
<a href="#">AWS Service Catalog</a>	servicecatalog
<a href="#">AWS Cloud Map</a>	servicediscovery
<a href="#">Service Quotas</a>	servicequotas
<a href="#">Amazon Simple Email Service</a>	ses
<a href="#">AWS Shield</a>	shield
<a href="#">AWS Signer</a>	signer

サービス	サービスプレフィックス
<a href="#">AWS SimSpace Weaver</a>	simspaceweaver
<a href="#">AWS Server Migration Service</a>	sms
<a href="#">Amazon Pinpoint SMS および音声サービス</a>	sms-voice
<a href="#">AWS Snowball</a>	snowball
<a href="#">Amazon Simple Queue Service</a>	sqs
<a href="#">AWS Systems Manager</a>	ssm
<a href="#">AWS Systems Manager Incident Manager</a>	ssm-incidents
<a href="#">AWS Systems Manager for SAP</a>	ssm-sap
<a href="#">AWS Step Functions</a>	states
<a href="#">AWS Security Token Service</a>	sts
<a href="#">Amazon Simple Workflow Service</a>	swf
<a href="#">Amazon CloudWatch Synthetics</a>	synthetics
<a href="#">AWS Resource Groups Tagging API</a>	タグ
<a href="#">Amazon Textract</a>	textract
<a href="#">Amazon Timestream</a>	timestream
<a href="#">AWS 通信ネットワークビルダー</a>	tnb
<a href="#">Amazon Transcribe</a>	transcribe
<a href="#">AWS Transfer Family</a>	移管
<a href="#">Amazon Translate</a>	translate
<a href="#">Amazon Connect Voice ID</a>	voiceid

サービス	サービスプレフィックス
<a href="#">Amazon VPC Lattice</a>	vpc-lattice
<a href="#">AWS WAFV2</a>	wafv2
<a href="#">AWS Well-Architected Tool</a>	wellarchitected
<a href="#">Amazon Connect Wisdom</a>	wisdom
<a href="#">Amazon WorkLink</a>	worklink
<a href="#">Amazon WorkSpaces</a>	ワークスペース
<a href="#">AWS X-Ray</a>	xray

## IAM Access Analyzer のクォータ

IAM Access Analyzer には、以下のクォータがあります。

リソース	デフォルトのクォータ	最大クォータ
各リージョンの AWS アカウントのアナライザータイプごとのアカウントレベルのアナライザーの最大数	1	1
各リージョンの AWS アカウントのアナライザータイプごとの組織レベルのアナライザーの最大数	5	20 <sup>1</sup>
アナライザーあたりの最大アーカイブルール数	100	1,000 <sup>1</sup>
	各アーカイブルールには、基準ごとに最大 20 個の値を指定できます。	

リソース	デフォルトのクォータ	最大クォータ
1 時間あたりのアナライザ ごとのアクセスプレビューの 最高数	1,000	1,000
ポリシー世代ごとに処理され る AWS CloudTrail ログファイ ル	100,000	100,000
ポリシー生成の同時実行	1	1
ポリシーの生成 AWS CloudTrail データサイズ	25 GB	25 GB
ポリシーの生成 AWS CloudTrail 時間範囲	90 日間	90 日間
1 日あたりのポリシー生成数	アフリカ (ケープタウン): 5 アジアパシフィック (香港): 5 ヨーロッパ (ミラノ): 5 中東 (バーレーン): 5 その他のサポートされている すべてのリージョン: 50	アフリカ (ケープタウン): 5 アジアパシフィック (香港): 5 ヨーロッパ (ミラノ): 5 中東 (バーレーン): 5 その他のサポートされている すべてのリージョン: 50

 **Note**

キャンセルされたポリ  
シー生成要求は、毎日  
のクォータに適用され  
ます。

<sup>1</sup> 一部のクォータは、[Service Quotas](#)を使用してお客様が設定できます。

# IAM のトラブルシューティング

アクセスが拒否されたり、AWS Identity and Access Management (IAM) を操作中にエラーが発生したりする場合は、このセクションのトピックを参照してください。

## トピック

- [一般的な IAM の問題のトラブルシューティング](#)
- [アクセス拒否エラーメッセージのトラブルシューティング](#)
- [IAM ポリシーのトラブルシューティング](#)
- [FIDO セキュリティキーのトラブルシューティング](#)
- [IAM ロールのトラブルシューティング](#)
- [IAM および Amazon EC2 のトラブルシューティング](#)
- [Amazon S3 および IAM のトラブルシューティング](#)
- [AWS での SAML 2.0 フェデレーションのトラブルシューティング](#)

## 一般的な IAM の問題のトラブルシューティング

この情報を使用して、AWS Identity and Access Management (IAM) を使用する際に一般的な問題の診断と修正を行います。

### 問題点

- [自分の AWS アカウントにサインインできません](#)
- [アクセスキーを紛失した](#)
- [ポリシーの変数が機能していない](#)
- [行った変更がすぐに表示されないことがある](#)
- [iam:DeleteVirtualMFADevice を実行する権限がありません](#)
- [IAM ユーザーを安全に作成するにはどうすればよいですか?](#)
- [その他のリソース](#)

## 自分の AWS アカウントにサインインできません

正しい認証情報を持っていること、およびサインインに正しい方法を使用していることを確認します。詳しい情報については、「AWS サインイン ユーザーガイド」の「[サインインに関する問題のトラブルシューティング](#)」を参照してください。

## アクセスキーを紛失した

アクセスキーは 2 つのパートで構成されます。

- アクセスキー識別子。これはシークレットではなく、ユーザー概要ページなど、IAM コンソールでアクセスキーがリストされるすべての場所に表示されます。
- シークレットアクセスキー。これは最初にアクセスキーペアを作成するときに提供されます。パスワードと同じように、後で取得することはできません。シークレットアクセスキーを紛失した場合は、新しいアクセスキーペアを作成する必要があります。[アクセスキーの最大数](#)に達している場合は、既存のペアを削除してから新しいものを作成する必要があります。

詳細については、「[紛失したり忘れたりしたパスワードまたは AWS のアクセスキーのリセット](#)」を参照してください。

## ポリシーの変数が機能していない

- 変数のあるすべてのポリシーで、バージョン番号: "Version": "2012-10-17" がポリシーに含まれていることを確認します。正しいバージョン番号がないと、評価時に変数は置き換えられません。代わりに、変数は逐語的に評価されます。変数の含まれていないポリシーは、最新のバージョン番号が含まれていれば正常に機能します。

Version ポリシー要素は、ポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。一方で、ポリシーバージョンは、IAM でカスタマー管理ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。Version ポリシー要素の詳細については、「[IAM JSON ポリシー要素 Version](#)」を参照してください。ポリシーのバージョンの詳細については、「[the section called "IAM ポリシーのバージョンング"](#)」を参照してください。

- お客様のポリシーの変数が正しく設定されていることを確認してください。詳細については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## 行った変更がすぐに表示されないことがある

世界中のデータセンター内のコンピューターを介してアクセスされるサービスとして、IAM は、[結果整合性](#)と呼ばれる分散コンピューティングモデルを採用しています。IAM (または他の AWS サービス) で行った変更は、[属性ベースのアクセスコントロール \(ABAC\)](#) で使用されているタグを含め、すべての可能なエンドポイントから認識されるまでに時間がかかります。この遅延は、サーバー間、レプリケーションゾーン間、世界中のリージョン間でのデータ送信にかかる時間から発生している場合もあります。IAM ではパフォーマンス向上のためにキャッシュも使用しているため、これが原因で遅延が発生することがあります。変更は、以前にキャッシュされたデータがタイムアウトになるまで反映されない場合があります。

発生する可能性のあるこれらの遅延を考慮して、グローバルなアプリケーションを設計する必要があります。ある場所で行われた変更が他の場所で直ちに表示されない場合でも、正常に動作することを確認します。このような変更には、ユーザー、グループ、ロール、またはポリシーの作成や更新が含まれます。アプリケーションの重要で高可用性のコードパスには、このような IAM の変更を含めないことをお勧めします。代わりに、実行頻度が低い別の初期化またはセットアップルーチンに IAM の変更を加えます。また、本番稼働ワークフローが依存する前に、変更が伝達済みであることを確認します。

AWS の他のいくつかのサービスがこの遅延からどのような影響を受けるかの詳細については、以下のリソースを参照してください。

- Amazon DynamoDB: DynamoDB のよくある質問の「[Amazon DynamoDB の整合性モデルとは何ですか?](#)」および Amazon DynamoDB デベロッパーガイドの「[読み込み整合性](#)」。
- Amazon EC2: [Amazon EC2 API リファレンス](#)の EC2 結果整合性。
- Amazon EMR: AWS ビッグデータブログの「[ETL ワークフローに Amazon S3 および Amazon Elastic MapReduce を使用する際の整合性の確保](#)」
- Amazon Redshift: [Amazon Redshift Database デベロッパーガイド](#)の データの一貫性の管理
- Amazon S3: [Amazon Simple Storage Service ユーザーガイド](#)の Amazon S3 データ整合性モデル

## iam:DeleteVirtualMFADevice を実行する権限がありません

自分または他のユーザーに仮想 MFA デバイスを割り当てるまたは削除しようとする、次のエラーが表示されることがあります。

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to perform:
iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego with an
explicit deny
```

これは、誰かが以前に IAM コンソールでユーザーに仮想 MFA デバイスの割り当てを開始し、プロセスをキャンセルした場合に発生する可能性があります。これにより、IAM でユーザー用の仮想化 MFA デバイスが作成されますが、ユーザーに決して割り当てられません。新しい仮想化 MFA デバイスを同じデバイス名で作成する前に、既存の仮想化 MFA デバイスを削除する必要があります。

この問題を修正するために、管理者はポリシーのアクセス許可を編集しないでください。代わりに、管理者は AWS CLI または AWS API を使用して、既存のしかし割り当てられていない仮想化 MFA デバイスを削除する必要があります。

既存の未割り当て仮想化 MFA デバイスを削除するには

1. アカウントの仮想 MFA デバイスを表示します。
  - AWS CLI: [aws iam list-virtual-mfa-devices](#)
  - AWS API: [ListVirtualMFADevices](#)
2. レスポンスで、修正しようとしているユーザーの仮想化 MFA デバイスの ARN を見つけます。
3. 仮想化 MFA デバイスを削除します。
  - AWS CLI: [aws iam delete-virtual-mfa-device](#)
  - AWS API: [DeleteVirtualMFADevice](#)

## IAM ユーザーを安全に作成するにはどうすればよいですか？

AWS へのアクセスを必要とする従業員がいる場合は、IAM ユーザーを作成するか、[認証に IAM Identity Center を使用する](#)かを選択できます。IAM を使用する場合、AWS では、IAM ユーザーを作成し、その認証情報を従業員に安全に伝達することを推奨しています。従業員の隣に物理的に配置されていない場合は、安全なワークフローを使用して、従業員に認証情報を伝達します。

IAM で新しいユーザーを安全に作成するには、次のワークフローを使用します。

1. AWS Management Console を使用して[新しいユーザーを作成](#)します。自動生成されたパスワードを使用して AWS Management Console アクセス権を付与することを選択します。必要に応じて、[Users must create a new password at next sign-in] (ユーザーは次回サインイン時に新し



- いパスワードを作成する必要があります) のチェックボックスをオンにします。ユーザーがパスワードを変更するまで、アクセス許可ポリシーをユーザーに追加しないでください。
2. ユーザーを追加したら、新しいユーザーのサインイン URL、ユーザー名、およびパスワードをコピーします。パスワードを表示するには、[Show] (表示) を選択します。
  3. E メール、チャット、チケットシステムなど、社内の安全な通信方法を使用して、従業員にパスワードを送信します。別途、IAM ユーザーコンソールのリンクおよびユーザー名をユーザーに提供します。アクセス許可を付与する前に、正常にサインインできることを確認するよう従業員に伝えます。
  4. 従業員が確認したら、必要なアクセス許可を追加します。セキュリティのベストプラクティスとして、認証情報を管理するために MFA を使用した認証をユーザーに要求するポリシーを追加します。ポリシーの例については、[AWS: MFA で認証された IAM ユーザーが \[セキュリティ認証情報\] ページで自分の認証情報を管理できるようにします](#)を参照してください。

## その他のリソース

AWS を使用する際のトラブルシューティングに役立つリソースを以下に示します。

- [AWS CloudTrail ユーザーガイド](#) – AWS CloudTrail を使用して、AWS への API コールの履歴を追跡し、その情報をログファイルに保存します。この情報は、どのユーザーおよびアカウントがお客様のアカウントのリソースにアクセスしたか、いつその呼び出しが行われたか、どのアクションがリクエストされたかなどを調べるために役立ちます。詳細については、「[AWS CloudTrail による IAM および AWS STS の API コールのログ記録](#)」を参照してください。
- [AWS ナレッジセンター](#) — 問題のトラブルシューティングに役立つ FAQ やその他のリソースへのリンクを検索できます。
- [AWS サポートセンター](#) — テクニカルサポートを利用できます。
- [AWS プレミアムサポートセンター](#) – プレミアムテクニカルサポートを利用できます。

## アクセス拒否エラーメッセージのトラブルシューティング

アクセス拒否エラーは、AWS が認証リクエストを明示的または暗黙的に拒否した場合に表示されます。明示的な拒否は、特定の AWS アクションに対する Deny ステートメントがポリシーに含まれている場合に発生します。該当する Deny ステートメントがなく、該当する Allow ステートメントもない場合に、暗黙的な拒否が発生します。IAM ポリシーはデフォルトで IAM プリンシパルを拒否するため、ポリシーではプリンシパルにアクションの実行を明示的に許可する必要があります。それ以

外の場合、ポリシーは暗黙的にアクセスを拒否します。詳細については、「[明示的な拒否と暗黙的な拒否の違い](#)」を参照してください。

ポリシータイプが同じである複数のポリシーが認証リクエストを拒否した場合、AWS はアクセス拒否エラーメッセージでポリシーの番号を特定しません。複数のポリシータイプが原因で認証リクエストが拒否された場合、AWS は、これらのポリシータイプのうち 1 つだけをエラーメッセージに含めます。

#### Important

AWS へのサインインに問題がある場合 ユーザーのタイプに応じて正しい [AWS サインインページ](#)が表示されていることを確認します。AWS アカウントのルートユーザー (アカウント所有者) の場合は、AWS アカウントの作成時に設定した認証情報を使用して AWS にサインインできます。IAM ユーザーの場合、アカウント管理者から AWS へのサインインに使用可能な認証情報をもらうことができます。サポートをリクエストする必要がある場合、このページのフィードバックリンクは使用しないでください。このフォームは AWS サポートではなく、AWS Support ドキュメントチームに送信されます。代わりに、「[お問い合わせ](#)」ページで [AWS アカウントにまだログインできない] を選択してから、表示されているサポートオプションのいずれかを選択します。

## AWS サービスに要求を送信すると "アクセス拒否" が発生する

- エラーメッセージに、アクセスを拒否するポリシーの種類が含まれているかどうかを確認します。たとえば、サービス制御ポリシー (SCP) によりアクセスが拒否されたというエラーが表示された場合は、SCP の問題のトラブルシューティングに集中できます。ポリシータイプがわかっている場合は、そのポリシータイプのポリシーの特定のアクションに対する拒否ステートメントまたは不足している許可をチェックすることもできます。エラーメッセージに、アクセスを拒否するポリシーの種類が記載されていない場合は、このセクションの残りのガイドラインを参考にして、さらにトラブルシューティングを行ってください。
- リクエストしたアクションとリソースを呼び出すアイデンティティベースのポリシーのアクセス許可を持っていることを確認します。条件が付けられている場合、要求を送信するにはそれらの条件も満たしている必要があります。IAM ユーザー、グループ、ロールのポリシーの表示や修正の詳細については、「[IAM ポリシーを管理する](#)」を参照してください。
- AWS Management Console から、アクションを実行する権限がないと通知された場合、管理者に問い合わせ、サポートを依頼する必要があります。管理者からサインイン資格情報またはサインインリンクが提供されました。

以下のエラー例は、mateojackson IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の `widgets:GetWidget` 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
widgets:GetWidget on resource: my-example-widget
```

この場合、Mateo は、*my-example-widget* アクションを使用して `widgets:GetWidget` リソースにアクセスできるように、ポリシーの更新を管理者に依頼する必要があります。

- Amazon S3、Amazon SNS、Amazon SQS などの [リソースベースのポリシー](#) をサポートするサービスにアクセスしようとしていますか。その場合は、ポリシーでお客様がプリンシパルに指定されていて、アクセス権限を付与されていることを確認します。アカウント内のサービスにリクエストする場合は、アイデンティティベースのポリシーまたはリソースベースのポリシーを使用して、アクセス許可を付与することができます。別のアカウントのサービスにリクエストする場合は、アイデンティティベースのポリシーとリソースベースのポリシーの両方でアクセス許可を付与する必要があります。リソースベースのポリシーをサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。
- キーバリューのペアを持つ条件がポリシーに含まれている場合は、そのポリシーを慎重に確認します。この例には、`aws:RequestTag/tag-key` グローバル条件キー、AWS KMS `kms:EncryptionContext:encryption_context_key`、および複数のサービスでサポートされている `ResourceTag/tag-key` 条件キーが含まれます。キー名が複数の結果に一致しないことを確認します。条件キー名では大文字と小文字が区別されないため、foo というキーを確認する条件は、foo、Foo、または F00 に一致します。リクエストに複数のキーバリューのペアが含まれていて、キー名の大文字と小文字のみが異なる場合、アクセスは予期せず拒否される可能性があります。詳細については、「[IAM JSON ポリシー要素Condition](#)」を参照してください。
- [アクセス許可の境界](#) が設定されている場合は、アクセス許可の境界として使用されているポリシーでリクエストが許可されるかどうかを確認します。リクエストがアイデンティティベースのポリシーでは許可されるが、アクセス許可の境界では許可されない場合、リクエストは拒否されます。アクセス許可の境界は、IAM プリンシパル (ユーザーまたはロール) に付与できるアクセス許可の上限を設定します。リソースベースのポリシーは、アクセス許可の境界では制限されません。アクセス許可の境界は一般向けの機能ではありません。AWS によるポリシーの評価方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。
- 手動でリクエストに署名する ([AWS SDK](#) を使用しない) 場合は、正確に [リクエストに署名](#) していることを確認します。

## 一時的な認証情報を使用して要求を送信すると "アクセス拒否" が発生する

- まず、一時的な認証情報とは別の理由でアクセスが拒否されていないことを確認してください。詳細については、「[AWS サービスに要求を送信すると "アクセス拒否" が発生する](#)」を参照してください。
- サービスが一時的セキュリティ認証情報を受け入れることを確認します。「[IAM と連携する AWS のサービス](#)」を参照してください。
- リクエストが正しく署名されており、そのリクエストの形式が整っていることを確認します。詳細については、「[ツールキット](#)」ドキュメントまたは「[AWS リソースを使用した一時的な認証情報の使用](#)」を参照してください。
- 一時的な認証情報が失効していないことを確認します。詳細については、「[IAM の一時的な認証情報](#)」を参照してください。
- IAM ユーザーまたはロールに正しいアクセス権限があるかどうかを確認します。一時的なセキュリティ認証情報のアクセス許可は IAM ユーザーまたはロールから取得されます。その結果、アクセス許可は、一時的な認証情報を引き受けたロールに付与されているものに制限されます。一時的なセキュリティ認証情報に対するアクセス許可の決定方法の詳細については、「[一時的なセキュリティ認証情報のアクセス権限を制御する](#)」を参照してください。
- ロールを引き受ける場合は、ロールセッションはセッションポリシーによって制限される場合があります。AWS STS を使用してプログラムで[一時的なセキュリティ認証情報をリクエストする](#)場合は、オプションでインラインまたは管理[セッションポリシー](#)を渡すことができます。セッションポリシーは、ロールの一時認証情報セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。Policy パラメータを使用して単一の JSON インラインセッションポリシードキュメントを渡すことができます。PolicyArns パラメータを使用して、最大 10 個の管理セッションポリシーを指定できます。結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。または、管理者またはカスタムプログラムから一時的な認証情報が提供されている場合は、アクセスを制限するためのセッションポリシーが含まれている可能性があります。
- フェデレーテッドユーザーの場合は、セッションはセッションポリシーによって制限される場合があります。AWS に IAM ユーザーとしてサインインしてフェデレーショントークンをリクエストすることでフェデレーテッドユーザーになります。フェデレーテッドユーザーの詳細については、「[GetFederationToken — カスタム ID ブローカーを介したフェデレーション](#)」を参照してください。フェデレーショントークンをリクエスト中に ID ブローカーがセッションポリシーに合格した場合、セッションはそれらのポリシーによって制限されます。結果として得られるセッションのアクセス許可は、IAM ユーザーの ID ベースのポリシーとセッションポリシーの共通部分です。セッションポリシーの詳細については、「[セッションポリシー](#)」を参照してください。

- ロールを使用してリソースベースのポリシーのあるリソースへアクセスする場合は、ポリシーからロールにアクセス権限が付与されていることを確認します。たとえば、次のポリシーではアカウント `MyRole` からの `111122223333` の `MyBucket` へのアクセスを許可しています。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::MyBucket/*"]
  }]
}
```

## アクセス拒否エラーメッセージの例

ほとんどのアクセス拒否のエラーメッセージは、User *user* is not authorized to perform *action* on *resource* because *context* の形式です。この例では、#### はアクセスを受信しない [Amazon リソースネーム \(ARN\)](#) であり、##### はポリシーが拒否するサービスアクションであり、#### は、ポリシーが作用するリソースの ARN です。[*context*] (コンテキスト) フィールドには、ポリシータイプについての追加のコンテキストが表示され、ポリシーがアクセスを拒否した理由が説明されています。

ポリシー内の Deny ステートメントによって明示的にポリシーが拒否された場合、AWS はアクセス拒否エラーメッセージに with an explicit deny in a *type* policy を含めます。ポリシーが暗黙的に拒否された場合は、AWS はアクセス拒否エラーメッセージに文字列 because no *type* policy allows the *action* action を含めます。

### Note

AWS のサービスによっては、このアクセス拒否エラーメッセージ形式をサポートしていません。アクセス拒否エラーメッセージの内容は、認証リクエストを行うサービスによって異なる場合があります。

次の例では、アクセス拒否エラーメッセージの形式のタイプを示します。

## サービスコントロールポリシーによるアクセスの拒否 - 暗黙的な拒否

1. サービスコントロールポリシー (SCP) のアクションで、Allow ステートメントが欠けていないかを確認します。次の例では、codecommit:ListRepositories がアクションです。
2. Allow ステートメントを追加して、ポリシーを更新してください。詳細については、「AWS Organizations ユーザーガイド」の「[SCP の更新](#)」を参照してください。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no service control policy allows the
codecommit:ListRespositories action
```

## サービスコントロールポリシーによるアクセスの拒否 - 明示的な拒否

1. サービスコントロールポリシー (SCP) のアクションで、Deny ステートメントを確認します。次の例では、codecommit:ListRepositories がアクションです。
2. Deny ステートメントを削除して、SCP を更新してください。詳細については、「AWS Organizations ユーザーガイド」の「[SCP の更新](#)」を参照してください。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories with an explicit deny in a service control policy
```

## VPC エンドポイントポリシーによるアクセスの拒否 - 暗黙的な拒否

1. 仮想プライベートクラウド (VPC) エンドポイントポリシーのアクションで、Allow ステートメントが欠落していないかを確認してください。次の例では、codecommit:ListRepositories がアクションです。
2. Allow ステートメントを追加して、VPC エンドポイントポリシーを更新します。詳細については、「AWS PrivateLink ガイド」の「[VPC エンドポイントポリシーを更新する](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no VPC endpoint policy allows the
codecommit:ListRepositories action
```



## VPC エンドポイントポリシーによるアクセスの拒否 – 明示的拒否

1. 仮想プライベートクラウド (VPC) エンドポイントポリシーにあるアクションに対する明示的 Deny ステートメントがあるのかを確認してください。次の例では、`codedeploy:ListDeployments` がアクションです。
2. Deny ステートメントを削除して、VPC エンドポイントポリシーを更新してください。詳細については、「AWS PrivateLink ガイド」の「[VPC エンドポイントポリシーを更新する](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a VPC endpoint policy
```

## アクセス許可の境界によりアクセスの拒否 – 暗黙的な拒否

1. アクセス許可の境界にあるアクションで、Allow ステートメントが欠落していないかを確認してください。次の例では、`codedeploy:ListDeployments` がアクションです。
2. IAM ポリシーに Allow ステートメントを追加して、アクセス許可の境界を更新してください。詳細については、「[IAM エンティティのアクセス許可境界](#)」および「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* because no permissions boundary allows the
codedeploy:ListDeployments action
```

## アクセス許可の境界によりアクセスの拒否 – 明示的拒否

1. アクセス許可の境界にあるアクションで、明示的な Deny ステートメントを確認してください。次の例では、`sagemaker:ListModels` がアクションです。
2. IAM ポリシーから Deny ステートメントを削除して、アクセス許可の境界を更新してください。詳細については、[IAM エンティティのアクセス許可境界](#)および[IAM ポリシーの編集](#)を参照してください。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
```

```
sagemaker:ListModels with an explicit deny in a permissions boundary
```

## セッションポリシーによるアクセスの拒否 – 暗黙的な拒否

1. セッションポリシーにあるアクションで、Allow ステートメントが欠落していないかを確認してください。次の例では、`codecommit:ListRepositories` がアクションです。
2. Allow ステートメントを追加して、セッションポリシーを更新してください。詳細については、「[セッションポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no session policy allows the
codecommit:ListRepositories action
```

## セッションポリシーによるアクセスの拒否 – 明示的拒否

1. セッションポリシーにあるアクションで、明示的な Deny ステートメントがあるのかを確認してください。次の例では、`codedeploy:ListDeployments` がアクションです。
2. Deny ステートメントを削除して、セッションポリシーを更新してください。詳細については、「[セッションポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a sessions policy
```

## リソースベースのポリシーによるアクセスの拒否 – 暗黙的な拒否

1. リソースベースのポリシーにあるアクションで、Allow ステートメントが欠落していないかを確認してください。次の例では、`secretsmanager:GetSecretValue` がアクションです。
2. Allow ステートメントを追加して、ポリシーを更新してください。詳細については、「[リソースベースのポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue because no resource-based policy allows the
secretsmanager:GetSecretValue action
```



## リソースベースのポリシーによるアクセスの拒否 – 明示的拒否

1. リソースベースのポリシーにあるアクションで、明示的な Deny ステートメントがあるのかを確認してください。次の例では、`secretsmanager:GetSecretValue` がアクションです。
2. Deny ステートメントを削除して、ポリシーを更新してください。詳細については、「[リソースベースのポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue on resource: arn:aws:secretsmanager:us-
east-1:123456789012:secret:* with an explicit deny in a resource-based policy
```

## ロール信頼ポリシーによるアクセスの拒否 – 暗黙的な拒否

1. ロール信頼ポリシーにあるアクションで、Allow ステートメントが欠落していないかを確認してください。次の例では、`sts:AssumeRole` がアクションです。
2. Allow ステートメントを追加して、ポリシーを更新してください。詳細については、「[リソースベースのポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
sts:AssumeRole because no role trust policy allows the sts:AssumeRole action
```

## ロール信頼ポリシーによるアクセスの拒否 – 明示的拒否

1. ロール信頼ポリシーにあるアクションで、明示的な Deny ステートメントがあるのかを確認してください。次の例では、`sts:AssumeRole` がアクションです。
2. Deny ステートメントを削除して、ポリシーを更新してください。詳細については、「[リソースベースのポリシー](#)」と「[IAM ポリシーの編集](#)」を参照してください。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sts:AssumeRole with an explicit deny in the role trust policy
```

## ID ベースのポリシーによるアクセスの拒否 – 暗黙的な拒否

1. ID にアタッチされた ID ベースのポリシーにあるアクションに対して、Allow ステートメントが欠落していないかを確認してください。次の例では、ユーザー JohnDoe にアタッチされた `codecommit:ListRepositories` がアクションです。
2. Allow ステートメントを追加して、ポリシーを更新してください。詳細については、「[アイデンティティベースのポリシー](#)」と [IAM ポリシーの編集](#) を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no identity-based policy allows the
codecommit:ListRepositories action
```

## ID ベースのポリシーによるアクセスの拒否 – 明示的拒否

1. ID にアタッチされた ID ベースのポリシーにあるアクションに対して、明示的な Deny ステートメントがあるのかを確認してください。次の例では、ユーザー JohnDoe にアタッチされた `codedeploy:ListDeployments` がアクションです。
2. Deny ステートメントを削除して、ポリシーを更新してください。詳細については、「[アイデンティティベースのポリシー](#)」と [IAM ポリシーの編集](#) を参照してください。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in an identity-based policy
```

## 別のポリシーが原因で VPC 要求が失敗した場合のアクセス拒否

1. サービスコントロールポリシー (SCP) にあるアクションで、明示的な Deny ステートメントがあるのかを確認してください。次の例では、`SNS:Publish` がアクションです。
2. Deny ステートメントを削除して、SCP を更新してください。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[SCP の更新](#)」を参照してください。

```
User: arn:aws:sts::111122223333:assumed-role/role-name/role-session-name is not
authorized to perform:
SNS:Publish on resource: arn:aws:sns:us-east-1:444455556666:role-name-2
```

with an explicit deny in a VPC endpoint policy transitively through a service control policy

## IAM ポリシーのトラブルシューティング

**ポリシー**は、ID やリソースにアタッチされるとそれらのアクセス許可を定義する、AWS のエンティティです。AWS は、ユーザーなどのプリンシパルがリクエストを行うときに、これらのポリシーを評価します。ポリシーでの許可により、リクエストが許可されるか拒否されるかが決まります。ポリシーはアイデンティティベースのポリシーとしてプリンシパルにアタッチされるか、リソースベースのポリシーとしてリソースにアタッチされた JSON ドキュメントとして AWS に保存されます。IAM グループ、ユーザー、ロールなどのプリンシパル (またはアイデンティティ) に、アイデンティティベースのポリシーをアタッチできます。アイデンティティベースのポリシーには、AWS 管理ポリシー、カスタマー管理ポリシー、およびインラインポリシーがあります。AWS Management Console で、[Visual] および [JSON] エディタオプションの両方を使用してカスタマー管理ポリシーの作成と編集ができます。AWS Management Consoleでポリシーを表示すると、そのポリシーによって付与されたアクセス許可の概要を表示できます。ビジュアルエディタとポリシー概要を使用して、IAM ポリシーの管理中に発生した一般的なエラーの診断と修正に役立てることができます。

すべての IAM ポリシーは、[JavaScript Object Notation \(JSON\)](#) のルールで始まる構文を使用して保存されることに注意してください。ポリシーを作成または管理するために、この構文を理解する必要はありません。AWS Management Consoleのビジュアルエディタを使用してポリシーを作成または編集できます。IAM ポリシーでの JSON 構文の詳細については、「[IAM JSON ポリシー言語の文法](#)」を参照してください。

### IAM ポリシーのトラブルシューティングに関するトピック

- [ビジュアルエディタを使用したトラブルシューティング](#)
  - [ポリシーの再構成](#)
  - [ビジュアルエディタでのリソース ARN の選択](#)
  - [ビジュアルエディタでのアクセス許可の拒否](#)
  - [ビジュアルエディタで複数のサービスの指定](#)
  - [ビジュアルエディタでのポリシーサイズの縮小](#)
  - [ビジュアルエディタでの認識されないサービス、アクション、またはリソースタイプの修正](#)
- [ポリシー概要を使用したトラブルシューティング](#)
  - [欠落しているポリシーの概要](#)
  - [ポリシー概要に認識されないサービス、アクション、リソースタイプが含まれる](#)

- [サービスが IAM ポリシー概要をサポートしていない](#)
- [使用するポリシーが予期するアクセス許可を付与しない](#)
- [ポリシー管理のトラブルシューティング](#)
  - [IAM アカウントでのポリシーのアタッチまたはデタッチ](#)
  - [アクティビティに基づいて IAM アイデンティティのポリシーを変更する](#)
- [JSON ポリシードキュメントのトラブルシューティング](#)
  - [ポリシーの検証](#)
  - [JSON エディタでポリシー検証のアクセス許可がありません](#)
  - [複数の JSON ポリシーオブジェクト](#)
  - [複数の JSON Statement 要素](#)
  - [JSON Statement 要素内の複数の Effect、Action、Resource 要素](#)
  - [JSON Version 要素の欠落](#)

## ビジュアルエディタを使用したトラブルシューティング

カスタマー管理ポリシーを作成または編集するときには、[ビジュアル] エディタの情報を使用して、ポリシーのエラーのトラブルシューティングに役立てることができます。ビジュアルエディタを使用してポリシーを作成する方法の例については、「[the section called “アイデンティティへのアクセスコントロール”](#)」を参照してください。

### ポリシーの再構成

ポリシーを作成すると、AWS はポリシーを検証、処理、変換してから保存します。AWS がユーザークエリに対してポリシーを返すか、コンソールに表示するとき、AWS はポリシーによって付与されたアクセス許可を変更することなく、ポリシーを人間が読み取れる形式に変換し直します。その結果、ポリシーのビジュアルエディタまたは [JSON] タブで表示されるポリシーに違いが発生する場合があります。ビジュアルエディタのアクセス許可ブロックは追加、削除、または順序変更でき、ブロック内のコンテンツは最適化できます。[JSON] タブで、重要でない空白は削除でき、JSON マップ内の要素は順序を変更できます。さらに、プリンシパル要素内の AWS アカウント ID は AWS アカウントのルートユーザーの ARN に置き換えることができます。このような変更の可能性があるため、JSON ポリシードキュメントを文字列として比較しないでください。

AWS Management Console でカスタマー管理ポリシーを作成するとき、[JSON] エディタで操作が完結するように選択できます。[Visual] エディタで一切変更を加えることなく、JSON タブで [次へ] を選択すると、ポリシーが再構成される可能性が低くなります。ただし、ポリシーを作成し、

[Visual] エディタを使用して変更を加えるか、[Visual] エディタから [次へ] を選択する場合、IAM はビジュアルエディタでの外観を最適化するために、ポリシーを再構成する可能性があります。

この再構成は編集セッション内のみ存在するものであり、自動的に保存されません。

編集セッションでポリシーが再構成された場合、IAM は次の状況に基づいて再構成を保存するかどうかを決定します。

このエディターオプションの使用	ポリシーを編集する場合	次に、このタブから [次へ] を選択します	[変更の保存] を選択する場合
Visual	編集されます	Visual	ポリシーが再構成されます
Visual	編集されます	JSON	ポリシーが再構成されます
Visual	編集されません	Visual	ポリシーが再構成されます
JSON	編集されます	Visual	ポリシーが再構成されます
JSON	編集されます	JSON	ポリシー構造は変更されません
JSON	編集されません	JSON	ポリシー構造は変更されません

IAM は、複雑なポリシーや、複数のサービス、リソースタイプ、または条件キーを許可するアクセス許可ブロックまたはステートメントを持つポリシーを再構成する場合があります。

## ビジュアルエディタでのリソース ARN の選択

ビジュアルエディタを使用してポリシーを作成または編集するときは、最初にサービスを選択し、そのサービスからアクションを選択する必要があります。選択したサービスとアクションが [特定のリソース](#) の選択をサポートしている場合、ビジュアルエディタに、サポートされるリソースタイプが一覧表示されます。次に、[Add ARN (ARN の追加)] を選択して、リソースの詳細を提供できます。リソースタイプの ARN を追加するために、以下のオプションから選択できます。

- ARN ビルダの使用 – リソースタイプに基づいて、ARN を構築するためのさまざまなフィールドが表示される場合があります。[Any (任意)] を選択して、指定した設定の任意の値に対するアクセス許可を付与することもできます。たとえば、Amazon EC2 の [Read (読み取り)] アクセスレベルグループを選択した場合、ポリシーのアクションは instance リソースタイプをサポートします。リソースの [Region (リージョン)]、[Account (アカウント)]、[インスタンス ID] の各値を指定する必要があります。アカウント ID を指定するが、リージョンとインスタンス ID に [Any (任意)] を選択する場合、ポリシーでアカウントの任意のインスタンスにアクセス許可が付与されます。
- ARN の入力または貼り付け – [Amazon リソースネーム \(ARN\)](#) によりリソースを指定できます。ワイルドカード文字 (\*) を ARN の任意のフィールドに含めることができます (コロンの各ペアの間)。詳細については、「[IAM JSON ポリシー要素Resource](#)」を参照してください。

## ビジュアルエディタでのアクセス許可の拒否

デフォルトでは、ビジュアルエディタを使用して作成するポリシーにより、選択したアクションが許可されます。その代わりに選択したアクションを拒否するには、[Switch to deny permissions (アクセス許可の拒否に切り替え)] を選択します。リクエストはデフォルトでは拒否されるため、ユーザーが必要とするアクションとリソースに対するアクセス許可のみを付与することを、セキュリティのベストプラクティスとしてお勧めします。別のステートメントまたはポリシーによって個別に許可されるアクセス許可を上書きする場合のみ、アクセス許可を拒否するステートメントを作成する必要があります。これにより、アクセス許可のトラブルシューティングがより困難になる可能性があるため、拒否ステートメントの数は最小限に制限することをお勧めします。IAM がポリシーのロジックを評価する方法の詳細については、「[ポリシーの評価論理](#)」を参照してください。

### Note

デフォルトでは、AWS アカウントのルートユーザーのみが、そのアカウントのすべてのリソースにアクセスできます。ワイルドカード文字 () を ARN の任意のフィールドに含めることができます (コロンの各ペアの間)。

## ビジュアルエディタで複数のサービスの指定

ビジュアルエディタを使用してポリシーを作成するときに、一度に 1 つのサービスのみを選択できます。これは、その後ビジュアルエディタではその 1 つのサービス用のアクションから選択できるため、ベストプラクティスとなります。次に、そのサービスと選択されたアクションでサポートされているリソースから選択できます。これにより、ポリシーの作成とトラブルシューティングが容易になります。



JSON 構文に精通している場合は、ワイルドカード文字 (\*) を使用して複数のサービスを手動で指定することもできます。たとえば、「**Code\***」と入力すると、CodeBuild や CodeCommit など、Code で始まるすべてのサービスに対するアクセス許可を付与できます。ただし、次にアクションとリソースの ARN を入力して、ポリシーを完了する必要があります。さらに、ポリシーを保存すると、各サービスが別のアクセス許可ブロックに含まれるように、ポリシーが**再構成**される場合があります。

または、JSON 構文 (ワイルドカードなど) をサービスに使用するには、[JSON] エディタオプションを使用してポリシーを作成、編集、保存します。

## ビジュアルエディタでのポリシーサイズの縮小

ビジュアルエディタを使用してポリシーを作成する場合、IAM はポリシーを保存する JSON ドキュメントを作成します。このドキュメントを表示するには、[JSON] エディタオプションに切り替えます。この JSON ドキュメントがポリシーのサイズ制限を超える場合、ビジュアルエディタにはエラーメッセージが表示され、ポリシーを確認または保存することはできません。管理ポリシーのサイズの IAM 制限を表示する方法については、「[IAM 文字制限および STS 文字制限](#)」を参照してください。

ビジュアルエディタでポリシーのサイズを減らすには、ポリシーを編集するか、アクセス許可ブロックを別のポリシーに移動します。エラーメッセージには、ポリシードキュメントに含まれている文字数が含まれ、この情報を使用してポリシーのサイズを縮小することができます。

## ビジュアルエディタでの認識されないサービス、アクション、またはリソースタイプの修正

ビジュアルエディタでポリシーを作成または編集すると、認識できないサービス、アクション、またはリソースタイプがポリシーに含まれているという警告が表示される場合があります。

### Note

IAM によって、ポリシー概要をサポートするサービスの名前、アクション、リソースタイプが確認されます。ただし、ポリシー概要には、存在しないリソース値または条件が含まれる可能性があります。必ず [Policy Simulator](#) でポリシーをテストしてください。

ポリシーに認識されないサービス、アクション、またはリソースタイプが含まれている場合は、以下のいずれかのエラーが発生しています。

- プレビューサービス – プレビュー中のサービスはビジュアルエディタをサポートしていません。プレビューに参加中の場合は、警告を無視して続行できます。ただし、ポリシーを完了するには、アクションとリソースの ARN を手動で入力する必要があります。または、[JSON] エディタオプションを選択して、JSON ポリシードキュメントを入力または貼り付けることができます。
- カスタムサービス – カスタムサービスはビジュアルエディタをサポートしていません。カスタムサービスを使用中の場合は、警告を無視して続行できます。ただし、ポリシーを完了するには、アクションとリソースの ARN を手動で入力する必要があります。または、[JSON] エディタオプションを選択して、JSON ポリシードキュメントを入力または貼り付けることができます。
- サービスがビジュアルエディタをサポートしていない – ポリシーに、ビジュアルエディタをサポートしていない一般公開された (GA) サービスが含まれている場合、警告を無視して続行できます。ただし、ポリシーを完了するには、アクションとリソースの ARN を手動で入力する必要があります。または、[JSON] エディタオプションを選択して、JSON ポリシードキュメントを入力または貼り付けることができます。

一般公開されたサービスとは、プレビューやカスタムサービスではない、一般にリリースされたサービスです。認識されないサービスが一般公開されていて、名前のスペルが正しくない場合、そのサービスはビジュアルエディタをサポートしません。GA サービスに対するビジュアルエディタまたはポリシー概要のサポートをリクエストする方法については、「[サービスが IAM ポリシー概要をサポートしていない](#)」を参照してください。

- アクションがビジュアルエディタをサポートしていない – ポリシーに含まれるサービスがサポートされていても、アクションがサポートされていない場合は、警告を無視して続行できます。ただし、ポリシーを完了するには、リソースの ARN を手動で入力する必要があります。または、[JSON] エディタオプションを選択して、JSON ポリシードキュメントを入力または貼り付けることができます。

ポリシーに含まれるサービスがサポートされていても、アクションがサポートされていない場合、そのサービスはビジュアルエディタを一部サポートしません。GA サービスに対するビジュアルエディタまたはポリシー概要のサポートをリクエストする方法については、「[サービスが IAM ポリシー概要をサポートしていない](#)」を参照してください。

- リソースタイプがビジュアルエディタをサポートしていない – ポリシーに含まれるアクションがサポートされていても、リソースタイプがサポートされていない場合は、警告を無視して続行できます。ただし、IAM は、選択したすべてのアクションのリソースを含めたことを確認できないため、追加の警告が表示されることがあります。
- タイプミス – ビジュアルエディタでサービス、アクション、またはリソースを手動で入力するときに、タイプミスを含むポリシーを作成する可能性があります。これを避けるために、ビジュアルエディタを使用してサービスとアクションのリストから項目を選択し、プロンプトに従ってリソー



スエクシジョンの操作を完了させることをお勧めします。ただし、サービスがビジュアルエディタを完全にサポートしていない場合は、ポリシーの一部を手動で入力しなければならないことがあります。

ポリシーに、確実に上記のエラーが含まれていない場合は、ポリシーにタイプミスが含まれている可能性があります。サービス、アクション、およびリソースタイプの名前にタイプミスがないか確認してください。たとえば、s2 ではなく s3 を使用したり、ListMyBuckets ではなく ListAllMyBuckets を使用する場合があります。アクションによくある別のタイプミスは、arn:aws:s3: : :\* など、ARN に不要なテキストが含まれているものや、iam.CreateUser など、アクションにコロンがないものです。タイプミスを含む可能性のあるポリシーは、[次へ] を選択してポリシー概要を表示し、そのポリシーが意図したアクセス許可を付与するかどうかを確認することにより評価できます。

## ポリシー概要を使用したトラブルシューティング

ポリシー概要に関連する問題を診断して解決できます。

### 欠落しているポリシーの概要

IAM コンソールには、ポリシー内の各サービスに対して許可または拒否されるアクセスレベル、リソース、条件を定義するポリシー概要テーブルが含まれます。ポリシーは、[ポリシー概要](#)、[サービス概要](#)、[アクション概要](#)の3つのテーブルにまとめられています。ポリシー概要テーブルには、選択したポリシーによって定義されているサービスとアクセス許可の概要のリストが含まれます。エンティティにアタッチされているポリシーの[ポリシーの概要](#)は、そのポリシーの[ポリシーの詳細]ページで表示できます。[ポリシー]ページで、管理ポリシーのポリシー概要を表示できます。AWS がポリシー概要をレンダリングできない場合は、概要ではなく JSON ポリシードキュメントが表示され、次のエラーが表示されます。

[A summary for this policy cannot be generated.] (このポリシー概要を生成できません。) [You can still view or edit the JSON policy document.] (JSON ポリシードキュメントは引き続き表示または編集できます。)

ポリシーに要約が含まれていない場合は、次のいずれかのエラーが発生しています。

- サポート対象外のポリシー要素 – IAM では、以下の[ポリシー要素](#)のうちいずれかを含むポリシー概要の生成がサポートされていません。
  - Principal
  - NotPrincipal

- NotResource
- ポリシーのアクセス許可なし – ポリシーによって有効なアクセス許可が付与されない場合、ポリシー概要を生成できません。たとえば、ポリシーに含まれる1つのステートメントでエリメント "NotAction": "\*" がある場合です。この場合、すべてのアクション (\*) を除くすべてのアクションへのアクセスを許可しています。つまり、Deny または Allow のいずれにもアクセスを付与しません。

#### Note

NotPrincipal、NotAction、NotResource など、これらのポリシーエリメントは慎重に使用する必要があります。ポリシーエリメントの使用に関する詳細については、「[IAM JSON ポリシー要素のリファレンス](#)」を参照してください。

作成したポリシーで、サービスとリソースが一致していないと、有効なアクセス権限が付与されないことがあります。たとえば、あるサービスのアクションを指定し、別のサービスのリソースを指定した場合です。この場合は、ポリシー概要が表示されます。概要のリソース列に別のサービスのリソースが含まれていることからしか、問題があることはわかりません。この列に不一致のリソースが含まれる場合は、ポリシーのエラーを確認する必要があります。ポリシーをより深く理解するために、必ず [Policy Simulator](#) でテストしてください。

## ポリシー概要に認識されないサービス、アクション、リソースタイプが含まれる

IAM コンソールで、[ポリシー概要](#)に警告記号



が含まれる場合、ポリシーに認識されないサービス、アクション、またはリソースタイプが含まれている可能性があります。ポリシー概要内での警告については、「[ポリシー概要 \(サービスの一覧\)](#)」を参照してください。

#### Note

IAM によって、ポリシー概要をサポートするサービスの名前、アクション、リソースタイプが確認されます。ただし、ポリシー概要には、存在しないリソース値または条件が含まれる可能性があります。必ず [Policy Simulator](#) でポリシーをテストしてください。

ポリシーに認識されないサービス、アクション、またはリソースタイプが含まれている場合は、以下のいずれかのエラーが発生しています。

- プレビューサービス – プレビュー中のサービスはポリシー概要をサポートしていません。
- カスタムサービス – カスタムサービスはポリシー概要をサポートしていません。
- サービスが概要をサポートしていない – |ポリシー概要をサポートしていない一般公開された (GA) サービスがポリシーに含まれる場合、そのサービスはポリシー概要テーブルの [Unrecognized services (認識されないサービス)] セクションに含まれます。一般公開されたサービスとは、プレビューやカスタムサービスではない、一般にリリースされたサービスです。認識されないサービスが一般公開されていて、名前のスペルが正しくない場合、そのサービスは IAM ポリシー概要をサポートしません。GA サービスに対するポリシー概要のサポートをリクエストする方法については、「[サービスが IAM ポリシー概要をサポートしていない](#)」を参照してください。
- アクションが概要をサポートしていない – ポリシーに含まれるサービスがサポートされていても、アクションがサポートされていない場合、そのアクションはサービス概要テーブルの [Unrecognized actions (認識されないアクション)] セクションに含まれます。サービス概要内での警告については、「[サービス概要 \(アクションのリスト\)](#)」を参照してください。
- リソースタイプが概要をサポートしていない – ポリシーに含まれるアクションがサポートされていても、リソースタイプがサポートされていない場合、そのリソースはサービス概要テーブルの [認識されないリソースタイプ] セクションに含まれます。サービス概要内での警告については、「[サービス概要 \(アクションのリスト\)](#)」を参照してください。
- タイプミス – AWS は、JSON が構文的に正しいこと、およびポリシーにタイプミスや[ポリシーの検証](#)の一部としてのその他のエラーが含まれていないことをチェックします。。

#### Note

[ベストプラクティス](#)として、IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的なアクセス許可を確保することをお勧めします。既存のポリシーを開き、ポリシーの検証レコメンデーションを確認して解決することをお勧めします。

## サービスが IAM ポリシー概要をサポートしていない

一般公開された (GA) サービスまたはアクションが IAM のポリシー概要またはビジュアルエディタで認識されない場合、サービスがこれらの機能をサポートしていない可能性があります。一般公開されたサービスとは、プレビューやカスタムサービスではない、一般にリリースされたサービスです。認識されないサービスが一般公開されていて、名前のスペルが正しくない場合、そのサービスはこれら

の機能をサポートしません。ポリシーに含まれるサービスがサポートされていても、アクションがサポートされていない場合、そのサービスは IAM ポリシー概要を一部サポートしません。

IAM ポリシー概要またはビジュアルエディタのサポートがサービスに追加されるようにリクエストするには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. サポートされていないサービスを含むポリシーを特定する
  - ポリシーが管理ポリシーの場合は、ナビゲーションペインの [ポリシー] を選択します。ポリシーの一覧で、表示するポリシーの名前を選択します。
  - ポリシーが、ユーザーにアタッチされているインラインポリシーの場合は、ナビゲーションペインの [ユーザー] を選択します。ユーザーのリストから、ポリシーを表示するユーザーを選択します。ユーザーのポリシーのテーブルで、表示するポリシー概要のヘッダーを展開します。
3. AWS Management Console のフッターの左側にある [Feedback (フィードバック)] を選択します。[IAM へのフィードバック] ボックスに、**I request that the <ServiceName> service add support for IAM policy summaries and the visual editor** と入力します。複数のサービスで概要をサポートする場合は、「**I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor**」と入力します。

サービスの欠落したアクション向けに IAM ポリシー概要のサポートが適用されるようにリクエストするには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. サポートされていないサービスを含むポリシーを特定する
  - ポリシーが管理ポリシーの場合は、ナビゲーションペインの [ポリシー] を選択します。ポリシーの一覧で、表示するポリシーの名前を選択します。
  - ポリシーが、ユーザーにアタッチされているインラインポリシーの場合は、ナビゲーションペインの [ユーザー] を選択します。ユーザーのリストから、ポリシーを表示するユーザーを選択します。ユーザーのポリシーのテーブルで、ポリシー概要を展開するために表示するポリシー名を選択します。
3. ポリシー概要で、サポートされていないアクションを含むサービスの名前を選択します。

4. AWS Management Console のフッターの左側にある [Feedback (フィードバック)] を選択します。[IAM へのフィードバック] ボックスに、**I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action** と入力します。サポートされていない複数のアクションをレポートする場合は、「**I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions**」と入力します。

不足しているアクションを別のサービスに含めるようにリクエストするには、最後の 3 つの手順を繰り返します。

### 使用するポリシーが预期するアクセス許可を付与しない

ユーザー、グループ、ロール、リソースにアクセス許可を割り当てるには、ポリシーを作成します。このポリシーは、アクセス許可を定義したドキュメントです。ポリシードキュメントには以下の要素が含まれます。

- Effect – ポリシーがアクセスを許可または拒否するかどうか
- Action – ポリシーによって許可または拒否されるアクションのリスト
- Resource – アクションを起こすことができるリソースのリスト
- Condition (オプション) ポリシーがアクセス許可を付与する状況

これらのポリシーの要素の詳細または他の要素については「[IAM JSON ポリシー要素のリファレンス](#)」を参照してください。

アクセスを許可するには、サポートされたリソースでポリシーにアクションを定義する必要があります。ポリシーに条件も含まれている場合は、条件に[グローバル条件キー](#)が含まれている必要があります。または、条件をアクションに適用する必要があります。アクションでサポートされているリソースを確認するには、サービスの [AWS ドキュメント](#) を参照してください。アクションでサポートされている条件については、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

アクセス許可を付与しないアクション、リソース、条件がポリシーで定義されていないかは、[IAM コンソール \(\)](#) で [ポリシー概要](#) を表示すると確認できます。ポリシー概要を使用すると、ポリシーの問題を特定して解決することができます。

次のような理由により、IAM ポリシーで定義されているにもかかわらず、要素がアクセス許可を付与しない場合があります。

- [該当するリソースのないアクションが定義されている](#)
- [該当するアクションのないリソースが定義されている](#)
- [該当するアクションのない条件が定義されている](#)

警告を含むポリシー概要の例を表示するには、「[the section called “ポリシー概要 \(サービスの一覧\)”](#)」を参照してください。

#### 該当するリソースのないアクションが定義されている

以下のポリシーは、すべての `ec2:Describe*` アクションを定義し、特定のリソースを定義します。すべての `ec2:Describe` アクションにはサポートされるリソースレベルのアクセス許可がないため、アクセスが付与されません。リソースレベルのアクセス許可とは、ポリシーの [Resource](#) 要素で [ARN](#) を使用するリソースをアクションがサポートしていることを意味します。アクションがリソースレベルのアクセス許可をサポートしない場合、ポリシーのステートメントは \* エlement でワイルドカード (Resource) を使用する必要があります。リソースレベルのアクセス許可をサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "arn:aws:ec2:us-east-2:ACCOUNT-ID:instance/*"
  ]
}
```

このポリシーはアクセス許可を指定せず、ポリシー概要には次のエラーが表示されます。

This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.

このポリシーを修正するには、\* エlement に Resource を使用する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  ]
}
```

```
}
```

該当するアクションのないリソースが定義されている

以下のポリシーは Amazon S3 バケットリソースを定義していますが、そのリソースで実行可能な S3 アクションは含まれていません。このポリシーはすべての Amazon CloudFront アクションに対するフルアクセスも供与しています。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "cloudfront:*",
    "Resource": [
      "arn:aws:cloudfront:*",
      "arn:aws:s3:::examplebucket"
    ]
  }]
}
```

このポリシーではすべての CloudFront アクションに対するアクセス許可が与えられています。しかし、ポリシーが S3 アクションを定義せずに S3 examplebucket リソースを定義しているために、ポリシー概要では次のように警告されます。

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition.

このポリシーを修正して S3 バケットのアクセス許可を与えるために、バケットリソースで実行可能な S3 アクションを定義する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "cloudfront:*",
      "s3:CreateBucket",
      "s3:ListBucket*",
      "s3:PutBucket*",
      "s3:GetBucket*"
    ]
  }]
}
```



```
    ],
    "Resource": [
      "arn:aws:cloudfront:*",
      "arn:aws:s3:::examplebucket"
    ]
  }]
}
```

あるいは、CloudFront アクセス許可のみが提供されるように、S3 リソースを削除してこのポリシーを修正してください。

該当するアクションのない条件が定義されている

以下のポリシーは、S3 プレフィックスが `custom` でバージョン ID が `1234` の場合に、すべての S3 リソースに対して 2 つの Amazon S3 アクションを定義しています。しかし、`s3:VersionId` 条件キーは、オブジェクトバージョンのタグ付けに使用され、定義済みのバケットアクションではサポートされていません。アクションでサポートされている条件については、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照して、条件キーのサービスドキュメントのリンクをクリックしてください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": [
            "custom"
          ],
          "s3:VersionId": [
            "1234"
          ]
        }
      }
    }
  ]
}
```



```
}
```

このポリシーでは `s3:ListBucketVersions` アクションに対するアクセス許可が与えられ、さらに、バケット名に `s3:ListBucket` プレフィックスが含まれている場合、`custom` アクションに対するアクセス許可が与えられています。しかし、定義済みのアクションでは `s3:VersionId` 条件がサポートされていないために、ポリシー概要には次のエラーが表示されます。

This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.

S3 オブジェクトのバージョンをタグ付けできるようにこのポリシーを修正するには、`s3:VersionId` 条件キーをサポートする S3 アクションを定義する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:GetObjectVersion"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": [
            "custom"
          ],
          "s3:VersionId": [
            "1234"
          ]
        }
      }
    }
  ]
}
```

このポリシーでは、ポリシー内のすべてのアクションと条件に対するアクセス許可が提供されています。それにもかかわらず、ポリシーからはどのアクセス許可もいまだに指定されていません。これは、1つのアクションで両方の条件に一致しているケースがないためです。この場合は、適用する条件付きの専用アクションが含まれている、個別のステートメントを2つ作成する必要があります。

このポリシーを修正するには、次のようにステートメントを2つ作成します。最初のステートメントには `s3:prefix` 条件をサポートするアクションが含まれるようにし、2番目のステートメントには `s3:VersionId` 条件をサポートするアクションが含まれるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": "custom"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObjectVersion",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:VersionId": "1234"
        }
      }
    }
  ]
}
```

## ポリシー管理のトラブルシューティング

ポリシー管理に関連する問題を診断して解決できます。

### IAM アカウントでのポリシーのアタッチまたはデタッチ

一部の AWS 管理ポリシーはサービスにリンクされています。これらのポリシーは、そのサービス用に作成した[サービスにリンクされたロール](#)でのみ使用されます。IAM コンソールでは、ポリシーの [ポリシー詳細] ページを表示すると、そのポリシーがサービスにリンクされていることを示すバナー

が、そのページに表示されます。このポリシーは IAM 内でユーザー、グループ、またはロールにアタッチできます。サービスにリンクされたロールをサービス用に作成すると、このポリシーは新しいロールに自動的にアタッチされます。ポリシーは必須であるため、サービスにリンクされたロールからポリシーをデタッチすることはできません。

## アクティビティに基づいて IAM アイデンティティのポリシーを変更する

IAM アイデンティティ (ユーザー、グループ、ロール) のポリシーは、アクティビティに基づいて更新することができます。これを行うには、CloudTrail イベント履歴でアカウントのイベントを表示します。CloudTrail イベントログには、ポリシーのアクセス許可の変更に使用できる詳細なイベント情報が含まれます。ユーザーまたはロールが AWS でアクションを実行しようとして、そのリクエストが拒否される場合があります。この場合、ユーザーまたはロールにアクションを実行するためのアクセス許可が必要かどうかを検討します。その場合、アクションだけでなく、ポリシーにアクセスしようとしたリソースの ARN を追加することができます。または、使用していないアクセス許可がユーザーまたはロールにある場合は、ポリシーからそのようなアクセス許可を削除することも検討してください。ポリシーにより、必要なアクションのみの実行に必要な[最小限の権限](#)が付与されていることを確認します。CloudTrail の使用の詳細については、[AWS CloudTrail ユーザーガイド](#)の「CloudTrail コンソールでの CloudTrail イベントの表示の」を参照してください。

## JSON ポリシードキュメントのトラブルシューティング

JSON ポリシードキュメントに関連する問題を診断して解決できます。

### ポリシーの検証

JSON ポリシーを作成または編集するときに、IAM はポリシー検証を実行し、効果的なポリシーを作成するのに役立ちます。IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックを提供します。ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。IAM Access Analyzer のポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

### JSON エディタでポリシー検証のアクセス許可がありません

AWS Management Console で、IAM Access Analyzer ポリシーの検証結果を表示するアクセス許可がない場合、次のエラーが表示されることがあります。

```
You need permissions. You do not have the permissions required to perform this operation. Ask your administrator to add permissions.
```

このエラーを解決するには、自分に `access-analyzer:ValidatePolicy` アクセス許可を追加するよう管理者に依頼します。

## 複数の JSON ポリシーオブジェクト

IAM ポリシーは、1 つの JSON オブジェクトのみで構成する必要があります。オブジェクトは括弧 ( `{}` ) で囲んで示します。外側の `{}` のペア内に追加の `{}` を埋め込むことによって JSON オブジェクト内で他のオブジェクトをネストすることができますが、ポリシーに含めることができるのは一番外側の `{}` のペアのみです。以下の例は、最上位に 2 つのオブジェクト (#で示した箇所) が含まれているので誤りです。

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  }
}
{
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::my-bucket/*"
  }
}
```

ただし、正しいポリシーの文法を使用して、前述の例の目的を果たすことができます。それぞれに独自の Statement エレメントを含む 2 つのポリシーオブジェクトを含める代わりに、1 つの Statement エレメントに 2 つのブロックを組み合わせる使用することができます。Statement 要素には、以下の例 (太字で示した箇所) に示すように 2 つのオブジェクトの配列を値として指定します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": " *"
  ]
}
```

```
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::my-bucket/*"
    }
  ]
}
```

## 複数の JSON Statement 要素

このエラーは、一見、前のセクションのバリエーションのように見えますが、構文上はエラーの種類が異なります。次の例には、最上位の 1 ペアの {} で示された 1 つのポリシーオブジェクトのみが存在します。そのオブジェクト内に 2 つの Statement 要素が含まれています。

IAM ポリシーには、名前 (Statement) の後にコロン、その後に値という形式で構成される 1 つの Statement 要素のみを指定する必要があります。Statement 要素の値は、{} で示され、1 つの Effect 要素、1 つの Action 要素、および 1 つの Resource 要素を含むオブジェクトである必要があります。以下の例は、ポリシーオブジェクト (# で示した箇所) に 2 つの Statement 要素が含まれているので誤りです。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  },
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::my-bucket/*"
  }
}
```

値のオブジェクトは複数の値がオブジェクトの配列でもかまいません。この問題を解決するには、以下の例 (太字で示した箇所) に示すように、2 つの Statement 要素を 1 つの要素に合わせます。

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
```

```
    "Action": "ec2:Describe*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "s3:* ",
    "Resource": "arn:aws:s3:::my-bucket/*"
  }
]
```

Statement 要素の値はオブジェクト配列です。この例の配列は 2 つのオブジェクトで構成され、各オブジェクトに Statement エlement の正しい値が指定されています。配列内の各オブジェクトはカンマで区切ります。

## JSON Statement 要素内の複数の Effect、Action、Resource 要素

Statement の名前と値のペアの値側のオブジェクトは 1 つの Effect エlement、1 つの Action エlement、1 つの Resource エlement のみで構成する必要があります。次のポリシーは、Effect の値オブジェクトに 2 つの Statement エlement が含まれているので誤りです。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Effect": "Allow",
    "Action": "ec2:* ",
    "Resource": "*"
  }
}
```

### Note

ポリシーエンジンでは、新規ポリシーまたは編集済みのポリシーでそのようなエラーを許可することはできません。ただし、ポリシーエンジンは、エンジンが更新される前に保存されたポリシーを引き続き許可します。エラーが存在する既存のポリシーの動作は次のとおりです。

- 複数の Effect エlement: 最後の Effect エlement のみが確認され、それ以外は無視されます。

- 複数の Action エlement: すべての Action エlement が内部的に結合され、単一のリストとして処理されます。
- 複数の Resource エlement: すべての Resource エlement が内部的に結合され、単一のリストとして処理されます。

ポリシーエンジンでは、構文エラーのあるポリシーを保存することはできません。保存する前にポリシーのエラーを修正する必要があります。ポリシーの正しい[ポリシー検証](#)のレコメンドーションを確認することを推奨します。

いずれの場合も、解決策は誤って追加した要素を削除することです。Effect 要素の場合は簡単です。前述の例で Amazon EC2 インスタンスへのアクセス許可を拒否するには、以下のように、ポリシーから "Effect": "Allow", の行を削除する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "ec2:* ",
    "Resource": "*"
  }
}
```

重複している Element が Action または Resource の場合、解決法はより複雑になることがあります。アクセス権限を許可 (または拒否) するアクションが複数存在したり、複数のリソースへのアクセスを制御したりすることが必要な場合があります。たとえば、以下の例 (#で示した箇所) は、複数の Resource 要素が含まれているので誤りです。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:* ",
    "Resource": "arn:aws:s3::my-bucket",
    "Resource": "arn:aws:s3::my-bucket/*"
  }
}
```

Statement エlementの値オブジェクトに必要なElementは、それぞれ1つだけ含めることができます。この問題を解決するには、各値を配列に指定します。以下の例は、値オブジェクト形式の配列 (太字で示した箇所) を使用して2つの異なるリソース要素を1つのResource 要素として扱う方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3::my-bucket",
      "arn:aws:s3::my-bucket/*"
    ]
  }
}
```

## JSON Version 要素の欠落

Version ポリシー要素は、ポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。一方で、ポリシーバージョンは、IAM でカスタマー管理ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。Version ポリシー要素の詳細については、「[IAM JSON ポリシー要素Version](#)」を参照してください。ポリシーのバージョンの詳細については、「[the section called “IAM ポリシーのバージョンング”](#)」を参照してください。

AWS 機能の拡張にともない、この機能をサポートする新しい機能が IAM ポリシーに追加されました。ポリシー構文を更新すると、新しいバージョン番号が含まれている場合があります。ポリシーでポリシー文法の新しい機能を使用する場合は、使用しているバージョンをポリシー解析エンジンに通知する必要があります。デフォルトのポリシーのバージョンは "2008-10-17" です。後から導入されたポリシー機能を使用する場合は、使用する機能をサポートするバージョン番号を指定する必要があります。必ず最新のポリシー構文のバージョン番号 (現時点では "Version": "2012-10-17") を指定することをお勧めします。たとえば、次のポリシーは、リソースの ARN でポリシー変数 `${...}` を使用しているので誤りです。ただし、ポリシー変数をサポートするポリシー構文のバージョンが指定されていません (`#`で表示されています)。

```
{
  "Statement":
```



```
{
  "Action": "iam:*AccessKey*",
  "Effect": "Allow",
  "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
}
```

ポリシーの先頭に `Version` エlementとその値 `2012-10-17` (ポリシー変数をサポートする IAM API の最初のバージョン) を追加することで、この問題 (太字で示した箇所) を解決します。

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

## FIDO セキュリティキーのトラブルシューティング

この情報を使用して、FIDO2 セキュリティキーを操作するときに発生する可能性がある一般的な問題の診断を行います。

### トピック

- [FIDO セキュリティキーを有効にできない](#)
- [自分の FIDO セキュリティキーを使用してサインインできない](#)
- [FIDO セキュリティキーの紛失または破損した場合](#)
- [その他の問題](#)

### FIDO セキュリティキーを有効にできない

IAM ユーザーまたはシステム管理者としてのステータスに応じて、以下の解決策を確認します。

#### IAM ユーザー

FIDO セキュリティキーを有効にできない場合は、以下の点を確認します。

- サポートされている設定を使用しているか？

WebAuthn と AWS で使用できるデバイスとブラウザについては、「[パスキーとセキュリティキーを使用するためのサポートされる設定](#)」を参照してください。

- Mozilla Firefox を使用しているか？

現在の Firefox バージョンは、デフォルトで WebAuthn をサポートしています。Firefox で WebAuthn のサポートを有効にするには、以下の操作を行います。

1. Firefox のアドレスバーに「**about:config**」と入力します。
2. 表示される画面の検索バーに「**webauthn**」と入力します。
3. security.webauth.webAuthn を選択し、その値を [true] (真) に変更します。

- 任意のブラウザプラグインを使用しているか？

AWS は、WebAuthn ブラウザのサポート WebAuthn を追加するためのプラグインの使用をサポートしていません。代わりに、WebAuthn 標準のネイティブサポートを提供するブラウザを使用します。

サポートされているブラウザを使用している場合でも、そのプラグインが WebAuthn と互換性のない可能性があります。互換性のないプラグインにより、FIDO 準拠のセキュリティキーを有効にして使用できなくなる場合があります。互換性のないプラグインを無効にして、ブラウザを再起動する必要があります。その後、FIDO セキュリティキーを再び有効にしてみます。

- 適切なアクセス許可があるか？

上記の互換性の問題がない場合は、適切なアクセス許可がない可能性があります。システム管理者に連絡してください。

## システム管理者

お客様が管理者であり、IAM ユーザーがサポートされている設定を使用している場合でも FIDO セキュリティキーを有効にできない場合は、それらのユーザーが適切なアクセス許可を持っていることを確認します。詳細例については、「[IAM チュートリアル: ユーザーに自分の認証情報および MFA 設定を許可する](#)」を参照してください。

## 自分の FIDO セキュリティキーを使用してサインインできない

お客様が IAM ユーザーであり、FIDO セキュリティキーを使用して AWS Management Console にサインインできない場合は、まず「[パスキーとセキュリティキーを使用するためのサポートされる設](#)

定」を参照してください。サポートされている設定を使用しているにもかかわらずサインインできない場合は、システム管理者に連絡してください。

## FIDO セキュリティキーの紛失または破損した場合

[現在サポートされている MFA タイプ](#)を任意に組み合わせた最大 8 台の MFA デバイスを 1 人のユーザーに割り当てることができます。複数の MFA デバイスがあっても、AWS Management Console にサインインするのに必要なのは 1 台の MFA デバイスだけです。FIDO セキュリティキーを入れ替えることは、ハードウェア TOTP トークンを入れ替えることに似ています。任意のタイプの MFA デバイスを紛失したり破損したりした場合の対処方法については、「[MFA デバイスの紛失および故障時の対応](#)」を参照してください。

## その他の問題

ここで説明されていない FIDO セキュリティキーの問題がある場合は、以下の対処を行います。

- IAM ユーザー: システム管理者にお問い合わせください。
- AWS アカウント ルートユーザー: [AWS サポート](#)にお問い合わせください。

## IAM ロールのトラブルシューティング

この情報を使用して、IAM ロールを操作するときに発生する可能性がある一般的な問題の診断や修復を行います。

### トピック

- [ルールを引き受けることができない](#)
- [AWS アカウントに新しいルールが表示される](#)
- [自分の AWS アカウント でルールを編集または削除できない](#)
- [次のことを実行する権限がない: iam:PassRole](#)
- [12 時間のセッションに使用するルールを引き受けることができない \(AWS CLI、AWS API\)](#)
- [IAM コンソールでルールを切り替えようとするエラーが発生する](#)
- [ルールには、アクションの実行を許可するポリシーがあるが、「アクセスが拒否されました」というメッセージが表示される](#)
- [サービスがロールのデフォルトのポリシーバージョンを作成しなかった](#)
- [コンソールにサービスロールのユースケースがない](#)

## ロールを引き受けることができない

以下をチェックしてください:

- ロールセッション内でユーザーが現在のロールを再び引き受けることができるようにするには、ロール信頼ポリシーでロール ARN または AWS アカウント ARN をプリンシパルとして指定します。Amazon EC2、Amazon ECS、Amazon EKS、Lambda などのコンピューティングリソースを提供する AWS のサービスは、一時的な認証情報を提供し、これらの認証情報を自動的に更新します。これにより、常に有効な認証情報セットを確保できます。これらのサービスでは、一時的な認証情報を取得するために現在のロールを再度引き受ける必要はありません。ただし、[セッションタグ](#)または[セッションポリシー](#)を渡す場合は、現在のロールを再度引き受ける必要があります。ロールの信頼ポリシーを変更してプリンシパルロールの ARN または AWS アカウント ARN を追加する方法については、[ロールの信頼ポリシーの変更 \(コンソール\)](#) を参照してください。
- AWS Management Console を使用してロールを引き受ける場合は、必ずロールの正確な名前を使用してください。ロール名では、ロールを引き受けるときに、大文字と小文字が区別されます。
- AWS STS API または AWS CLI を使用してロールを引き受ける場合は、必ず ARN のロールの正確な名前を使用してください。ロール名では、ロールを引き受けるときに、大文字と小文字が区別されます。
- IAM ポリシーによって、引き受けるロールの `sts:AssumeRole` を呼び出すアクセス許可が付与されていることを確認します。IAM ポリシーの Action 要素で、AssumeRole アクションの呼び出しを許可する必要があります。さらに、IAM ポリシーの Resource 要素で、引き受けるロールを指定する必要があります。たとえば、Resource エlement では Amazon リソースネーム (ARN) またはワイルドカード (\*) で、ロールを指定できます。たとえば、ユーザーに該当する 1 つ以上のポリシーで、以下のようなアクセス許可を付与する必要があります。

```
"Effect": "Allow",  
"Action": "sts:AssumeRole",  
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- IAM アイデンティティが、IAM ポリシーで義務付けられている任意のタグでタグ付けされていることを確認します。たとえば、次のポリシーのアクセス許可では、Condition 要素で、ロールを引き受けることをリクエストするプリンシパルとして、特定のタグを持っていることを義務付けています。department = HR または department = CS でタグ付けされている必要があります。それ以外の場合は、ロールを引き受けることはできません。IAM ユーザーおよびロールのタグ付けについては、「[the section called “IAM リソースのタグ付け”](#)」を参照してください。

```
"Effect": "Allow",
```

```

"Action": "sts:AssumeRole",
"Resource": "*",
"Condition": {"StringEquals": {"aws:PrincipalTag/department": [
    "HR",
    "CS"
  ]}}

```

- ロールの信頼ポリシーで指定されているすべての条件が満たされていることを確認します。1つの Condition で、失効日、外部 ID、またはリクエスト発行元の IP アドレスを定義することができます。次の例では、現在の日付が指定日より後の日付である場合、ポリシーが一致しないため、ルールを引き受けるアクセス権限をユーザーに付与できません。

```

"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
"Condition": {
  "DateLessThan" : {
    "aws:CurrentTime" : "2016-05-01T12:00:00Z"
  }
}

```

- AssumeRole の呼び出し元である AWS アカウントが、引き受けようとしているルールにとって信頼されたエンティティであることを確認します。信頼されたエンティティは、ルールの信頼ポリシーで Principal として定義されます。次の例は、引き受けるルールにアタッチされている信頼ポリシーです。この例の場合、サインインに使用した IAM ユーザーのアカウント ID が 123456789012 である必要があります。ルールの信頼ポリシーの Principal 要素にアカウント番号が表示されていない場合、ルールを引き受けることはできません。アクセスポリシーでどのようなアクセス許可が付与されているかは関係ありません。サンプルポリシーでは、2017 年 7 月 1 日 ~ 2017 年 12 月 31 日 (UTC) (この日付を含む) に発生するアクションのアクセス許可のみ付与できます。これらの日付の前後にログインした場合、ポリシーは一致しないため、ルールを引き受けることはできません。

```

"Effect": "Allow",
"Principal": { "AWS": "arn:aws:iam::123456789012:root" },
"Action": "sts:AssumeRole",
"Condition": {
  "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
  "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
}

```

- ソース ID — 管理者は、ソース ID と呼ばれる AWS でアクションを実行している個人またはアプリケーションを識別するカスタム文字列を渡すように ID を要求するようにロールを構成できます。引き受けるロールがソース ID を設定する必要があるかどうかを確認します。ソース ID の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

## AWS アカウントに新しいロールが表示される

一部の AWS のサービスでは、サービスに直接リンクされた一意のタイプのサービスロールを使用する必要があります。この[サービスにリンクされたロール](#)はサービスによって事前に定義され、サービスで必要なすべてのアクセス許可が含まれます。これにより、必要なアクセス許可を手動で追加する必要がなくなるため、サービスの設定が簡単になります。サービスにリンクされたロールの一般情報については、「[サービスリンクロールの使用](#)」を参照してください。

サービスにリンクされたロールのサポートを開始するときに、既にサービスを使用している可能性があります。その場合、アカウントに新しいロールについて伝える E メールが届くことがあります。このロールには、サービスがお客様に代わってアクションを実行するために必要なすべてのアクセス権限が含まれています。このロールをサポートするために、お客様が実行する必要があるアクションはありません。ただし、アカウントからロールを削除しないでください。ロールを削除すると、サービスが AWS リソースにアクセスするために必要なアクセス許可が削除される可能性があります。アカウントのサービスにリンクされたロールを表示するには、IAM コンソールの IAM ロールページに移動します。サービスにリンクされたロールが、テーブルの [Trusted entities] (信頼されたエンティティ) 列の [(Service-linked role)] ((サービスにリンクされたロール)) に表示されます。

サービスにリンクされたロールをサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照してください。これらのサービスでは、[Service-Linked Role] (サービスにリンクされたロール) 列が、[Yes] (はい) になっています。サービスにリンクされたロールをサービスで使用するには、[Yes (はい)] リンクを選択します。

## 自分の AWS アカウント でロールを編集または削除できない

IAM の「[サービスにリンクされたロール](#)」のアクセス許可を削除または編集することはできません。これらのロールには、サービスがお客様に代わってアクションを実行するために必要な事前定義された信頼とアクセス許可が含まれます。サービスにリンクされたロールの説明は、IAM コンソール、AWS CLI、API のいずれかでのみ編集できます。アカウントのサービスにリンクされたロールを表示するには、コンソールの IAM ロール ページに移動します。サービスにリンクされたロールが、テーブルの [Trusted entities] (信頼されたエンティティ) 列の [(Service-linked role)] ((サービスにリンクされたロール)) に表示されます。ロールの [Summary (概要)] ページのバナーにも、そのロールが



サービスにリンクされたロールであることが示されています。サービスでアクションがサポートされている場合、リンクされたサービスを通じてのみこれらのロールを管理および削除できます。サービスにリンクされたロールを変更または削除すると、サービスが AWS リソースにアクセスするために必要なアクセス許可が削除される可能性があるため、注意してください。

サービスにリンクされたロールをサポートするサービスについては、「[IAM と連携する AWS のサービス](#)」を参照してください。これらのサービスでは、[Service-Linked Role] (サービスにリンクされたロール) 列が、[Yes] (はい) になっています。

## 次のことを実行する権限がない: iam:PassRole

リンクされたサービスロールを作成する場合、サービスにそのロールを渡す権限を持っている必要があります。一部のサービスでは、そのサービスでアクションを実行する際にアカウント内にサービスにリンクされたロールが自動的に作成されます。たとえば Amazon EC2 Auto Scaling では、Auto Scaling グループを初めて作成すると、サービスにリンクされたロール `AWSServiceRoleForAutoScaling` が作成されます。PassRole アクセス許可がない状態で Auto Scaling グループを作成しようとすると、以下のようなエラーが表示されます。

```
ClientError: An error occurred (AccessDenied) when calling the PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/Testrole/Diego is not authorized to perform: iam:PassRole on resource: arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling
```

このエラーを解決するには、自分に `iam:PassRole` アクセス許可を追加するよう管理者に依頼します。

サービスにリンクされたロールをサポートするサービスを確認するには、「[IAM と連携する AWS のサービス](#)」を参照してください。サービスにリンクされたロールをサービスで作成できるかどうかを確認するには、[はい] リンクを選択して、該当サービスのサービスにリンクされたロールに関するドキュメントを確認します。

## 12 時間のセッションに使用するロールを引き受けることができない (AWS CLI、AWS API)

AWS STS AssumeRole\* API または `assume-role*` CLI オペレーションを使用してロールを引き受ける場合は、`DurationSeconds` パラメータの値を指定できます。900 秒 (15 分) からロールの最大セッション期間設定までの値を指定できます。この設定よりも高い値を指定した場合、オペレーションは失敗します。この設定の最大値は 12 時間です。たとえば、12 時間のセッションの期間を

指定したが、管理者が最大のセッション期間を 6 時間に設定した場合、オペレーションは失敗します。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。

[ロールの連鎖](#) (ロールを使用して 2 つ目のロールを引き受ける) を使用している場合、セッションは最大 1 時間に制限されます。この場合 DurationSeconds パラメータを使用して 1 時間より大きい値を指定すると、オペレーションは失敗します。

## IAM コンソールでロールを切り替えようとするとエラーが発生する

[ロールの切り替え] ページに入力する情報は、ロールの情報と一致している必要があります。そうでない場合は、オペレーションは失敗し、次のエラーが表示されます。

```
Invalid information in one or more fields. Check your information or contact your administrator.
```

このエラーが表示された場合は、次の情報が正しいことを確認します。

- アカウント ID またはエイリアス - AWS アカウント ID は 12 桁の数字です。アカウントには、AWS アカウント ID の代わりに使用できる会社名などのわかりやすい識別子であるエイリアスがある場合があります。このフィールドでは、アカウント ID またはエイリアスのいずれかを使用できます。
- ロール名 - ロール名では、大文字と小文字が区別されます。アカウント ID とロール名は、ロールに対して設定されているものと一致する必要があります。

引き続きエラーメッセージが表示される場合は、管理者に問い合わせ、以前の情報を確認してください。ロールの信頼ポリシーまたは IAM ユーザーポリシーによって、アクセスが制限される場合があります。管理者は、これらのポリシーのアクセス許可を確認できます。

## ロールには、アクションの実行を許可するポリシーがあるが、「アクセスが拒否されました」というメッセージが表示される

ロールセッションはセッションポリシーによって制限される場合があります。AWS STS を使用してプログラムで [一時的なセキュリティ認証情報をリクエストする](#) 場合は、オプションでインラインまたは管理 [セッションポリシー](#) を渡すことができます。セッションポリシーは、ロールの一時認証情報セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。Policy パラメータを使用して単一の JSON インラインセッションポリシードキュメントを渡すことができます。PolicyArns パラメータを使用して、最大 10 個の管理セッションポリシーを指定できます。



結果として得られるセッションのアクセス許可は、ロールの ID ベースのポリシーとセッションポリシーの共通部分です。または、管理者またはカスタムプログラムから一時的な認証情報が提供されている場合は、アクセスを制限するためのセッションポリシーが含まれている可能性があります。

## サービスがロールのデフォルトのポリシーバージョンを作成しなかった

サービスロールは、サービスがお客様に代わってお客様のアカウントでアクションを実行するために引き受けるロールです。AWS のサービス環境には、セットアップ時にサービスが引き受けるロールを定義する必要があるものもあります。場合によっては、サービスによってサービスロールとそのポリシーが IAM に作成されます。IAM 内でサービスロールとそのポリシーを変更または削除することはできますが、AWS では推奨しません。ロールとポリシーは、そのサービスでのみ使用することを目的としています。ポリシーを編集して別の環境を設定した場合、サービスが同じロールとポリシーを使用しようとする、オペレーションが失敗する可能性があります。

たとえば、AWS CodeBuild を初めて使用する場合、サービスは `codebuild-RWBCore-service-role` という名前のロールを作成します。このサービスロールは、`codebuild-RWBCore-managed-policy` という名前のポリシーを使用します。ポリシーを編集すると、新しいバージョンが作成され、そのバージョンがデフォルトバージョンとして保存されます。AWS CodeBuild で後続のオペレーションを実行すると、サービスがポリシーの更新を試みることがあります。その場合は、次のエラーが表示されます。

```
codebuild.amazon.com did not create the default version (V2) of the codebuild-RWBCore-managed-policy policy that is attached to the codebuild-RWBCore-service-role role. To continue, detach the policy from any other identities and then delete the policy and the role.
```

このエラーが表示された場合は、サービスオペレーションを続行する前に IAM で変更を行う必要があります。まず、デフォルトポリシーバージョンを V1 に設定し、オペレーションを再試行します。V1 が以前に削除された場合、または V1 を選択しても機能しない場合は、既存のポリシーとロールをクリーンアップして削除します。

管理ポリシーの編集の詳細については、「[カスタマー管理ポリシーの編集 \(コンソール\)](#)」を参照してください。ポリシーのバージョンの詳細については、「[IAM ポリシーのバージョンング](#)」を参照してください。

サービスロールとそのポリシーを削除するには

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

2. ナビゲーションペインで、[ポリシー] を選択します。
3. ポリシーの一覧で、削除するポリシーの名前を選択します。
4. [アタッチされたエンティティ] タブを選択して、このポリシーを使用する IAM ユーザー、グループ、またはロールを表示します。これらの ID のいずれかがポリシーを使用する場合は、次のタスクを実行します。
  - a. 必要なアクセス許可を持つ新しい管理ポリシーを作成します。アクションの前後に ID が同じアクセス許可を持つようにするには、既存のポリシーから JSON ポリシードキュメントをコピーします。次に、新しい管理ポリシーを作成し、[JSON エディタを使用したポリシーの作成](#)の説明に従って JSON ドキュメントを貼り付けます。
  - b. 影響を受ける ID ごとに、新しいポリシーをアタッチしてから、古いポリシーをデタッチします。詳細については、「[IAM ID のアクセス許可の追加および削除](#)」を参照してください。
5. ナビゲーションペインで Roles (ロール) を選択します。
6. ロールのリストで、削除するロールの名前を選択します。
7. [信頼関係] タブを選択して、ロールを引き受けることができるエンティティを表示します。サービス以外のエンティティが一覧表示されている場合は、次のタスクを実行します。
  - a. これらのエンティティを信頼する[新しいロールを作成します](#)。
  - b. 前のステップで作成したポリシー。このステップを省略した場合は、新しい管理ポリシーをここで作成します。
  - c. ロールを引き受けていたすべてのユーザーに、もう実行できないことを通知します。新しいロールを引き受ける方法と、同じアクセス許可を持つ方法に関する情報を提供します。
8. [ポリシーを削除します](#)。
9. [ロールを削除します](#)。

## コンソールにサービスロールのユースケースがない

一部のサービスでは、お客様に代わってアクションを実行するサービスアクセス許可を付与するために、サービスロールを手動で作成する必要があります。サービスが IAM コンソールに表示されない場合は、信頼されたプリンシパルとしてサービスを手動で一覧表示する必要があります。使用しているサービスまたは機能のドキュメントに、信頼されたプリンシパルとしてサービスを一覧表示する手順が含まれていない場合は、ページについてのフィードバックを提供します。

サービスロールを手動で作成するには、そのロールを引き受けるサービスの[サービスプリンシパル](#)を知っている必要があります。サービスプリンシパルは、サービスにアクセス許可を付与するために使用される識別子です。サービスプリンシパルはサービスによって定義されます。

次の項目を確認することで、一部のサービスのサービスプリンシパルを検索できます。

1. [IAM と連携する AWS のサービス](#) を開きます。
2. サービスの [Service-linked roles] (サービスにリンクされたロール) 列に [Yes] (はい) が表示されているかどうかを確認します。
3. サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] リンクを選択します。
4. [サービスプリンシパル](#)を表示するには、そのサービスの [Service-Linked Role Permissions] (サービスにリンクされたロールのアクセス許可) のセクションを探します。

[AWS CLI コマンド](#)または [AWS API オペレーション](#)を使用して、サービスロールを手動で作成できます。IAM コンソールを使用してサービスロールを手動で作成するには、次のタスクを実行します。

1. アカウント ID を使用して IAM ロールを作成します。ポリシーをアタッチしたり、アクセス許可を付与したりしないでください。詳細については、「[IAM ユーザーにアクセス許可を委任するロールの作成](#)」を参照してください。
2. ロールを開き、信頼関係を編集します。ロールはアカウントを信頼する代わりに、サービスを信頼する必要があります。例えば、次の Principal 要素を更新します。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

プリンシパルをサービスの値 (IAM など) に変更します。

```
"Principal": { "Service": "iam.amazonaws.com" }
```

3. アクセス許可ポリシーをロールにアタッチして、サービスが必要とするアクセス許可を追加します。
4. アクセス許可を必要とするサービスに戻り、文書化されたメソッドを使用して、新しいサービスロールについてサービスに通知します。

## IAM および Amazon EC2 のトラブルシューティング

この情報を使用して、アクセス拒否された問題や、Amazon EC2 および IAM を操作する際に発生する可能性のある他の問題のトラブルシューティングや修復を行います。

### トピック

- [インスタンスの起動時に、Amazon EC2 コンソールの \[IAM ロール\] リストにあるべきロールが見当たらない](#)
- [インスタンスの認証情報が間違っただロールのものになっている](#)
- [AddRoleToInstanceProfile を呼び出そうとすると、AccessDenied エラーが発生する。](#)
- [Amazon EC2: ロールを使用してインスタンスを起動しようとする、AccessDenied エラーが発生する](#)
- [EC2 インスタンスにある一時的なセキュリティ認証情報にアクセスできない](#)
- [IAM サブツリーの info ドキュメントのエラーは何を意味しますか？](#)

### インスタンスの起動時に、Amazon EC2 コンソールの [IAM ロール] リストにあるべきロールが見当たらない

以下をチェックしてください。

- IAM ユーザーとしてサインインしている場合、ListInstanceProfiles を呼び出す権限があることを確認します。ロールを使用した作業に必要なアクセス許可の詳細については、[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#) の「Amazon EC2 でのロールの使用に必要なアクセス許可」を参照してください。ユーザーに権限を付与する方法の詳細については、「[IAM ポリシーを管理する](#)」を参照してください。

自身のアクセス許可を修正できない場合は、IAM を操作できる管理者へ問い合わせる必要があります。

- IAM CLI または API を使用してロールを作成した場合は、インスタンスプロファイルを作成して、そのインスタンスプロファイルにロールを追加していることを確認します。また、ロールとインスタンスプロファイルに異なる名前を付けた場合は、Amazon EC2 コンソールの IAM ロールのリストに正しいロール名が表示されません。Amazon EC2 コンソールの [IAM ロール] リストには、ロール名ではなくインスタンスプロファイル名が表示されます。使用するロールが含まれるインスタンスプロファイルの名前を選択する必要があります。インスタンスプロファイルの詳細については、[インスタンスプロファイルの使用](#) を参照してください。

**Note**

IAM コンソールを使用してロールを作成する場合は、インスタンスプロファイルを使用する必要はありません。IAM コンソールで作成する各ロールでは、ロールと同じ名前のインスタンスプロファイルが作成され、ロールは自動的にそのインスタンスプロファイルに追加されます。インスタンスプロファイルに含めることができる IAM ロールは 1 つのみであり、緩和できません。

## インスタンスの認証情報が間違っ たロールのものになっている

インスタンスプロファイルのロールが最近置き換えられている可能性があります。その場合は、アプリケーションは自動的に予定された次回の認証情報の更新を終えてからでなければ、ロールの認証情報にアクセスすることはできません。

変更を強制的に実行するには、[インスタンスプロファイル](#)の関連付けを解除してから、[インスタンスプロファイルを関連付ける](#)か、インスタンスを停止してから再起動します。

## **AddRoleToInstanceProfile** を呼び出そうとすると、**AccessDenied** エラーが発生する。

IAM ユーザーとしてリクエストを発行する場合は、以下の条件が満たされていることを確認します。

- iam:AddRoleToInstanceProfile でインスタンスプロファイルの ARN (arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile など) に一致するリソースが定義されている。

ロールを使用した作業に必要なアクセス許可の詳細については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」の中 ユーザーに権限を付与する方法の詳細については、「[IAM ポリシーを管理する](#)」を参照してください。

## Amazon EC2: ロールを使用してインスタンスを起動しようとする と、**AccessDenied** エラーが発生する

以下をチェックしてください。

- インスタンスプロファイルを使用せずにインスタンスを起動します。この作業は、問題が Amazon EC2 インスタンスの IAM ロールに限定していることを確認するものです。
- IAM ユーザーとしてリクエストを発行する場合は、以下の条件が満たされていることを確認します。
  - ec2:RunInstances でリソースがワイルドカード (\*) で定義されている
  - iam:PassRole でロールの ARN (arn:aws:iam::999999999999:role/ExampleRoleName など) に一致するリソースが定義されている
- IAM GetInstanceProfile アクションを呼び出し、有効なインスタンスプロファイル名またはインスタンスプロファイル ARN を使用していることを確認します。詳細については、「[Amazon EC2 インスタンスで IAM ロールを使用する](#)」を参照してください。
- IAM GetInstanceProfile アクションを呼び出し、インスタンスプロファイルにロールがあることを確認します。空のインスタンスプロファイルは、AccessDenied エラーとなります。ロールの作成に関する詳細については、「[IAM ロールの作成](#)」を参照してください。

ロールを使用した作業に必要なアクセス許可の詳細については、「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」の中 ユーザーに権限を付与する方法の詳細については、「[IAM ポリシーを管理する](#)」を参照してください。

## EC2 インスタンスにある一時的なセキュリティ認証情報にアクセスできない

EC2 インスタンスの一時的なセキュリティ認証情報にアクセスするには、まず IAM コンソールを使用してロールを作成する必要があります。次に、そのロールを使用する EC2 インスタンスを起動し、実行中のインスタンスを確認します。詳細については、[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#) の「開始方法」を参照してください。

引き続き EC2 インスタンスで一時的なセキュリティ認証情報にアクセスできない場合は、以下を確認してください。

- インスタンスメタデータサービス (IMDS) の他の部分にアクセスできますか? アクセスできない場合、IMDS への要求アクセスを遮断するファイアウォールのルールがないことを確認します。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/  
hostname; echo
```



- IMDS の iam サブツリーは存在していますか？ 存在していない場合、EC2 DescribeInstances API オペレーションを呼び出すか、aws ec2 describe-instances CLI コマンドを使用して、インスタンスに関連する IAM インスタンスプロファイルがあることを確認します。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam; echo
```

- IAM サブツリーの info ドキュメントにエラーがあるか確認します。エラーがある場合は、[IAM サブツリーの info ドキュメントのエラーは何を意味しますか？](#) をご覧ください。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/info; echo
```

## IAM サブツリーの info ドキュメントのエラーは何を意味しますか？

### iam/info ドキュメントで "Code": "InstanceProfileNotFound" が表示される

IAM インスタンスプロファイルが削除され、Amazon EC2 がインスタンスに認証情報を発行できません。有効なインスタンスプロファイルを Amazon EC2 インスタンスにアタッチする必要があります。

その名前が付いたインスタンスプロファイルが存在する場合、そのインスタンスプロファイルが削除されておらず、他のプロファイルが同じ名前で作成されていることを確認します。

1. IAM の GetInstanceProfile オペレーションを呼び出し、InstanceProfileId を取得します。
2. DescribeInstances の Amazon EC2 オペレーションを呼び出し、インスタンスの IamInstanceProfileId を取得します。
3. IAM オペレーションで取得した InstanceProfileId が Amazon EC2 オペレーションで取得した IamInstanceProfileId と一致することを確認します。

ID が異なる場合、インスタンスに付属したインスタンスプロファイルはすでに無効になっています。有効なインスタンスプロファイルをインスタンスにアタッチする必要があります。

## iam/info ドキュメントには完了と表示されるが、**"Message": "Instance Profile does not contain a role..."** が表示される

IAM の `RemoveRoleFromInstanceProfile` アクションによりロールがインスタンスプロファイルから削除されています。IAM の `AddRoleToInstanceProfile` アクションを使用すると、インスタンスプロファイルにロールをアタッチすることができます。アプリケーションは、次回に予定された更新を終えてからでなければ、ロールの認証情報にアクセスすることはできません。

変更を強制的に実行するには、[インスタンスプロファイル](#)の関連付けを解除してから、[インスタンスプロファイルを関連付ける](#)か、インスタンスを停止してから再起動します。

## iam/security-credentials/[role-name] ドキュメントで **"Code": "AssumeRoleUnauthorizedAccess"** が表示される

Amazon EC2 にロールを引き受けるアクセス許可がありません。以下の例のように、ロールを引き受けるアクセス許可は、そのロールにアタッチされた信頼ポリシーで管理されます。IAM `UpdateAssumeRolePolicy` API を使用して、信頼ポリシーを更新します。

```
{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": ["ec2.amazonaws.com"]}, "Action": ["sts:AssumeRole"]}]}
```

アプリケーションは、自動的に予定された次回の更新を終えてからでなければ、ロールの認証情報にアクセスすることはできません。

変更を強制的に実行するには、[インスタンスプロファイル](#)の関連付けを解除してから、[インスタンスプロファイルを関連付ける](#)か、インスタンスを停止してから再起動します。

## Amazon S3 および IAM のトラブルシューティング

この情報を使用して、Amazon S3 および IAM を操作する際に発生する可能性がある問題のトラブルシューティングや修復を行います。

### Amazon S3 バケットへの匿名アクセスを付与する方法

`principal` 要素でワイルドカード (\*) を指定する Amazon S3 バケットポリシーを使用すると、だれでもバケットにアクセスできるようになります。匿名アクセスでは、誰でも (AWS アカウントのないユーザーでも) バケットにアクセスできます。Amazon S3 バケットポリシーの例について



は、Amazon Simple Storage Service ユーザーガイドの「[バケットポリシーの例](#)」を参照してください。

AWS アカウント のルートユーザーとしてサインインしているが、Amazon S3 バケットにアクセスできない。

場合によっては、IAM と Amazon S3 へのフルアクセス許可を持つ IAM ユーザーがいる場合があります。IAM ユーザーが Amazon S3 バケットにバケットポリシーを割り当て、AWS アカウントのルートユーザー をプリンシパルとして指定しない場合、ルートユーザーはそのバケットへのアクセスを拒否されます。ただし、ルートユーザーとして、バケットへアクセスすることは可能です。そのためには、Amazon S3 コンソールまたは AWS CLI から アクセスを許可するようにバケットポリシーを変更します。次のプリンシパルを使用します。**123456789012** を AWS アカウント の ID に置き換えます。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

## AWS での SAML 2.0 フェデレーションのトラブルシューティング

この情報を使用して、IAM で SAML 2.0 とフェデレーションを操作するときに発生する可能性がある問題を診断して修復します。

### トピック

- [エラー: Your request included an invalid SAML response. To logout, click here.](#)
- [エラー: RoleSessionName is required in AuthnResponse \(service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken\)](#)
- [エラー: Not authorized to perform sts:AssumeRoleWithSAML \(service: AWSSecurityTokenService; status code: 403; error code: AccessDenied\)](#)
- [エラー: RoleSessionName in AuthnResponse must match \[a-zA-Z\\_0-9+=,.@-\]{2,64} \(service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken\)](#)
- [Error: Source Identity must match \[a-zA-Z\\_0-9+=,.@-\]{2,64} and not begin with "aws:" \(service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken\)](#)
- [エラー: Response signature invalid \(service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken\)](#)
- [エラー: Failed to assume role: Issuer not present in specified provider \(service: AWSOpenIdDiscoveryService; status code: 400; error code: AuthSamlInvalidSamlResponseException\)](#)

- [エラー: Could not parse metadata.](#)
- [エラー: Specified provider doesn't exist.](#)
- [エラー: Requested DurationSeconds exceeds MaxSessionDuration set for this role.](#)
- [エラー: Response does not contain the required audience.](#)
- [トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)

エラー: Your request included an invalid SAML response. To logout, click [here](#).

このエラーは、ID プロバイダーからの SAML レスポンスに、Name が `https://aws.amazon.com/SAML/Attributes/Role` に設定された属性が含まれない場合に発生することがあります。属性には、それぞれにカンマ区切りの文字列のペアを持つ、1 つ以上の AttributeValue エlement が含まれている必要があります。

- ユーザーをマッピングできるロールの ARN
- SAML プロバイダーの ARN

詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。ブラウザで SAML レスポンスを表示するには、「[トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)」のステップに従います。

エラー: RoleSessionName is required in AuthnResponse (service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken)

このエラーは、ID プロバイダーからの SAML レスポンスに、Name が `https://aws.amazon.com/SAML/Attributes/RoleSessionName` に設定された属性が含まれない場合に発生することがあります。属性値は、ユーザーの ID で、通常は ID または E メールアドレスです。

詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。ブラウザで SAML レスポンスを表示するには、「[トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)」のステップに従います。

## エラー: Not authorized to perform sts:AssumeRoleWithSAML (service: AWSSecurityTokenService; status code: 403; error code: AccessDenied)

このエラーは、SAML レスポンスで指定された IAM ロールのスペルが間違っているか存在しない場合に発生することがあります。ロール名は大文字と小文字が区別されるため、ロールの正確な名前を使用してください。SAML サービスプロバイダー設定のロール名を修正します。

ロール信頼ポリシーに sts:AssumeRoleWithSAML アクションが含まれる場合にのみ、アクセスが許可されます。SAML アサーションが [PrincipalTag](#) 属性を使用するように設定されている場合、信頼ポリシーにも sts:TagSession アクションを含める必要があります。セッションタグの詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。

このエラーは、ロール信頼ポリシーに sts:SetSourceIdentity アクセス許可がない場合に発生する可能性があります。SAML アサーションが [SourceIdentity](#) 属性を使用するように設定されている場合、信頼ポリシーにも sts:SetSourceIdentity アクションを含める必要があります。ソース ID の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

このエラーは、フェデレーションユーザーにロールを引き受けるアクセス権限がない場合に発生することがあります。ロールには、IAM SAML ID プロバイダーの ARN を Principal として指定する信頼ポリシーが必要です。ロールには、ロールを引き受けることができるユーザーを管理する条件も含まれています。ユーザーが条件を満たすことを確認します。

このエラーは、SAML レスポンスに Subject を含む NameID がない場合に発生することがあります。

詳細については、「[フェデレーティッドユーザーのために AWS でアクセス許可を確立する](#)」および「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。ブラウザで SAML レスポンスを表示するには、「[トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)」のステップに従います。

## エラー: RoleSessionName in AuthnResponse must match [a-zA-Z\_0-9+=,.,@-]{2,64} (service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken)

このエラーは、RoleSessionName 属性値が長すぎるか、無効な文字が含まれる場合に発生することがあります。有効な最大長は 64 文字です。

詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。ブラウザで SAML レスポンスを表示するには、「[トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)」のステップに従います。

**Error: Source Identity must match [a-zA-Z\_0-9+=,.\@-]{2,64} and not begin with "aws:" (service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken)**

このエラーは、sourceIdentity 属性値が長すぎるか、無効な文字が含まれる場合に発生することがあります。有効な最大長は 64 文字です。ソース ID の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

SAML アサーション作成の詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。ブラウザで SAML レスポンスを表示するには、「[トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法](#)」のステップに従います。

**エラー: Response signature invalid (service: AWSSecurityTokenService; status code: 400; error code: InvalidIdentityToken)**

このエラーは、ID プロバイダーのフェデレーションメタデータが、IAM ID プロバイダーのメタデータに一致しない場合に発生することがあります。たとえば、ID サービスプロバイダーのメタデータファイルが、失効した証明書を更新するために変更される場合があります。更新された SAML メタデータファイルを ID サービスプロバイダーからダウンロードします。次に、IAM で定義する AWS ID プロバイダーエンティティで、aws iam update-saml-provider クロスプラットフォーム CLI コマンドまたは Update-IAMSAMLProvider PowerShell コマンドレットを使ってこれを更新します。

**エラー: Failed to assume role: Issuer not present in specified provider (service: AWSOpenIdDiscoveryService; status code: 400; error code: AuthSamlInvalidSamlResponseException)**

このエラーは、SAML レスポンスの発行元がフェデレーションメタデータファイルで宣言されている発行者と一致しない場合に発生することがあります。メタデータファイルは、ID プロバイダーを IAM で作成したときに AWS にアップロードされました。

**エラー: Could not parse metadata.**

このエラーはメタデータファイルを適切にフォーマットしてない場合に発生することがあります。

AWS Management Console で、[SAML ID プロバイダーを作成または管理](#)する場合、ID プロバイダーから SAML メタデータドキュメントを取得する必要があります。

このメタデータファイルには発行者の名前、失効情報、およびキーが含まれており、これらを使用して、IdP から受け取った SAML 認証レスポンス (アサーション) を検証できます。メタデータファイルはバイトオーダーマーク (BOM) なしで UTF-8 形式でエンコードする必要があります。BOM を削除するには、Notepad++ などのテキスト編集ツールを使用して UTF-8 としてファイルをエンコードできます。

SAML メタデータドキュメントの一部として含まれている x.509 証明書では、少なくとも 1024 ビットのキーサイズを使用する必要があります。また、x.509 証明書には、拡張領域が繰り返されていないことが必要です。拡張領域は使用できますが、証明書に 1 回しか出現できません。x.509 証明書がいずれかの条件を満たしていない場合、IdP の作成は失敗し、「メタデータを解析できない」エラーが返されます。

[SAML V2.0 メタデータ相互運用性プロファイルバージョン 1.0](#) で定義されているように、IAM はメタデータドキュメントの X.509 証明書の有効期限を評価したりアクションを実行したりすることはできません。

## エラー: Specified provider doesn't exist.

このエラーは、SAML アサーションで指定したプロバイダーの名前が IAM で設定されたプロバイダーの名前と一致しない場合に発生することがあります。プロバイダー名の表示の詳細については、「[IAM で SAML ID プロバイダーを作成する](#)」を参照してください。

## エラー: Requested DurationSeconds exceeds MaxSessionDuration set for this role.

このエラーは、AWS CLI または API からロールを引き受ける場合に発生することがあります。

[assume-role-with-saml](#) CLI または [AssumeRoleWithSAML](#) API オペレーションを使用してロールを引き受ける場合は、DurationSeconds パラメータの値を指定できます。900 秒 (15 分) からロールの最大セッション期間設定までの値を指定できます。この設定よりも高い値を指定した場合、オペレーションは失敗します。たとえば、12 時間のセッションの期間を指定したが、管理者が最大のセッション期間を 6 時間に設定した場合、オペレーションは失敗します。ロールの最大値を確認する方法については、「[ロールの最大セッション期間設定の表示](#)」を参照してください。

## エラー: Response does not contain the required audience.

このエラーは、SAML 設定のオーディエンス URL と ID プロバイダーが一致しない場合に発生する可能性があります。ID プロバイダー (IdP) 依存パーティ識別子が SAML 設定で提供されたオーディエンス URL (エンティティ ID) と完全に一致することを確認してください。

## トラブルシューティングのためにブラウザで SAML レスポンスを表示する方法

次の手順では、SAML 2.0 に関連する問題のトラブルシューティング時に、ブラウザからサービスプロバイダーからの SAML 応答を表示する方法について説明します。

すべてのブラウザに共通して、問題を再現できるページに移動します。次に、ブラウザごとの適切なステップに従います。

### トピック

- [Google Chrome](#)
- [Mozilla Firefox](#)
- [Apple Safari](#)
- [Base64 エンコードされた SAML レスポンスの処理](#)

## Google Chrome

Chrome で SAML レスポンスを表示するには

これらのステップは、Google Chrome のバージョン 106.0.5249.103 (公式ビルド) (arm64) を使用してテストされました。別のバージョンを使用している場合、それに応じてステップの調整が必要になることがあります。

1. F12 キーを押して、[Developer Tools] (デベロッパーツール) コンソールを開始します。
2. [Network] (ネットワーク) タブを選択し、[Developer Tools] (デベロッパーツール) ウィンドウの左上にある [Preserve log] (ログを保存) を選択します。
3. 問題を再現します。
4. (オプション) [Developer Tools] (デベロッパーツール) の [Network] (ネットワーク) ログペインに [Method] (メソッド) 列が表示されない場合は、任意の列ラベルを右クリックし、[Method] (メソッド) を選択して列を追加します。



5. [Developer Tools] (デベロッパーツール) の [Network] (ネットワーク) ログペインで [SAML Post] (SAML 投稿) を探します。その行を選択し、上部にある [Payload] (ペイロード) タブを表示します。エンコードされたリクエストを含む [SAMLResponse] 要素を探します。関連する値は、Base64 でエンコードされたレスポンスです。

## Mozilla Firefox

Firefox で SAML レスポンスを表示するには

この手順は、Mozilla Firefox のバージョン 105.0.3 (64 ビット) でテストされています。別のバージョンを使用している場合、それに応じてステップの調整が必要になることがあります。

1. F12 を押して、[Web Developer Tools] (ウェブデベロッパーツール) コンソールを起動します。
2. [Network (ネットワーク)] タブを選択します。
3. [Web Developer Tools] (ウェブデベロッパーツール) ウィンドウの右上で、オプション (小さな歯車のアイコン) を選択します。[Persist logs] (永続ログ) を選択します。
4. 問題を再現します。
5. (オプション) [Web Developer Tools] (ウェブデベロッパーツール) の [Network] (ネットワーク) ログペインに [Method] (メソッド) 列が表示されない場合は、任意の列ラベルを右クリックし、[Method] (メソッド) を選択して列を追加します。
6. テーブルで [POST SAML] を見つけます。その行を選択し、[Request] (リクエスト) タブを表示して SAMLResponse 要素を見つけてください。関連する値は、Base64 でエンコードされたレスポンスです。

## Apple Safari

Safari で SAML レスポンスを表示するには

これらのステップは、Apple Safari のバージョン 16.0 (17614.1.25.9.10, 17614) を使用してテストされました。別のバージョンを使用している場合、それに応じてステップの調整が必要になることがあります。

1. Safari で Web Inspector を有効にします。[設定] ウィンドウを開き、[Advanced (詳細)] タブを選択して、[Show Develop menu in the menu bar (メニューバーに開発メニューを表示)] を選択します。
2. ここで Web Inspector を開くことができます。メニューバーで [Develop] (開発) を選択し、[Show Web Inspector] (Web インスペクターを表示) を選択します。

3. [Network (ネットワーク)] タブを選択します。
4. [Web Inspector] (Web インスペクター) ウィンドウの左上で、オプション (3本の水平線を含む小さな円のアイコン) を選択します。[Preserve Log] (ログを保存) を選択します。
5. (オプション) [Web Inspector] (Web インスペクター) の [Network] (ネットワーク) ログペインに [Method] (メソッド) 列が表示されない場合は、任意の列ラベルを右クリックし、[Method] (メソッド) を選択して列を追加します。
6. 問題を再現します。
7. テーブルで [POST SAML] を見つけます。その行を選択し、[Headers] (ヘッダー) タブを表示します。
8. エンコードされたリクエストを含む [SAMLResponse] 要素を探します。下にスクロールして、Request Data という名前の SAMLResponse を探します。関連する値は、Base64 でエンコードされたレスポンスです。

## Base64 エンコードされた SAML レスポンスの処理

ブラウザで Base64 エンコードされた SAML レスポンスエレメントが見つかったら、それをコピーし、任意の Base64 デコードツールを使用して、XML タグの付いたレスポンスを抽出します。

### セキュリティのヒント

表示している SAML レスポンスデータには重要なデータが含まれる場合があるため、オンラインの base64 デコーダを使用しないことをお勧めします。代わりに、ローカルコンピュータにインストールされた、ネットワーク経由で SAML データを送信しないツールを使用します。

Windows システムの組み込みオプション (PowerShell):

```
PS C:\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext"))
```

MacOS および Linux システムの組み込みオプション:

```
$ echo "base64encodedtext" | base64 --decode
```

デコードされたファイルの値を確認する



デコードされた SAML レスポンスファイルの値を確認します。

- saml:NameID 属性の値が認証されたユーザーのユーザー名と一致することを確認します。
- <https://aws.amazon.com/SAML/Attributes/Role> の値を確認します。ARN と SAML プロバイダーでは大文字と小文字が区別され、[ARN](#) はアカウントのリソースと一致する必要があります。
- <https://aws.amazon.com/SAML/Attributes/RoleSessionName> の値を確認します。値は、[クレームルール](#)の値と一致する必要があります。
- E メールアドレスまたはアカウント名に属性値を設定する場合は、値が正しいことを確認してください。値は、認証されたユーザーの E メールアドレスまたはアカウント名に対応する必要があります。

エラーをチェックして設定を確認する

値にエラーが含まれているかどうかを確認し、次の設定が正しいことを確認します。

- クレームルールは必須要素を満たし、すべての ARN が正しい。詳細については、「[証明書利用者の信頼およびクレームの追加によって SAML 2.0 IdP を設定する](#)」を参照してください。
- IdP から最新のメタデータファイルを SAML プロバイダーで AWS にアップロードしました。詳細については、「[SAML 2.0 フェデレーティッドユーザーが AWS Management Consoleにアクセス可能にする](#)」を参照してください。
- IAM ロールの信頼ポリシーを正しく設定しました。詳細については、「[ロールの修正](#)」を参照してください。

# AWS Identity and Access Management のリファレンス情報

このセクションのトピックを使用して、IAM および AWS STS のさまざまな側面に関する詳細なリファレンス情報を検索します。

## トピック

- [Amazon リソースネーム \(ARN\)](#)
- [IAM ID](#)
- [IAM と AWS STSクォータ](#)
- [インターフェイス VPC エンドポイント](#)
- [IAM と連携する AWS のサービス](#)
- [AWS API リクエストの署名](#)
- [IAM JSON ポリシーリファレンス](#)

## Amazon リソースネーム (ARN)

Amazon リソースネーム (ARN) は、AWS リソースを一意に識別します。IAM ポリシー、Amazon Relational Database Service (Amazon RDS) タグ、API コールなど、すべての AWS 全体でリソースを明確に指定する必要がある場合は ARN が必要になります。

## ARN 形式

ARN の一般的な形式を次に示します。具体的な形式は、リソースによって異なります。ARN を使用するには、*italicized* のテキスト (arn 以外のすべて) をリソース固有の情報に置き換えます。一部のリソースの ARN では、リージョン、アカウント ID、または、リージョンとアカウント ID の両方が省略されていることに注意してください。

```
arn:partition:service:region:account-id:resource-id
arn:partition:service:region:account-id:resource-type/resource-id
arn:partition:service:region:account-id:resource-type:resource-id
```

### *partition*

リソースが置かれているパーティション。パーティションは AWS リージョンのグループです。各 AWS アカウントの範囲は 1 つのパーティションです。

サポートされているパーティションは以下のとおりです。

- aws - AWS リージョン
- aws-cn - 中国リージョン
- aws-us-gov - AWS GovCloud (US) リージョン

### *service*

AWS 製品を識別するサービス名前空間。

### *region*

リージョンコード。たとえば、米国東部 (オハイオ) の場合は、us-east-2 を使用します。リージョンコードの一覧については、「AWS 全般のリファレンス」の「[リージョンエンドポイント](#)」を参照してください。

### *account-id*

リソースを所有しておりハイフンがない AWS アカウントの ID。例えば、123456789012 と指定します。

### *resource-type*

リソースタイプ。たとえば、vpc は仮想プライベートクラウド (VPC) 用です。

### *resource-id*

リソース識別子。これはリソースの名前、リソースの ID、または[リソースパス](#)です。一部のリソース識別子には、親リソース (sub-resource-type/parent-resource/sub-resource) またはバージョン (resource-type:resource-name:qualifier) などの修飾子が含まれます。

## 例

### IAM ユーザー

```
arn:aws:iam::123456789012:user/johndoe
```

### SNS トピック

```
arn:aws:sns:us-east-1:123456789012:example-sns-topic-name
```

### VPC

```
arn:aws:ec2:us-east-1:123456789012:vpc/vpc-0e9801d129EXAMPLE
```

## リソースの ARN 形式を検索する

ARN の正確な形式は、サービスとリソースのタイプによって異なります。一部のリソース ARN には、パス、変数、またはワイルドカードを含めることができます。特定の AWS リソースの ARN 形式を検索するには、「[Service Authorization Reference](#)」(サービス認証リファレンス)を開き、サービスのページを開いて、リソースタイプの形式を表示します。

### ARN のパス

リソース ARN にはパスを含めることができます。たとえば Amazon S3 では、リソース ID はスラッシュ (/) を挿入してパスを作成することができるオブジェクト名です。同様に、IAM ユーザー名とグループ名にはパスを含めることができます。IAM パスに使用できる文字は、英数字と、スラッシュ (/)、プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (\_)、ハイフン (-) のみです。

### パスでのワイルドカードの使用

パスには、ワイルドカード文字、アスタリスク (\*) を含めることができます。たとえば、IAM ポリシーを記述する場合、次のようなワイルドカードを使用して、パス `product_1234` を持つすべての IAM ユーザーを指定できます。

```
arn:aws:iam::123456789012:user/Development/product_1234/*
```

同様に、次の例に示すように、全ユーザーを意味する `user/*` や全グループを意味する `group/*` を指定できます。

```
"Resource": "arn:aws:iam::123456789012:user/*"  
"Resource": "arn:aws:iam::123456789012:group/*"
```

次の例は、リソース名にパスが含まれる Amazon S3 バケットの ARN を示しています。

```
arn:aws:s3::my_corporate_bucket/*  
arn:aws:s3::my_corporate_bucket/Development/*
```

### ワイルドカードの使用方法が正しくありません

IAM ARN の用語 `user` など、リソースタイプを指定する ARN の一部では、ワイルドカードを使用することはできません。例えば、以下のことは許可されません。

```
arn:aws:iam::123456789012:u* <== not allowed
```

## IAM ID

IAM は、ユーザー、ユーザーグループ、ロール、ポリシー、およびサーバー証明書に対していくつかの異なる ID を使います。このセクションでは、ID およびそれぞれの ID の使い方について説明します。

### トピック

- [フレンドリ名とパス](#)
- [IAM ARN](#)
- [一意の識別子](#)

## フレンドリ名とパス

ユーザー、ロール、ユーザーグループ、またはポリシーを作成する場合、またはサーバー証明書をアップロードする場合は、わかりやすい名前を付けます。たとえば、Bob、TestApp1、開発者、ManageCredentialsPermissions、ProdServerCert などがあります。

IAM API または AWS Command Line Interface (AWS CLI) を使用して IAM リソースを作成する場合、オプションのパスを追加できます。単一のパスを使用することも、複数のパスをフォルダー構造としてネストすることもできます。たとえば、会社の組織構造に一致するように、入れ子パス /division\_abc/subdivision\_xyz/product\_1234/engineering/ を使用できます。その後、パスのユーザーすべてが Policy Simulator API にアクセス許可するためのポリシーを作成できます。このポリシーを表示するには、[IAM: ユーザーパスに基づいた Policy Simulator API へのアクセス](#) を参照してください。フレンドリ名の指定方法については、[User API のドキュメント](#) を参照してください。パスの使用法のその他の例については、[IAM ARN](#) を参照してください。

AWS CloudFormation を使用してリソースを作成する場合、ユーザー、ユーザーグループ、ロール、およびカスタマーマネージドポリシーのパスを指定できます。

同じパスにユーザーとユーザーグループがある場合、IAM はそのユーザーグループに自動的に配置されません。例えば、Developers ユーザーグループを作成し、そのパスを /division\_abc/subdivision\_xyz/product\_1234/engineering/ と指定したとします。Bob という名のユーザーを追加して同様のパスを追加しても、自動的に Bob が Developers ユーザーグループに分類されるわけではありません。IAM は、パスに基づいてユーザーまたはユーザーグループ間の境界を強制しません。パスの異なるユーザーでも、それらのリソースにアクセス許可さえ与えられていれば )、

同じリソース使うことができます。AWS アカウントの IAM リソースの数とサイズには制限があります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。

## IAM ARN

ほとんどのリソースはフレンドリ名を持っています (Bob という名前のユーザー、Developers という名前のユーザーグループなど)。ただし、アクセス許可ポリシー言語では、以下の Amazon リソースネーム (ARN) 形式を使用して、リソースを指定する必要があります。

```
arn:partition:service:region:account:resource
```

コードの説明は以下のとおりです。

- `partition` は、そのリソースがあるパーティションを識別します。標準の AWS リージョンの場合、パーティションは `aws` です。他のパーティションにリソースがある場合、パーティションは `aws-partitionname` です。例えば、中国 (北京) リージョンにあるリソースのパーティションは、`aws-cn` です。異なるパーティションのアカウント間で[アクセスを委任](#)することはできません。
- `service` は AWS 製品を示します。IAM リソースは常に `iam` を使用します。
- `region` は、リソースのリージョンを識別します。IAM リソースの場合、これは常に空白にしておきます。
- `account` が、ハイフンなしの AWS アカウント ID を指定します。
- `resource` は、特定のリソースを名前で識別します。

IAM および AWS STS ARN は、次の構文を使用して指定できます。IAM リソースはグローバルに識別されるため、ARN のリージョンの割り当ては空白です。

構文:

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:sts::account:self
arn:aws:iam::account:mfa/virtual-device-name-with-path
```

```
arn:aws:iam::account:u2f/u2f-token-id  
arn:aws:iam::account:server-certificate/certificate-name-with-path  
arn:aws:iam::account:saml-provider/provider-name  
arn:aws:iam::account:oidc-provider/provider-name
```

次の例の多くには、ARNのリソースパートにパスが含まれています。パスをAWS Management Consoleで作成、操作することはできません。パスを使用するには、リソースを操作するには、AWS API、AWS CLI または Windows PowerShell 用 Tools for Windows PowerShell を使用する

例:

```
arn:aws:iam::123456789012:root  
arn:aws:iam::123456789012:user/JohnDoe  
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe  
arn:aws:iam::123456789012:group/Developers  
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers  
arn:aws:iam::123456789012:role/S3Access  
arn:aws:iam::123456789012:role/application_abc/component_xyz/RDSAccess  
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/  
AWSServiceRoleForAccessAnalyzer  
arn:aws:iam::123456789012:role/service-role/QuickSightAction  
arn:aws:iam::123456789012:policy/UsersManageOwnCredentials  
arn:aws:iam::123456789012:policy/division_abc/subdivision_xyz/UsersManageOwnCredentials  
arn:aws:iam::123456789012:instance-profile/Webserver  
arn:aws:sts::123456789012:federated-user/JohnDoe  
arn:aws:sts::123456789012:assumed-role/Accounting-Role/JaneDoe  
arn:aws:sts::123456789012:self  
arn:aws:iam::123456789012:mfa/JaneDoeMFA  
arn:aws:iam::123456789012:u2f/user/JohnDoe/default (U2F security key)  
arn:aws:iam::123456789012:server-certificate/ProdServerCert  
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/  
ProdServerCert  
arn:aws:iam::123456789012:saml-provider/ADFSPProvider  
arn:aws:iam::123456789012:oidc-provider/GoogleProvider  
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/  
a1b2c3d4567890abcdefEXAMPLE11111  
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

以下の例は、さまざまなタイプの IAM および AWS STS リソースの ARN 形式を理解するのに役立つ詳細情報を示します。

- アカウントの IAM ユーザー:

**Note**

各 IAM ユーザー名 は一意です。ユーザー名は、サインイン処理など、ユーザーに対して大文字と小文字が区別されませんが、ポリシー内で使用する場合や ARN の一部として使用する場合は大文字と小文字が区別されます。

```
arn:aws:iam::123456789012:user/JohnDoe
```

- 組織図を反映するパスを持つ別のユーザー:

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
```

- IAM ユーザーグループの場合:

```
arn:aws:iam::123456789012:group/Developers
```

- パスを持つ IAM ユーザーグループ:

```
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
```

- IAM ロール:

```
arn:aws:iam::123456789012:role/S3Access
```

- [サービスにリンクされたロール](#):

```
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/  
AWSServiceRoleForAccessAnalyzer
```

- [サービスロール](#):

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
```

- 管理ポリシー:

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- Amazon EC2 インスタンスと関連付けることができるインスタンスプロファイル:



```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- 「Paulo」として IAM で識別されるフェデレーテッドユーザー:

```
arn:aws:sts::123456789012:federated-user/Paulo
```

- 引き受けるロールが「Accounting-Role」、ロールのセッション名が「Mary」であるユーザーのアクティブセッション:

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- 呼び出し元のセッションに対して操作を行う API コール (AWS STS [SetContext](#) API など) でリソースとして使用された場合、呼び出し元のセッションを表します。

```
arn:aws:sts::123456789012:self
```

- Jorge という名前のユーザーに割り当てられた多要素認証デバイス:

```
arn:aws:iam::123456789012:mfa/Jorge
```

- サーバー証明書:

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- 組織図を反映したパスを持つサーバー証明書:

```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/  
ProdServerCert
```

- ID プロバイダー (SAML および OIDC):

```
arn:aws:iam::123456789012:saml-provider/ADFSPROvider  
arn:aws:iam::123456789012:oidc-provider/GoogleProvider  
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

- Amazon EKS OIDC ID プロバイダー URL を反映するパスを持つ OIDC ID プロバイダー:

```
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/  
a1b2c3d4567890abcdefEXAMPLE11111
```

もう 1 つの重要な ARN は ルートユーザー ARN です。これは IAM リソースではありませんが、この ARN の形式に精通している必要があります。多くの場合、リソースベースのポリシーの [Principal 要素](#) で使用されます。

- AWS アカウント は次の項目を表示します。

```
arn:aws:iam::123456789012:root
```

以下に示すポリシーでは、Richard に対して、本人のアクセスキーを管理する権限を割り当てています。リソースは、IAM ユーザー Richard であることに注目してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageRichardAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/division_abc/subdivision_xyz/Richard"
    },
    {
      "Sid": "ListForConsole",
      "Effect": "Allow",
      "Action": "iam:ListUsers",
      "Resource": "*"
    }
  ]
}
```

#### Note

ARN を使用して IAM ポリシー内のリソースを識別する場合、ポリシー変数を含めることができます。ポリシー変数には、ARN の一部としてランタイム情報 ( ユーザー名など ) のプレースホルダを含めることができます。詳細については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## ARN でのワイルドカードとパスの使用

ARN の *resource* 部分でワイルドカードを使用して、複数のユーザー、ユーザーグループ、またはポリシーを指定することもできます。例えば、product\_1234 の作業に携わっている全ユーザーを表すには、次のように指定します。

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

名前が app\_ という文字列で始まるユーザーがいる場合、次の ARN を使用してこれらすべてのユーザーを参照ができます。

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

AWS アカウント の全ユーザー、ユーザーグループ、またはポリシーを表すには、それぞれ ARN の user/、group/、または policy/ に続いて、ワイルドカードを指定します。

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
arn:aws:iam::123456789012:policy/*
```

ユーザー arn:aws:iam::111122223333:user/\* に次の ARN を指定すると、次の両方の例に一致します。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

ただし、ユーザー arn:aws:iam::111122223333:user/division\_abc\* に次の ARN を指定すると、2番目の例と一致しますが、最初の例には一致しません。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

ARN の user/、group/、または policy/ 部分にワイルドカードを使用しないでください。例えば、IAM では次のことは許可されません。

```
arn:aws:iam::123456789012:u*
```

## Example プロジェクトベースのユーザーグループ用のパスと ARN の使用例

パスをAWS Management Consoleで作成、操作することはできません。パスを使用するには、AWS API、AWS CLI、またはTools for Windows PowerShellを使用してリソースを操作する必要があります。

この例では、Marketing\_Admin ユーザーグループの Jules が /marketing/ パス内にプロジェクトベースのユーザーグループを作成します。Jules は、会社のさまざまな部分のユーザーをユーザーグループに割り当てます。この例では、ユーザーのパスが本人の所属するユーザーグループには関連のないことを示しています。

マーケティング部門には、リリース間近の新製品があり、そのため Jules は /marketing/ パスに Widget\_Launch という新しいユーザーグループを作成します。次に、Jules はグループに以下のポリシーを割り当てます。これにより、Widget\_Launch ユーザーグループには、新製品プロジェクト用に確保された example\_bucket 領域のオブジェクトにアクセスできる権限が与えられます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket*",
      "Resource": "arn:aws:s3:::example_bucket",
      "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/widget/*"}}
    }
  ]
}
```

そして Jules は、この新製品プロジェクトに参加しているユーザーを Widget\_Launch ユーザーグループに割り当てます。具体的には、/marketing/ パスから Patricia と Eli が、/sales/ パスからは Chris と Chloe が、/legal/ パスからは Alice と Jim が追加されます。

## 一意の識別子

IAM がユーザー、ユーザーグループ、ロール、ポリシー、インスタンスプロファイル、サーバー証明書を作成するとき、各リソースには一意の ID が割り当てられます。一意の ID は次のようになります。

```
AIDAJQABLZS4A3QDU576Q
```

ほとんどの場合、IAM リソースを操作するときは、フレンドリ名と [ARN](#) を使用します。そうすれば、特定のリソースの一意の ID を知る必要はありません。しかしながら、フレンドリ名を用いることが実際的でないときなど、場合によっては一意の ID が役立つこともあります。

1 つの例では、AWS アカウント 内でフレンドリ名を再利用します。アカウント内において、ユーザー、ユーザーグループ、ロール、ポリシーに対するフレンドリ名は一意である必要があります。たとえば、John という IAM ユーザーを作成するとしましょう。会社は、Amazon S3 を使用し、各従業員用のフォルダがあるバケットを持っています。IAM ユーザー John は、バケット内の自分のフォルダにのみアクセスが許可される User-S3-Access という名前の IAM ユーザーグループのメンバーです。IAM ユーザーのフレンドリ名を使用して S3 内の自分のバケットオブジェクトへのアクセスを許可するアイデンティティベースのポリシーの作成例については、「[Amazon S3: IAM ユーザーが自分の S3 ホームディレクトリにプログラムによりコンソールでアクセスすることを許可する](#)」を参照してください。

そこで、John という名の従業員が退職することになり、John という名の関連する IAM ユーザーを削除するとします。しかし後に、John という名前を持つ別の従業員が入社してきて、John という名の新しい IAM ユーザーを作成することになりました。John という名前の新しい IAM ユーザーが、既存の IAM ユーザーグループ User-S3-Access に追加されます。ユーザーグループに関連付けられたポリシーで、John という名前の IAM ユーザーのフレンドリ名が指定されていた場合、このポリシーにより、新しい John は、以前の John によって残された情報にアクセスできるようになります。

一般に、ポリシーでは、一意の ID ではなく、リソースの ARN を指定することをお勧めします。一方で、新しい IAM ユーザーが以前削除したフレンドリ名を再利用したとしても、すべてのユーザーが一意の ID を持っています。この例では、前の IAM ユーザーである John と、新しい IAM ユーザーである John は、それぞれ異なる一意の ID を持ちます。ユーザー名だけでなく一意の ID でアクセスを許可するリソースポリシーを作成できます。これにより、従業員が所有してはならない情報へのアクセスを誤って許可する可能性が低くなります。

次の例は、リソースベースのポリシーの [Principal 要素](#) に一意の ID を指定する方法を示しています。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:role/role-name",
    "AIDACKCEVSQ6C2EXAMPLE",
    "AROADBQP57FF2AEXAMPLE"
  ]
}
```

次の例は、グローバル条件キー [aws:userid](#) を使用して、ポリシーの [Condition 要素](#) に一意の ID を指定する方法を示しています。

```
"Condition": {
  "StringLike": {
    "aws:userId": [
      "AIDACKCEVSQ6C2EXAMPLE",
      "AROADBQP57FF2AEXAMPLE:role-session-name",
      "AROA1234567890EXAMPLE:*",
      "111122223333"
    ]
  }
}
```

また IAM ユーザーまたはロールの情報に関する独自のデータベース (またはその他のデータストア) を維持するときなども、ユーザー ID が役立ちます。一意の ID を使用すれば、IAM ユーザーまたはロールを作成する際に、それぞれに一意の識別子を割り当てられます。この例も前の例と同様、名前を再度使用する IAM ユーザーまたはロールが存在するケースです。

## 一意の ID プレフィックスを理解する

IAM は、一意の各 ID が適用するリソースタイプを示すために、以下のプレフィックスを使用します。プレフィックスは、作成日時によって異なる場合があります。

プレフィックス	リソースタイプ
ABIA	<a href="#">AWS STS サービスベアラートークン</a>
ACCA	コンテキスト固有の認証情報
AGPA	ユーザーグループ
AIDA	IAM ユーザー

プレフィックス	リソースタイプ
AIPA	Amazon EC2 インスタンスプロファイル
AKIA	アクセスキー
ANPA	マネージドポリシー
ANVA	管理ポリシーのバージョン
APKA	パブリックキー
AROA	ロール
ASCA	証明書
ASIA	<a href="#">一時 (AWS STS) アクセスキー ID</a> はこのプレフィックスを使用しますが、シークレットアクセスキーとセッショントークンとの組み合わせでのみ一意です。

## 一意識別子を取得する

IAM リソースに対する一意の ID は、IAM コンソールでは取得できません。一意の ID を取得するには、以下の AWS CLI コマンドまたは IAM API コールを使用します。

AWS CLI:

- [get-caller-identity](#)
- [get-group](#)
- [get-role](#)
- [get-user](#)
- [get-policy](#)
- [get-instance-profile](#)
- [get-server-certificate](#)

IAM API:

- [GetCallerIdentity](#)
- [GetGroup](#)
- [GetRole](#)
- [GetUser](#)
- [GetPolicy](#)
- [GetInstanceProfile](#)
- [GetServerCertificate](#)

## IAM と AWS STSクォータ

AWS Identity and Access Management (IAM) と AWS Security Token Service (STS) には、オブジェクトのサイズを制限するクォータがあります。これは、オブジェクトに名前を付ける方法、作成できるオブジェクトの数、オブジェクトを渡すときに使用できる文字数に影響します。

### Note

IAM の使用状況とクォータに関するアカウントレベルの情報を取得するには、[GetAccountSummary](#) API オペレーション、または [get-account-summary](#) AWS CLI コマンドを使用します。

## IAM 名前の要件

IAM の名前には以下の要件と制約があります。

- ポリシードキュメントには、次の Unicode 文字のみを含めることができます。水平タブ (U+0009)、ラインフィード (U+000A)、キャリッジリターン (U+000D)、および U+0020~U+00FF の範囲内の文字。
- ユーザー、グループ、ロール、ポリシー、インスタンスプロファイル、サーバー証明書、およびパスの名前は、英数字で指定する必要があります。これには、プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (\_)、ハイフン (-) も含まれます。パス名の前後にはスラッシュ (/) を指定する必要があります。
- ユーザー、グループ、ロール、インスタンスプロファイルの名前は、アカウント内で一意である必要があります。大文字と小文字は区別されません。例えば、「ADMINS」と「admins」というグループ名を両方作成することはできません。



- ロールを引き受けるためにサードパーティーが使用する外部 ID の値は、2~1,224 文字で構成されている必要があります。この値は、空白のない英数字にする必要があります。次の記号を含めることもできます。プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、コロンの (:)、スラッシュ (/)、およびハイフン (-)。外部 ID の詳細については、[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#) を参照してください。
- [インラインポリシー](#) のポリシー名は、それが埋め込まれているユーザー、グループ、またはロールに対して一意である必要があります。名前には、次の予約文字を除く基本ラテン (ASCII) 文字を含めることができます: バックスラッシュ (\)、スラッシュ (/)、アスタリスク (\*)、疑問符 (?)、空白。これらの文字は [RFC 3986](#)、[セクション 2.2](#) に従って予約されています。
- ユーザーパスワード (ログインプロファイル) には、基本ラテン (ASCII) 文字を含めることができます。
- AWS アカウント ID のエイリアスは、AWS 製品全体で一意となるのもので、DNS の命名規則に従って英数字にする必要があります。エイリアスには小文字を使用する必要があり、先頭や末尾にハイフンを使用することはできません。また、ハイフンを 2 つ連続することも、12 桁の数字にすることもできません。

基本ラテン (ASCII) 文字の一覧については、「[Library of Congress Basic Latin \(ASCII\) Code Table](#)」を参照してください。

## IAM オブジェクトクォータ

AWS のクォータ (制限とも呼ばれます) は、AWS アカウント のリソース、アクション、および制限の最大値です。Service Quotas を使用して IAM クォータを管理します。

IAM サービスエンドポイントとサービスクォータのリストについては、「AWS 全般のリファレンス」の「[AWS Identity and Access Management エンドポイントとクォータ](#)」を参照してください。

クォータの引き上げをリクエストするには

1. AWS Management Console にサインインするには、「AWS サインインユーザーガイド」の「[AWS へのサインイン方法](#)」のトピックで説明されているように、ユーザータイプに適したサインイン手順に従います。
2. Service Quotas コンソール を開きます。
3. ナビゲーションペインで、[AWS services] (AWS のサービス) を選択します。
4. ナビゲーションバーで、[US East (N. Virginia)] リージョンを選択します。次に、IAM を検索します。

5. AWS Identity and Access Management (IAM) を選択し、クォータを選択して、指示に従ってクォータの引き上げをリクエストします。

詳細については、Service Quotas ユーザーガイドの [Requesting a Quota Increase](#) を参照してください。

Service Quotas コンソールを使用して IAM クォータの増加をリクエストする方法の例については、次のビデオをご覧ください。

[Service Quotas コンソールを使用して IAM クォータの増加をリクエストする。](#)

調整可能な IAM クォータではデフォルトのクォータの引き上げをリクエストできます。 [maximum quota](#) までのリクエストは自動的に承認され、数分で完了します。

次の表には、クォータの引き上げが自動的に承認される範囲についてリソースの一覧が掲載されています。

#### IAM リソースの調整可能なクォータ

リソース	デフォルトのクォータ	最大クォータ
アカウントあたりのカスタマー管理ポリシー数	1500	5000
アカウントあたりのグループ数	300	500
アカウントあたりのインスタンスプロファイル数	1000	5000
ロールあたりの管理ポリシー	10	20
ユーザーあたりの管理ポリシー	10	20
ロールの信頼ポリシーの長さ	2048 文字	4096 文字
アカウントあたりのロール数	1000	5000
アカウントあたりのサーバー証明書数	20	1000

## IAM Access Analyzer のクォータ

IAM Access Analyzer のサービスエンドポイントとサービスクォータの一覧については、「AWS 全般のリファレンス」の「[IAM Access Analyzer エンドポイントとクォータ](#)」を参照してください。




## IAM Roles Anywhere クォータ

IAM Roles Anywhere のサービスエンドポイントとサービスクォータのリストについては、「AWS 全般のリファレンス」の「[AWS Identity and Access Management ロール Anywhere エンドポイントとクォータ](#)」を参照してください。

## IAM 文字制限および STS 文字制限

IAM および AWS STS の最大文字数とサイズの制限は、次のとおりです。以下の制限の引き上げをリクエストすることはできません。


説明	制限
AWS アカウント ID のエイリアス	3 ~ 63 文字
<a href="#">インラインポリシー</a> の場合	<p>IAM ユーザー、ロール、またはグループに必要な数のインラインポリシーを追加できます。ただし、エンティティごとの総ポリシーサイズ (すべてのインラインポリシーの合計サイズ) は以下の制限を超えることはできません。</p> <ul style="list-style-type: none"><li>ユーザーポリシーサイズは 2,048 文字を超えることはできません。</li><li>ロールポリシーサイズは 10,240 文字を超えることはできません。</li><li>グループポリシーサイズは 5,120 文字を超えることはできません。</li></ul>

説明	制限
	<p> <b>Note</b></p> <p>IAM ではこれらの制限に対するポリシーのサイズを計算する際に空白をカウントしません。</p>
<p><u>管理ポリシーの場合</u></p>	<ul style="list-style-type: none"> <li>各管理ポリシーのサイズは、6,144 文字を超えることはできません。</li> </ul> <p> <b>Note</b></p> <p>IAM ではこの制限に対するポリシーのサイズを計算する際に空白をカウントしません。</p>
グループ名	128 文字
インスタンスプロファイル名	128 文字
ログインプロファイルのパスワード	1 ~ 128 文字
パス	512 文字
ポリシー名	128 文字
ロール名	<p>64 文字</p> <p> <b>Important</b></p> <p>AWS Management Console の [ロールの切り替え] 機能でロールを使用する場合は、Path と RoleName の合計が 64 文字を超えることはできません。</p>

説明	制限
ロールセッションの期間	12 時間ごと  AWS CLI または API からロールを引き受けると、duration-seconds CLI パラメータまたは DurationSeconds API パラメータを使用して、より長いロールセッションをリクエストできます。900 秒 (15 分) からロールの最大セッション期間設定 (1 時間 ~ 12 時間の範囲) までの値を指定できます。DurationSeconds パラメータの値を指定しない場合、セキュリティ認証情報は 1 時間有効です。コンソールでロールを切り替える IAM ユーザーには、最大セッション期間、またはユーザーのセッションの残り時間のいずれか短い方が付与されます。最大セッション期間の設定では、AWS のサービスが引き受けるセッションは制限されません。ロールの最大値を確認する方法については、「 <a href="#">ロールの最大セッション期間設定の表示</a> 」を参照してください。
ロールセッション名	64 文字

説明	制限
ロール <a href="#">セッションポリシー</a>	<ul style="list-style-type: none"><li>渡された JSON ポリシードキュメントと渡されたすべての管理ポリシー ARN 文字の合計サイズは、2,048 文字を超えることはできません。</li><li>セッションを作成するときに、最大 10 個のマネージドポリシー ARN を渡すことができます。</li><li>ロールまたはフェデレーテッドユーザーの一時セッションをプログラムで作成するときに渡すことができる JSON ポリシードキュメントは 1 つだけです。</li><li>また、AWS 変換では、渡されたセッションポリシーとセッションタグが、個別の制限を持つ一括のバイナリ形式に圧縮されます。PackedPolicySize レスポンス要素は、リクエストのポリシーとタグがサイズ制限にどの程度近づいているかをパーセントで示します。</li><li>AWS CLI または AWS API を使用してセッションポリシーを渡すことをお勧めします。AWS Management Console は、パックされたポリシーにコンソールセッション情報を追加することがあります。</li></ul>

説明	制限
ロール <a href="#">セッションタグ</a>	<ul style="list-style-type: none"><li>セッションタグは、タグキーの制限 128 文字、タグ値の制限 256 文字を満たす必要があります。</li><li>最大 50 個のセッションタグを渡すことができます。</li><li>AWS 変換では、渡されたセッションポリシーとセッションタグが、個別の制限を持つひとまとめのバイナリ形式に圧縮されます。セッションタグは、AWS CLI または AWS API を使用して渡すことができます。PackedPolicySize レスポンス要素は、リクエストのポリシーとタグがサイズ制限にどの程度近づいているかをパーセントで示します。</li></ul>
SAML 認証レスポンス base64 エンコード	100,000 文字  この文字制限は <a href="#">assume-role-with-saml</a> CLI または <a href="#">AssumeRoleWithSAML</a> API オペレーションに適用されます。
タグキー	128 文字  この文字制限は、IAM リソースと <a href="#">セッションタグ</a> に適用されます。
タグ値	256 文字  この文字制限は、IAM リソースと <a href="#">セッションタグ</a> に適用されます。  タグの値は 0 文字にすることができます。つまり、タグの値は 0 文字にすることができます。

説明	制限
IAM によって作成された一意の ID	128 文字 例:。 <ul style="list-style-type: none"><li>• AIDA で始まるユーザー ID</li><li>• AGPA で始まるグループ ID</li><li>• AROA で始まるロール ID</li><li>• ANPA で始まる管理ポリシー ID</li><li>• ASCA で始まるサーバー証明書 ID</li></ul> <div data-bbox="829 646 1507 961"><p> Note</p><p>これは網羅的なリストではありません。また、特定のタイプの ID が必ずしも指定された文字の組み合わせのみで始まるとは限りません。</p></div>
[User name] (ユーザー名)	64 文字

## インターフェイス VPC エンドポイント

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストすると、VPC と AWS Security Token Service (AWS STS) の間のプライベート接続を確立できます。この接続を使用すると、AWS STS はパブリックインターネットを経由せずに、VPC 内のリソースと通信できます。

Amazon VPC は、ユーザー定義の仮想ネットワークで AWS リソースを起動するために使用できる AWS のサービスです。VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC を AWS STS に接続するには、AWS STS の インターフェイス VPC エンドポイントを定義します。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続を必要とせず、信頼性が高くスケーラブルな AWS STS への接続を提供します。詳細については、Amazon VPC ユーザーガイドの [「Amazon VPC とは」](#) を参照してください。



インターフェイス VPC エンドポイントは AWS PrivateLink を利用しています。これは、Elastic Network Interface とプライベート IP アドレスを使用して AWS のサービス間のプライベート通信を可能にする AWS のテクノロジーです。詳細については、「[AWS のサービス向けの AWS PrivateLink](#)」を参照してください。

以下の情報は Amazon VPC のユーザーを対象としています。詳細については、「Amazon VPC ユーザーガイド」の「[Amazon VPC の使用開始](#)」を参照してください。

## 可用性

現在、AWS STS は、次のリージョンで VPC エンドポイントをサポートしています。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (メルボルン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)

- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

## AWS STS の VPC エンドポイントを作成する

VPC で AWS STS の使用を開始するには、AWS STS のインターフェイス VPC エンドポイントを作成します。詳細については、「Amazon VPC ユーザーガイド」の「[インターフェイス VPC エンドポイントを使用して AWS のサービスにアクセスする](#)」を参照してください。

VPC エンドポイントを作成した後は、一致するリージョンのエンドポイントを使用して AWS STS リクエストを送信する必要があります。AWS STS では、`setRegion` メソッドと `setEndpoint` メソッドの両方を使用してリージョンのエンドポイント呼び出すことを推奨しています。`setRegion` メソッドは、アジアパシフィック (香港) など、手動で有効になっているリージョンに単独で使用できます。この場合、呼び出しは STS リージョン別エンドポイントに送信されます。手動でリージョンを有効にする方法については、「AWS 全般のリファレンス」の「[AWS リージョンの管理](#)」を参照してください。デフォルトで有効になっているリージョンに `setRegion` メソッドを単独で使用する場合、呼び出しは <https://sts.amazonaws.com> のグローバルエンドポイントに送信されます。

リージョンのエンドポイントを使用すると、AWS STS は、パブリックエンドポイントまたはプライベートインターフェイス VPC エンドポイントのうち使用中のいずれかを使って、他の AWS のサービス呼び出します。たとえば、AWS STS のインターフェイス VPC エンドポイントを作成し、VPC にあるリソースの一時的な認証情報を AWS STS からリクエスト済みであるとし、その場合、これらの認証情報は、デフォルトではそのインターフェイス VPC エンドポイントを経由し

で流れ始めます。AWS STS を使用してリージョンのリクエストを作成する方法の詳細については、「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

## IAM と連携する AWS のサービス


以下に一覧表示されている AWS のサービスは、アルファベット順にグループ化されています。また、サポートされる IAM の機能に関する情報を含んでいます。






- サービス – サービスの名前を選択し、このサービスの IAM 認証とアクセスに関する AWS ドキュメントを表示できます。
- アクション – ポリシーで個々のアクションを指定できます。サービスでこの機能がサポートされていない場合、[\[visual editor\]](#) (ビジュアルエディタ) で [\[All actions\]](#) (すべてのアクション) を選択します。JSON ポリシードキュメントでは、\* 要素に Action を使用する必要があります。各サービスのアクションのリストについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。
- リソースレベルのアクセス許可 – [ARN](#) を使用してポリシーで個々のリソースを指定できます。サービスでこの機能がサポートされていない場合、[\[policy visual editor\]](#) (ポリシービジュアルエディタ) で [\[All resources\]](#) (すべてのリソース) を選択します。JSON ポリシードキュメントでは、\* 要素に Resource を使用する必要があります。List\* アクションなど一部のアクションは、複数のリソースを返すように設計されているため、ARN の指定をサポートしていません。サービスでこの機能があるリソースでサポートされている場合は、その旨が表の [\[Partial\]](#) (一部) で示されます。詳細については、該当するサービスのドキュメントを参照してください。
- リソースベースのポリシー – リソースベースのポリシーをサービス内のリソースにアタッチできます。リソースベースのポリシーには、リソースにアクセスできる IAM ID を指定する Principal 要素が含まれます。詳細については、「[アイデンティティベースおよびリソースベースのポリシー](#)」を参照してください。
- ABAC (タグに基づく承認) – タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの [\[condition element\]](#) (条件要素) でタグ情報を提供します。サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーすべてをサポートする場合、値は部分的です。タグなどの属性に基づくアクセス許可の定義の詳細については、「[AWS の ABAC とは](#)」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、[属性に基づくアクセスコントロール \(ABAC\) を使用する](#)を参照してください。
- 一時的な認証情報 – IAM Identity Center を使用してサインインするときに取得したり、コンソールでロールを切り替えたり、AWS CLI または AWS API の AWS STS を使用して生成するときに


















取得する短期的な認証情報を使用できます。長期的な IAM ユーザー認証情報を使用している間のみ、[No] (いいえ) の値を使用してサービスにアクセスできます。これには、ユーザー名とパスワード、またはユーザーアクセスキーが含まれます。詳細については、「[IAM の一時的な認証情報](#)」を参照してください。





































- サービスにリンクされたロール – [サービスにリンクされたロール](#)は、ユーザーに代わって他のサービスのリソースにアクセスするアクセス許可をサービスに与える特殊なタイプのサービスロールです。これらのロールをサポートするサービスのドキュメントを参照するには、はいまたは部分的のリンクを選択してください。この列は、サービスが標準のサービスロールを使用するかどうかを示しません。詳細については、「[サービスリンクロールの使用](#)」を参照してください。
- 詳細情報 – サービスが機能を完全にサポートしていない場合は、エントリの脚注を確認して、制限および関連情報へのリンクを参照できます。


## IAM と連携するサービス

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Account Management</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Activate Console</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ




























サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Amplify Admin</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Amplify</a>	 はい	 はい	 いいえ	 部分的	 はい	 いいえ
<a href="#">AWS Amplify UI Builder</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon MSK クラスター用の Apache Kafka API</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon API Gateway</a>	 はい	 はい	 はい	 いいえ	 はい	 <a href="#">はい</a>
<a href="#">Amazon API Gateway Management</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ








サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon API Gateway Management V2</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS App2Container</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS AppConfig</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS AppFabric</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon AppFlow</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon アプリインテグレーション</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Application Auto Scaling</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Application Cost Profiler</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Application Discovery Arsenal</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Application Discovery Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Application Migration Service</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS アプリケーション変換サービス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ




























サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS App Mesh</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS App Mesh プレビュー</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS App Runner</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon AppStream 2.0</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS AppSync</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Artifact</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
























































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Athena</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Audit Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Auto Scaling</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS B2B Data Interchange</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Backup</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>
<a href="#">AWS Backup ゲートウェイ</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ


























サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Backup ストレージ</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Batch</a>	 はい	 <small>部</small> <u>分的</u>	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Bedrock</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Billing and Cost Management</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Billing and Cost Management Data Exports</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Billing Conductor</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Braket</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Budget Service</a>	 はい	 はい	 いいえ	 いいえ	 いいえ	 いいえ
<a href="#">AWS BugBust</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Certificate Manager (ACM)</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Chatbot</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon Chime</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Clean Rooms</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Clean Rooms ML</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Client VPN</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Cloud9</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>
<a href="#">AWS クラウド Control API</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Cloud Directory</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ




































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS CloudFormation</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudFront</a>	 はい	 はい	 いいえ	 部分的	 はい	 部分的 (情報)
<a href="#">Amazon CloudFront KeyValueStore</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS CloudHSM</a>	 はい	 はい	 いいえ	 はい	 はい	 はい
<a href="#">AWS Cloud Map</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CloudSearch</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS CloudShell</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS CloudTrail</a>	 はい	 はい	 <a href="#">部分的 (情報)</a>	 <a href="#">部分的 (情報)</a>	 はい	 <a href="#">はい</a>
<a href="#">AWS CloudTrailデータ</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">部分的 (情報)</a>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CloudWatch Application Insights</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon CloudWatch Application Signals</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch Evidently</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch Internet Monitor</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch Logs</a>	 はい	 はい	 はい	 <a href="#">部分的</a>	 はい	 <a href="#">はい</a>
<a href="#">Amazon CloudWatch Network Monitor</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CloudWatch Observability Access Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch RUM</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon CloudWatch Synthetics</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeArtifact</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 いいえ
<a href="#">AWS CodeBuild</a>	 はい	 はい	 はい (情報)	 部分的 (情報)	 はい	 いいえ



























































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CodeCatalyst</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS CodeCommit</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeConnections</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeDeploy</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeDeploy セキュアホストコマンドサービス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon CodeGuru Profiler</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>



































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CodeGuru Reviewer</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon CodeGuru Security</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodePipeline</a>	 はい	 部分的	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeStar</a>	 はい	 部分的	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS CodeStar 接続</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS CodeStar 通知</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon CodeWhisperer</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Cognito</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Cognito Sync</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon Cognito ユーザープール</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Comprehend</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Comprehend Medical</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Compute Optimizer</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Config</a>	 はい	 部分的 (情報)	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Connect</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Connect Cases</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Connect Customer Profiles</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Connect のハイブリッドなアウトバウンド通信</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Connect Voice ID</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Console Mobile Application</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS 一括請求 (コンソリデティッドビルディング)</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Control Catalog</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Control Tower</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ




































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Cost and Usage Report</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Cost Explorer</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Cost Optimization Hub</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS 顧客検証サービス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Database Migration Service</a>	 はい	 はい	 いいえ ( <a href="#">情報</a> )	 はい	 はい	 <a href="#">はい</a>































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Database Query Metadata Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Data Exchange</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Data Lifecycle Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Data Pipeline</a>	 はい	 はい	 いいえ	 部分的	 はい	 いいえ
<a href="#">AWS DataSync</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon DataZone</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ











サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS デッドラインクラウド</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS DeepComposer</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS DeepRacer</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Detective</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Device Farm</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon DevOps Guru</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>












































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS 診断ツール</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Direct Connect</a>	 はい	 はい	 いいえ	 <u>はい</u>	 はい	 <u>はい</u>
<a href="#">AWS Directory Service</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon DocumentDB Elastic Clusters</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon DynamoDB Accelerator (DAX)</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon DynamoDB</a>	 はい	 はい	 はい	 いいえ	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Elastic Compute Cloud (Amazon EC2)</a>	 はい	 部分的	 いいえ	 <u>はい</u>	 はい	 部分的 (情報)
<a href="#">Amazon EC2 Auto Scaling</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">EC2 Image Builder</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon EC2 Instance Connect</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon ElastiCache</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Elastic Beanstalk</a>	 はい	 部分的	 いいえ	 <u>はい</u>	 はい	 <u>はい</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Elastic Block Store (Amazon EBS)</a>	 はい	 部分的	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Elastic Container Registry (Amazon ECR)</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Elastic Container Registry パブリック (Amazon ECR パブリック)</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Elastic Container Service (Amazon ECS)</a>	 はい	 部分的 (情報)	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Elastic Disaster Recovery</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Elastic File System (Amazon EFS)</a>	 はい	 はい	 はい	 <a href="#">部分的</a>	 はい	 <a href="#">はい</a>
<a href="#">Amazon Elastic Inference</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Elastic Kubernetes Service (Amazon EKS)</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">Amazon Elastic Kubernetes Service (Amazon EKS) Auth</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Elastic Load Balancing</a>	 はい	 部分的	 いいえ	 部分的	 はい	 <a href="#">はい</a>
<a href="#">Amazon Elastic Transcoder</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Elemental Appliances と Software Activation Service</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Elemental アプライアンスとソフトウェア</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Elemental MediaConnect</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Elemental MediaConvert</a>	 はい	 はい	 いいえ	 <u>はい</u>	 はい	 いいえ
<a href="#">AWS Elemental MediaLive</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ


サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Elemental MediaPackage</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">部分的 (情報)</a>
<a href="#">AWS Elemental MediaPackage V2</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Elemental MediaPackage VOD</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">部分的 (情報)</a>
<a href="#">AWS Elemental MediaStore</a>	 はい	 はい	 はい	 はい	 はい	 いいえ
<a href="#">AWS Elemental MediaTailor</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>




サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Elemental のサポートケース</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Elemental のサポートコンテンツ</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon EMR</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon EMR on EKS</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon EMR Serverless</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Entity Resolution</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon EventBridge</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 いいえ
<a href="#">Amazon EventBridge パイプ</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon EventBridge スケジューラ</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon EventBridge スキーマ</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 いいえ
<a href="#">AWS Fault Injection Service</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon FinSpace</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>







































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon FinSpace API</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Firewall Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">部分的</a>
<a href="#">Fleet Hub for AWS IoT Device Management</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Forecast</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Fraud Detector</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">FreeRTOS</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS 無料利用枠</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon FSx</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">Amazon GameLift</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Global Accelerator</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">AWS Glue</a>	 はい	 はい	 はい	 <a href="#">部分的</a>	 はい	 いいえ
<a href="#">AWS Glue DataBrew</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ























サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Ground Station</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Ground Truth Labeling</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon GuardDuty</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Health API と通知</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS HealthImaging</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS HealthLake</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS HealthOmics</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Honeycode</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS IAM Identity Center</a>	 はい	 はい	 いいえ	 部分的	 はい	 <u>はい</u>
<a href="#">IAM Identity Center Directory</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">IAM Identity Center Identity Store</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">IAM Identity Center OIDC サービス</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Identity and Access Management (IAM)</a>	 はい	 はい	 部分的 (情報)	 部分的 (情報)	 部分的 (情報)	 いいえ
<a href="#">AWS Identity and Access Management Access Analyzer</a>	 はい	 はい	 いいえ	 はい	 はい	 部分的
<a href="#">AWS Identity and Access Management Roles Anywhere</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Identity Store Auth</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Identity Sync</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Import/Export</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Inspector</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Inspector Classic</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon InspectorScan</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Interactive Video Service</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Interactive Video Service Chat</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Invoicing</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS IoT 1-Click</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IoT Analytics</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IoT</a>	 <u>はい</u>	 <u>はい</u>	 部分的 (情報)	 <u>はい</u>	 はい	 いいえ
<a href="#">AWS IoT Core Device Advisor</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ




































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS IoT Device Tester</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS IoT Events</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IoT FleetWise</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IoT Greengrass</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IoT Greengrass V2</a>	 はい	 はい	 いいえ	 部分的	 はい	 いいえ
<a href="#">AWS IoT Jobs DataPlane</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ







































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS IoT RoboRunner</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS IoT SiteWise</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS IoT TwinMaker</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS IoT Wireless</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS IQ</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS IQ アクセス許可</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ



サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Kendra</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Kendra インテリジェントランキング</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Key Management Service (AWS KMS)</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Keyspaces (Apache Cassandra 向け)</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Managed Service for Apache Flink</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Managed Service for Apache Flink V2</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
















サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Data Firehose</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Kinesis Data Streams</a>	 はい	 はい	 はい	 いいえ	 はい	 いいえ
<a href="#">Amazon Kinesis Video Streams</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Lake Formation</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Lambda</a>	 はい	 はい	 <u>はい</u>	 <u>部分的 (情報)</u>	 はい	 <u>部分的 (情報)</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Launch Wizard</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Lex</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Lex V2</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 <u>はい</u>
<a href="#">AWS License Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS License Manager Linux サブスクリプションマネージャー</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS License Manager ユーザーサブスクリプション</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Lightsail</a>	 はい	 部分的 (情報)	 いいえ	 部分的 (情報)	 はい	 <u>はい</u>
<a href="#">Amazon Location Service</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Lookout for Equipment</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Lookout for Metrics</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Lookout for Vision</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ















サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Machine Learning</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Macie</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">AWS Mainframe Modernization</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">AWS Mainframe Modernization アプリケーションテスト</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Managed Blockchain</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Managed Blockchain Query</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Managed Grafana</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Managed Service for Prometheus</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Managed Streaming for Apache Kafka (MSK)</a>	 はい	 はい	 部分的 (情報)	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Managed Streaming for Kafka Connect</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon Managed Workflows for Apache Airflow</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ



































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Marketplace</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Marketplace カタログ</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Marketplace Commerce Analytics</a>	 はい	 いいえ	 いいえ	 いいえ	 いいえ	 いいえ
<a href="#">AWS Marketplace Deployment Service</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Marketplace Discovery</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace Entitlement Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ





























































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Marketplace イメージ構築サービス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace 管理ポータル</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace Metering Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace プライベートマーケットプレイス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace 調達システムの統合</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Marketplace 販売者レポート</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Marketplace Vendor Insights</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Mechanical Turk</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon MediaImport</a>	 はい	 いいえ	 いいえ	 いいえ	 いいえ	 いいえ
<a href="#">Amazon MemoryDB for Redis</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Message Delivery Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Message Gateway Service</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ




































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Microservice Extractor for .NET</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS移行促進プログラムのクレジット</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Migration Hub</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Migration Hub Orchestrator</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Migration Hub Refactor Spaces</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>
<a href="#">AWS Migration Hub Strategy Recommendations</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Monitron</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon MQ</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Neptune</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon Neptune Analytics</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Network Firewall</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Network Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい (情報)</u>





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Network Manager Chat</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Nimble Studio</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon One Enterprise</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon OpenSearch Ingestion</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon OpenSearch Serverless</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon OpenSearch Service</a>	 はい	 はい	 はい	 はい	 はい	 <u>はい</u>































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS OpsWorks</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS OpsWorks 設定管理</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Organizations</a>	 はい	 はい	 いいえ	 はい	 いいえ	 <u>はい</u>
<a href="#">AWS Outposts</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Panorama</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Partner セントラルのアカウント管理</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ





サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Payment Cryptography</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS 支払い</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Performance Insights</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Personalize</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Pinpoint</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Pinpoint Eメールサービス</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Pinpoint SMS および音声サービス</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Pinpoint SMS and Voice Service V2</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Polly</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS の料金表</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS プライベート 5G</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Private CA Connector for Active Directory</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Private CA Connector for SCEP</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Private Certificate Authority (AWS Private CA)</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 いいえ
<a href="#">AWS Proton</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Purchase Orders Console</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Q Business</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Q Business Q アプリ</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ





サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Q Developer</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">Amazon Q in Connect</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Quantum Ledger Database (Amazon QLDB)</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon QuickSight</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon RDS Data API</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon RDS IAM 認証</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS ごみ箱</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Redshift</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">Amazon Redshift Data API</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Redshift Serverless</a>	 はい	 はい	 はい	 はい	 はい	 いいえ
<a href="#">Amazon Rekognition</a>	 はい	 はい	 部分的 (情報)	 はい	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Relational Database Service (Amazon RDS) (情報)</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS re:Post Private</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Resilience Hub</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Resource Access Manager (AWS RAM)</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Resource Explorer</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Resource Groups</a>	 はい	 はい	 いいえ	 はい	 部分的 (情報)	 いいえ
<a href="#">AWS Resource Groups Tagging API</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon RHEL ナレッジベースポータル</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS RoboMaker</a>	 はい	 はい	 いいえ	 <u>はい</u>	 はい	 <u>はい</u>
<a href="#">Amazon Route 53</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Route 53 Application Recovery Controller - ゾーンシフト</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Route 53 ドメイン</a>	 はい	 いいえ	 いいえ	 いいえ	 いいえ	 いいえ
<a href="#">Amazon Route 53 プロファイル</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Route 53 Recovery クラスター</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Route 53 Recovery Control Config</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Route 53 Recovery 準備状況</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>



サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Route 53 Resolver</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon S3 Express</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon S3 Glacier</a>	 はい	 はい	 はい	 はい	 はい	 いいえ
<a href="#">Amazon SageMaker</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>部分</u> <u>的 (情報)</u>
<a href="#">Amazon SageMaker 地理空間機能</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ





































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon SageMaker</a> <a href="#">Ground Truth Synthetic</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Savings Plans</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Secrets Manager</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 いいえ
<a href="#">AWS Security Hub</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Security Lake</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Security Token Service (AWS STS)</a>	 はい	 部分的 (情報)	 いいえ	 はい	 部分的 (情報)	 いいえ
<a href="#">AWS Serverless Application Repository</a>	 はい	 はい	 はい	 いいえ	 はい	 いいえ
<a href="#">AWS Service Catalog</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Service Quotas</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Shield</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>







サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Signer</a>	 はい	 はい	 はい	 はい	 はい	 いいえ
<a href="#">AWS サインイン</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon SimpleDB</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Simple Email Service - メールマネージャー</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon Simple Email Service (Amazon SES) v2</a>	 はい	 部分的 (情報)	 はい	 はい	 部分的 (情報)	 <u>はい</u>



サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Simple Notification Service (Amazon SNS)</a>	 はい	 はい	 はい	 はい	 はい	 いいえ
<a href="#">Amazon Simple Queue Service (Amazon SQS)</a>	 はい	 はい	 はい	 <u>部分的</u>	 はい	 いいえ
<a href="#">Amazon Simple Storage Service (Amazon S3)</a>	 はい	 はい	 はい	 <u>部分的 (情報)</u>	 はい	 <u>部分的 (情報)</u>
<a href="#">Amazon Simple Storage Service (Amazon S3) オブジェクト Lambda</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Outposts での Amazon Simple Storage Service (Amazon S3)</a>	 はい	 はい	 はい	 いいえ	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Simple Workflow Service (Amazon SWF)</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS SimSpace Weaver</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Site-to-Site VPN</a>	 はい	 はい	 いいえ	 いいえ	 はい	 <u>はい</u>
<a href="#">AWS Snowball</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Snowball Edge</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Snow Device Management</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS SQL Workbench</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Step Functions</a>	 はい	 はい	 いいえ	 <u>はい</u>	 はい	 いいえ
<a href="#">AWS Storage Gateway</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Supply Chain</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Support App in Slack</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Support</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 <u>はい</u>






























サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Support プラン</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Support の推奨事項</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS サステナビリティ</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Systems Manager</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Systems Manager for SAP</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Systems Manager GUI Connect</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ


サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Systems Manager Incident Manager</a>	 はい	 はい	 <u>はい</u>	 はい	 はい	 <u>はい</u>
<a href="#">AWS Systems Manager Incident Manager お問い合わせ</a>	 はい	 はい	 <u>はい</u>	 いいえ	 はい	 いいえ
<a href="#">タグエディタ</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS 税金設定</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS 通信ネット ワークビルダー</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Textract</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ































サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Timestream</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Timestream Influxdb</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">AWS Tiros API (Reachability Analyzer 用)</a>	 はい	 いいえ	 いいえ	 いいえ	 いいえ	 いいえ
<a href="#">Amazon Transcribe</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Transfer Family</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon Translate</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ



サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS Trusted Advisor</a>	 部分的 (情報)	 はい	 いいえ	 いいえ	 部分的	 <a href="#">はい</a>
<a href="#">AWSユーザー通知</a>	 はい	 はい	 いいえ	 はい	 はい	 <a href="#">はい</a>
<a href="#">AWSユーザー通知連絡先</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS ユーザーサブスクリプション</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS Verified Access</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon Verified Permissions</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon Virtual Private Cloud (Amazon VPC)</a>	 はい	 部分的 (情報)	 部分的 (情報)	 はい	 はい	 部分的 (情報)
<a href="#">Amazon VPC Lattice</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon VPC Lattice サービス</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">AWS WAF</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">AWS WAF Classic</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS WAF リージョン別</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">AWS Well-Architected Tool</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS Wickr</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon WorkDocs</a>	 はい	 いいえ	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon WorkMail</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>

サービス	アクション	リソースレベルのアクセス許可	リソースベースのポリシー	ABAC	一時的な認証情報	サービスリンクロール
<a href="#">Amazon WorkMail</a> <a href="#">Message Flow</a>	 はい	 はい	 いいえ	 いいえ	 はい	 いいえ
<a href="#">Amazon WorkSpaces</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">Amazon WorkSpaces</a> <a href="#">セキュアブラウザ</a>	 はい	 はい	 いいえ	 はい	 はい	 <u>はい</u>
<a href="#">Amazon WorkSpaces</a> <a href="#">Thin Client</a>	 はい	 はい	 いいえ	 はい	 はい	 いいえ
<a href="#">AWS X-Ray</a>	 はい	 <u>部分的 (情報)</u>	 いいえ	 部分的 (情報)	 はい	 いいえ

## 詳細情報

### Amazon CloudFront

CloudFront にはサービスにリンクされたロールはありませんが、Lambda@Edge にはあります。詳細については、Amazon CloudFront デベロッパーガイドの「[Lambda@Edge のサービスにリンクされたロール](#)」を参照してください。

### AWS CloudTrail

CloudTrail は、[AWS の外部にあるイベントソースと CloudTrail Lake との統合](#)に使用される CloudTrail チャンネルでのみリソースベースのポリシーをサポートしています。

CloudTrail は、CloudTrail Lake イベントデータストアとチャンネルのタグをベースにしたアクセスコントロールをサポートしています。CloudTrail は、証跡のタグをベースにしたアクセスコントロールをサポートしていません。

### Amazon CloudWatch

CloudWatch のサービスにリンクされたロールは AWS Management Console を使用して作成することはできず、[アラームアクション](#)機能のみをサポートしています。

### AWS CodeBuild

CodeBuild は、AWS RAM を使用したアカウント間のリソース共有をサポートしています。

また、プロジェクトベースのアクションの ABAC もサポートしています。

### AWS Config

AWS Config は、マルチアカウントマルチリージョンのデータ集約、および AWS Config ルールについて、リソースレベルのアクセス許可をサポートしています。サポートされているリソースのリストについては、「[AWS Config API ガイド](#)」の「マルチアカウントマルチリージョンのデータ集約」セクション、および「AWS Config ルール」セクションを参照してください。

### AWS Database Migration Service

サポートされているターゲットエンドポイントに移行されるデータを暗号化するために作成する AWS KMS 暗号化キーにアタッチされるポリシーを作成および変更できます。サポートされて

いるターゲットエンドポイントには Amazon Redshift や Amazon S3 があります。詳細については、[AWS KMS ユーザーガイド](#)の「[Amazon Redshiftターゲットデータを暗号化するための AWS KMS キーの作成と使用](#)」および「[Amazon S3ターゲットオブジェクトを暗号化するための AWS Database Migration Service キーの作成](#)」を参照してください。

## Amazon Elastic Compute Cloud

Amazon EC2 サービスリンクロールは、[スポットインスタンスリクエスト](#)や[スポットフリートリクエスト](#)、[Amazon EC2 Fleet](#)、および [Windows インスタンス用の高速起動](#)の機能のみに使用できます。

## Amazon Elastic Container Service

[リソースレベルのアクセス許可をサポート](#)しているのは、一部の Amazon ECS アクションのみです。

## AWS Elemental MediaPackage

MediaPackage は、カスタマーアクセスログを CloudWatch に公開するためにサービスにリンクされたロールをサポートしていますが、他の API アクションではサポートしていません。

## AWS Identity and Access Management

IAM では、ロールの信頼ポリシーと呼ばれるリソースベースのポリシーのタイプを 1 つのみサポートします。これは、IAM ロールにアタッチされます。詳細については、「[ロールを切り替えるアクセス許可をユーザーに付与する](#)」を参照してください。

IAM は、ほとんどの IAM リソースでタグベースのアクセス制御をサポートしています。詳細については、「[IAM リソースのタグ付け](#)」を参照してください。

一時的な認証情報では、IAM の一部の API アクションのみ呼び出すことができます。詳細については、「[API オプションの比較](#)」を参照してください。

## AWS IoT

AWS IoT に接続されたデバイスは、X.509 証明書あるいは Amazon Cognito ID を使用して認証されます。X.509 証明書または Amazon Cognito ID に AWS IoT ポリシーをアタッチして、この操作が許可されるデバイスを管理できます。詳細については、[AWS IoT 開発者ガイド](#)の「[AWS IoT のセキュリティと ID](#)」を参照してください。

## AWS Lambda

Lambda は、必要なリソースとして Lambda 関数を使用する API アクションに対応した属性ベースのアクセス制御 (ABAC) をサポートしています。レイヤー、イベントソースマッピング、およびコード署名設定リソースはサポートされていません。

Lambda にはサービスにリンクされたロールはありませんが、Lambda@Edge にはあります。詳細については、「Amazon CloudFront デベロッパーガイド」の「[Lambda@Edge 用のサービスにリンクされたロール](#)」を参照してください。

## Amazon Lightsail

Lightsail は、リソースレベルのアクセス許可および ABAC を部分的にサポートしています。詳細については、「[Amazon Lightsail のアクション、リソース、および条件キー](#)」を参照してください。

## Amazon Managed Streaming for Apache Kafka (MSK)

[マルチ VPC 接続](#)用に設定された Amazon MSK クラスターにクラスターポリシーをアタッチできません。

## AWS Network Manager

AWS Cloud WAN は、サービスにリンクされたロールもサポートしています。詳細については、Amazon VPC AWS Cloud WAN ガイドの[AWS Cloud WAN のサービスにリンクされたロール](#)を参照してください。

## Amazon Relational Database Service

Amazon Aurora はフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。Amazon RDS を通じて新しいデータベースサーバーをセットアップする際には、DB エンジンのオプションで Aurora MySQL or Aurora PostgreSQL を選択できます。詳細については、「Amazon Aurora ユーザーガイド」の「[Amazon Aurora でのアイデンティティとアクセス管理](#)」を参照してください。

## Amazon Rekognition

リソースベースのポリシーは、Amazon Rekognition Custom Labels モデルのコピーに対してのみサポートされています。

## AWS Resource Groups

ユーザーは、Resource Groups オペレーションを許可するポリシーが関連付けられたロールを引き受けることができます。

## Amazon SageMaker

サービスにリンクされたロールは、現在 SageMaker Studio および SageMaker トレーニングジョブで使用できます。

## AWS Security Token Service

AWS STS には「リソース」はありませんが、ユーザーと同じようにアクセスを制限できます。詳細については、「[一時的セキュリティ認証情報のアクセスを名前で拒否する](#)」を参照してください。

一時的な認証情報を使用した呼び出しは、AWS STS の一部の API オペレーションでのみサポートされています。詳細については、「[API オプションの比較](#)」を参照してください。

## Amazon Simple Email Service

ses:SendEmail または ses:SendRawEmail など、Eメールの送信に関連するアクションを参照するポリシーステートメントでのみリソースレベルのアクセス許可を使用できます。他のアクションを参照するポリシーステートメントについては、Resource 要素は \* のみを含めることができます。

一時的なセキュリティ認証情報をサポートしているのは、Amazon SES API のみです。Amazon SES SMTP インターフェイスは、一時的なセキュリティ認証情報から派生した SMTP 認証情報をサポートしていません。

## Amazon Simple Storage Service

Amazon S3 は、オブジェクトリソースに対してのみタグベースの認証をサポートしています。

Amazon S3 では、Amazon S3 Storage Lens 用のサービスにリンクされたロールをサポートしています。

## AWS Trusted Advisor

Trusted Advisor への API アクセスは AWS Support API を介して行われ、AWS Support IAM ポリシーによって制御されます。



## Amazon Virtual Private Cloud

IAM ユーザーポリシーでは、特定の Amazon VPC エンドポイントへのアクセス許可を制限することはできません。Action または `ec2:*VpcEndpoint*` API アクションを含むいずれの `ec2:DescribePrefixLists` 要素にも、`"Resource": "*"` を指定する必要があります。詳細については、「AWS PrivateLink ガイド」の「[VPC エンドポイントおよび VPC エンドポイントサービスの Identity and Access Management](#)」を参照してください。

Amazon VPC では、VPC エンドポイントへの単一リソースポリシーのアタッチがサポートされており、そのエンドポイント経由でアクセスできるコンテンツを制限できます。リソースベースのポリシーを使用して、特定の Amazon VPC エンドポイントからリソースへのアクセスを管理する方法の詳細については、AWS PrivateLink ユーザーガイドの「[VPC エンドポイントでサービスへのアクセスを制御する](#)」を参照してください。

Amazon VPC にはサービスにリンクされたロールはありませんが、AWS Transit Gateway にはあります。詳細については、「Amazon VPC AWS Transit Gateway ガイド」の「[転送ゲートウェイのサービスにリンクされたロールの使用](#)」を参照してください。

## AWS X-Ray

X-Ray は、すべてのアクションにおけるリソースレベルのアクセス許可をサポートしていません。

X-Ray は、グループおよびサンプリングルールのタグベースのアクセス制御をサポートしています。

## AWS API リクエストの署名

### Important

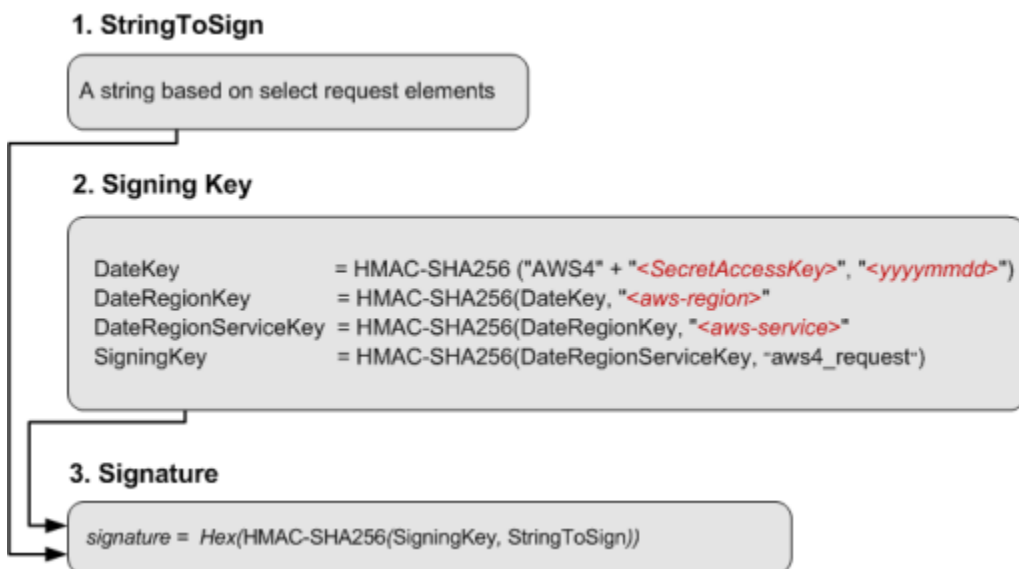
AWS SDK (「[サンプルコードとライブラリ](#)」を参照) または AWS コマンドライン (CLI) ツールを使用して API リクエストを AWS に送信する場合、SDK および CLI クライアントが指定したアクセスキーを使用してリクエストを認証するため、このセクションをスキップできます。正当な理由がない限り、常に SDK または CLI を使用することをお勧めします。複数の署名バージョンをサポートするリージョンでは、リクエストに手動で署名する場合、使用する署名バージョンを指定する必要があります。マルチリージョンアクセスポイントにリクエストを送信すると、SDK と CLI は Signature Version 4A の使用に自動的に切り替えます。追加の設定は不要です。

リクエストで送信する認証情報には、署名が含まれている必要があります。署名を計算するには、選択されているリクエストの要素を連結し、文字列を作成します。これは、署名する文字列と呼ばれます。次に、署名キーを使用して、署名する文字列の Hash-based Message Authentication Code (HMAC) を計算します。

AWS の署名バージョン 4 では、リクエストの署名にシークレットアクセスキーは使用しません。代わりに、まずシークレットアクセスキーを使用して署名キーを生成します。生成される署名キーは、日付、サービス、およびリージョンによって異なります。さまざまなプログラミング言語の署名キーを取得する方法の詳細については、「[リクエスト署名の例](#)」を参照してください。

署名バージョン 4 は、AWS 署名プロトコルです。AWS は、マルチリージョン API リクエストの署名をサポートする署名バージョン 4A という拡張機能もサポートしています。詳細については、GitHub で「[sigv4a-signing-examples](#)」プロジェクトを参照してください。

次の図は、署名を計算する一般的なプロセスを示しています。



- 署名する文字列はリクエストのタイプによって異なります。たとえば、認証に HTTP Authorization ヘッダーまたはクエリパラメータを使用する場合、リクエスト要素のさまざまな組み合わせを使用して、署名する文字列を作成します。HTTP POST リクエストでは、リクエスト内の POST ポリシーは、署名する文字列です。
- 署名キーについては、図に一連の計算が示されており、各手順の結果は次の手順に入力されます。最後の手順は署名キーです。
- AWS サービスが認証されたリクエストを受け取ると、リクエストに含まれる認証情報を使用して署名が再作成されます。署名が一致すると、サービスはリクエストを処理します。それ以外の場合、リクエストを拒否します。

## コンテンツ

- [リクエストに署名するタイミング](#)
- [リクエストに署名する理由](#)
- [AWS API リクエスト署名の要素](#)
- [認証方法](#)
- [署名付き AWS API リクエストを作成する](#)
- [リクエスト署名の例](#)
- [AWS API の署名済みリクエストのトラブルシューティング](#)

## リクエストに署名するタイミング

AWS に API リクエストを送信するためのカスタムコードを記述するときは、リクエストに署名するためのコードを含める必要があります。カスタムコードを書く理由には、以下のような場合が考えられます。

- AWS SDK がないプログラミング言語を使用しているためです。
- AWS にリクエストを送る方法を完全に管理する必要がある場合。

## リクエストに署名する理由

署名プロセスは、次のような点でリクエストのセキュリティ確保に役立ちます。

- リクエストの ID の確認

認証されたリクエストには、アクセスキー (アクセスキー ID、シークレットアクセスキー) を使用して作成した署名が必要です。一時的なセキュリティ認証情報を使用している場合、署名の計算にはセキュリティトークンも必要です。詳細については、「[AWS security credentials programmatic access](#)」(セキュリティ認証情報のプログラムによるアクセス) を参照してください。

- 送信中のデータの保護

送信中のリクエストの改ざんを防ぐために、一部のリクエスト要素からリクエストのハッシュ (ダイジェスト) を計算し、得られたハッシュ値をリクエストの一部として含めます。AWS のサービスがリクエストを受け取ると、同じ情報を使用してハッシュを計算し、リクエストに含まれているハッシュ値と比較します。ハッシュ値が一致しない場合、AWS はそのリクエストを拒否します。

- 潜在的なリプレイ攻撃の防止

多くの場合、リクエストは、リクエストのタイムスタンプの 5 分以内に AWS に到達する必要があります。その条件を満たさない場合、AWS はリクエストを拒否します。

## AWS API リクエスト署名の要素

### Important

AWS SDK または CLI を使用していない限り、リクエストの認証情報を提供する署名を計算するコードを記述する必要があります。AWS Signature Version 4 での署名計算は複雑な作業になる場合があるため、可能な限り AWS SDK または CLI を使用することをお勧めします。

Signature Version 4 の署名を使用する各 HTTP/HTTPS リクエストには、これらの要素を含める必要があります。

#### [Elements] (要素)

- [エンドポイント仕様](#)
- [アクション](#)
- [アクションパラメータ](#)
- [日付](#)
- [認証情報](#)

### エンドポイント仕様

リクエストの送信先となるエンドポイントの DNS 名を指定します。この名前には通常、サービスコードとリージョンが含まれます。たとえば、us-east-1 リージョンの Amazon DynamoDB のエンドポイントは `dynamodb.us-east-1.amazonaws.com` です。

HTTP/1.1 リクエストには、Host ヘッダーを含める必要があります。HTTP/2 リクエストの場合には、`:authority` ヘッダー、または Host ヘッダーを含められます。HTTP/2 仕様にはのみ準拠するには、`:authority` ヘッダーのみ使用します。すべてのサービスが HTTP/2 リクエストをサポートしているわけではありません。

各サービスでサポートされているエンドポイントについては、「AWS 全般のリファレンス」の「[サービスエンドポイントとクォータ](#)」を参照してください。

## アクション

サービスの API アクションを指定します。たとえば、DynamoDB CreateTable アクションや Amazon EC2 DescribeInstances アクションなどです。

各サービスでサポートされているアクションについては、「[サービス認証リファレンス](#)」を参照してください。

## アクションパラメータ

リクエストで指定されたアクションのパラメータを指定します。各 AWS API アクションには、必須およびオプションのパラメータのセットがあります。API バージョンは通常、必須パラメータです。

API アクションでサポートされるパラメータについては、サービスの「[API リファレンス](#)」を参照してください。

## 日付

リクエストの日付と時刻を指定します。日付と時刻をリクエストに含めると、サードパーティーによるリクエストの傍受や再送信の防止に役立ちます。認証情報スコープで指定した日付は、リクエストの日付と一致する必要があります。

タイムスタンプは UTC で、YYYYMMDDTHHMMSSZ の ISO 8601 形式を使用する必要があります。例えば、20220830T123600Z と指定します。タイムスタンプにミリ秒を含めないでください。

date または x-amz-date ヘッダーを使用するか、x-amz-date をクエリパラメータとして含めることができます。x-amz-date ヘッダーが見つからない場合は、date ヘッダーを探します。

## 認証情報

送信する各リクエストには、次の情報を含める必要があります。AWS はこの情報を使用してリクエストの有効性と信頼性を確保しています。

- アルゴリズム — HMAC-SHA256 ハッシュアルゴリズムで Signature Version 4 を指定する場合に AWS4-HMAC-SHA256 を使用します。
- 認証情報 — アクセスキー ID、YYYYMMDD 形式の日付、リージョンコード、サービスコード、およびスラッシュ (/) で区切った aws4\_request 終了文字列から成る文字列。リージョンコード、サービスコード、および終了文字列には、小文字を使用する必要があります。

```
AKIAIOSFODNN7EXAMPLE/YYYYMMDD/region/service/aws4_request
```

- 署名付きヘッダー — 署名に含める HTTP ヘッダーを、セミコロン (;) で区切ります。例えば、host;x-amz-date と指定します。
- 署名 — 計算された署名を表す 16 進エンコードされた文字列。Algorithm パラメータで指定したアルゴリズムを使用して署名を計算する必要があります。

## 認証方法

### Important

AWS SDK または CLI を使用していない限り、リクエストの認証情報を提供する署名を計算するコードを記述する必要があります。AWS Signature Version 4 での署名計算は複雑な作業になる場合があるため、可能な限り AWS SDK または CLI を使用することをお勧めします。

認証情報は、次のいずれかの方法で表現できます。

### HTTP 認証ヘッダー

HTTP Authorization ヘッダーはリクエストを認証する最も一般的な方法です。すべての REST API 操作 (POST リクエストを使用したブラウザベースのアップロードを除く) には、このヘッダーが必要です。認証ヘッダー値、署名の計算方法、および関連オプションの詳細については、「Amazon S3 API リファレンス」の「[リクエストの認証: 認証ヘッダーの使用 \(AWS Signature Version 4\)](#)」を参照してください。

以下は、Authorization ヘッダー値の例です。この例では、読みやすいように改行が追加されています。コードでは、ヘッダーは連続した文字列である必要があります。アルゴリズムと認証情報の間にカンマはありませんが、他の要素はカンマで区切る必要があります。

```
Authorization: AWS4-HMAC-SHA256  
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,  
SignedHeaders=host;range;x-amz-date,  
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

次の表では、前述の例にある認証ヘッダー値のさまざまなコンポーネントについて説明しています。

コンポーネント	説明
認証	<p>署名の計算に使用されたアルゴリズム。認証に AWS Signature Version 4 を使用する場合は、この値を指定する必要があります。文字列は AWS Signature Version 4 (AWS4) と署名アルゴリズム (HMAC-SHA256) を指定します。</p>
Credential	<p>アクセスキー ID と、署名の計算に使用された日付、リージョン、およびサービスを含むスコープ情報。</p> <p>この文字列の形式は次のとおりです。</p> <pre>&lt;your-access-key-id&gt;/&lt;date&gt;/ &lt;aws-region&gt;/&lt;aws-service&gt;/ aws4_request</pre> <p>Where: &lt;date&gt; 値は YYYYMMDD 形式で指定されます。Amazon S3 にリクエストを送信するときの &lt;aws-service&gt; 値は s3 です。</p>
SignedHeaders	<p>Signature の計算に使用したリクエストヘッダーをセミコロンで区切ったリスト。リストにはヘッダー名のみが含まれており、ヘッダー名は小文字である必要があります。例:</p> <pre>host;range;x-amz-date</pre>
署名	<p>64 個の小文字の 16 進数文字で表現される 256 ビットの署名。以下に例を示します。</p> <pre>fe5f80f77d5fa3beca038a248fff 027d0445342fe2855ddc9631766 30326f1024</pre> <p>署名計算は、ペイロードの転送に選択したオプションによって異なります。</p>



## クエリ文字列パラメータ

クエリ文字列を使用すると、リクエスト全体を URL で表現できます。この場合、クエリパラメータを使用して、認証情報を含むリクエスト情報を提供します。リクエスト署名は URL の一部であるため、この種類の URL は署名付き URL と呼ばれることがよくあります。署名付き URL を使用して、クリック可能なリンクを HTML に埋め込むことができます。このリンクは最大 7 日間有効です。詳細については、「Amazon S3 API リファレンス」の「[リクエストの認証: クエリパラメータの使用 \(AWS Signature Version 4\)](#)」を参照してください。

署名付き URL の例を次に示します。この例では、読みやすいように改行が追加されています。

```
https://s3.amazonaws.com/examplebucket/test.txt ?
X-Amz-Algorithm=AWS4-HMAC-SHA256 &
X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request &
X-Amz-Date=20130721T201207Z &
X-Amz-Expires=86400 &
X-Amz-SignedHeaders=host &X-Amz-Signature=<signature-value>
```

### Note

URL 内の X-Amz-Credential 値には、読みやすさのためにのみ挿入された「/」文字が表示されています。実際には、%2F としてエンコードする必要があります。例:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

次の表では、認証情報を提供する URL 内のクエリパラメータについて説明します。

クエリ文字列パラメータ名	説明
X-Amz-Algorithm	AWS Signature のバージョンと、署名の計算に使用したアルゴリズムを識別します。AWS Signature Version 4 では、このパラメータ値を AWS4-HMAC-SHA256 に設定します。この文字列は、AWS Signature Version 4 (AWS4) および HMAC-SHA256 アルゴリズム (HMAC-SHA256 ) を識別します。



クエリ文字列パラメータ名	説明
X-Amz-Credential	<p>このパラメータは、アクセスキー ID の他に、署名が有効なスコープ (AWS リージョンおよびサービス) も指定します。この値は、次のセクションで説明する署名計算で使用するスコープと一致する必要があります。</p> <p>このパラメータ値の一般的な形式は次のとおりです。</p> <pre>&lt;your-access-key-id&gt;/&lt;date&gt;/ &lt;AWS Region&gt;/&lt;AWS-service&gt;/aws4_ request</pre> <p>例: AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</p> <p>AWS リージョン文字列のリストについては、「AWS 全般のリファレンス」の「<a href="#">リージョンのエンドポイント</a>」を参照してください。</p>
X-Amz-Date	<p>日付と時刻の形式は ISO 8601 規格に準拠している必要があるため、yyyyMMddTHHmssZ 形式でフォーマットする必要があります。例えば、日付と時刻が「08/01/2016 15:32:41.982-700」の場合、まず UTC (協定世界時) に変換してから「20160801T223241Z」として送信する必要があります。</p>
X-Amz-Expires	<p>生成された署名付き URL が有効な期間を秒単位で指定します。例えば、86400 (24 時間) と指定します。この値は整数です。設定できる最小値は 1 で、最大値は 604800 (7 日間) です。署名計算に使用する署名キーは最大 7 日間有効であるため、署名付き URL は最大 7 日間有効です。</p>

クエリ文字列パラメータ名	説明
X-Amz-SignedHeaders	<p>署名の計算に使用したヘッダーを一覧表示します。署名計算には次のヘッダーが必要です。</p> <ul style="list-style-type: none"><li>• HTTP ホストヘッダー。</li><li>• リクエストに追加する予定のすべての x-amz-* ヘッダー。</li></ul> <p>セキュリティの強化のために、リクエストに含める予定のすべてのリクエストヘッダーに署名する必要があります。</p>
X-Amz-Signature	<p>リクエストを認証するための署名を指定します。この署名は、サービスが計算する署名と一致する必要があります。一致しない場合、サービスはリクエストを拒否します。例えば、733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7</p> <p>署名計算については、次のセクションで説明します。</p>
X-Amz-Security-Token	<p>STS サービスから取得した認証情報を使用する場合のオプションの認証情報パラメータ。</p>

## 署名付き AWS API リクエストを作成する

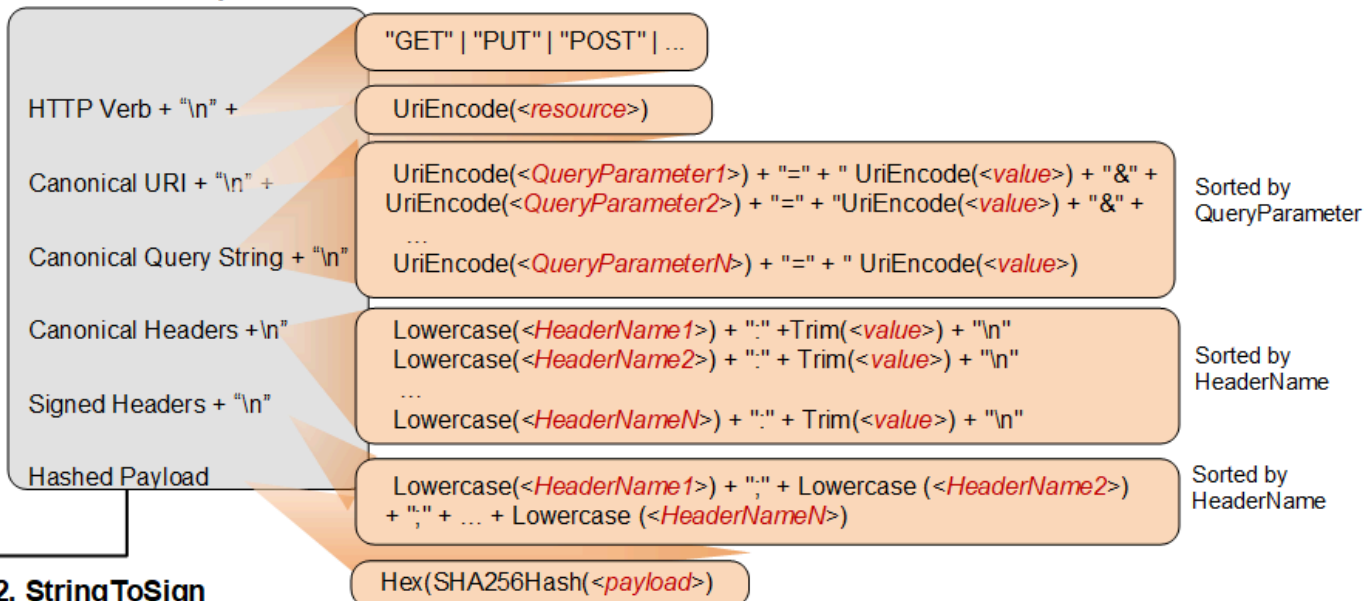
### Important

AWS SDK (「[サンプルコードとライブラリ](#)」を参照) または AWS コマンドライン (CLI) ツールを使用して API リクエストを AWS に送信する場合、SDK および CLI クライアントが指定したアクセスキーを使用してリクエストを認証するため、このセクションをスキップできません。正当な理由がない限り、常に SDK または CLI を使用することをお勧めします。複数の署名バージョンをサポートするリージョンでは、リクエストに手動で署名する場合、使用する署名バージョンを指定する必要があります。マルチリージョンアクセスポイントに

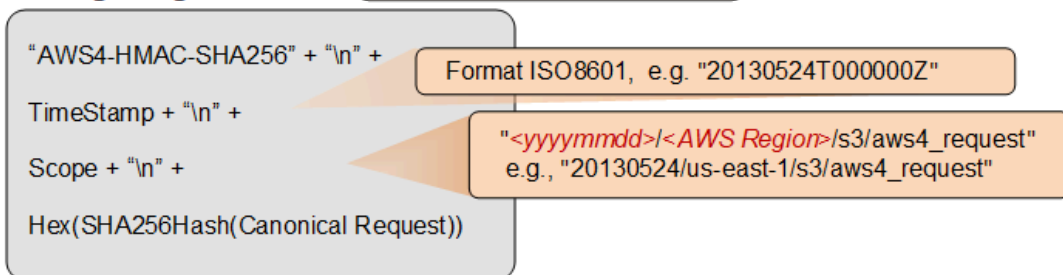
リクエストを送信すると、SDK と CLI は Signature Version 4A の使用に自動的に切り替えます。追加の設定は不要です。

以下は、署名付きリクエストを作成するプロセスの概要です。署名を計算するには、まず署名する文字列が必要です。次に、署名キーを使用して署名する文字列の HMAC-SHA256 ハッシュを計算します。次の図は、署名用に作成する文字列のさまざまなコンポーネントを含むプロセスを示しています。

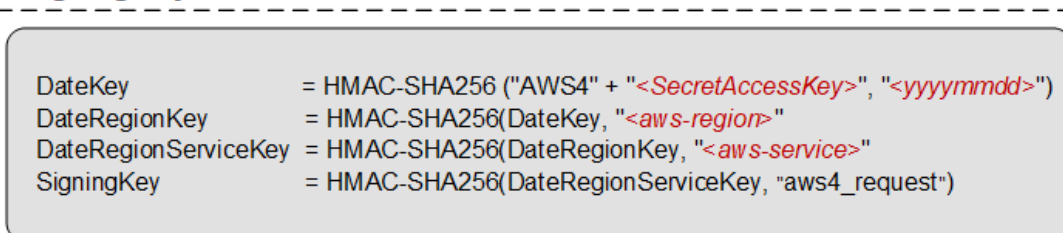
### 1. Canonical Request



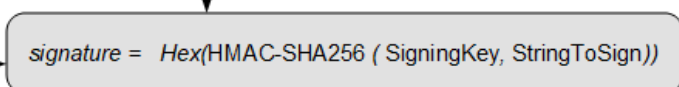
### 2. StringToSign



### 3. Signing Key



### 4. Signature



次の表では、図に示されている関数について説明します。これらの関数のコードを実装する必要があります。詳細については、「[AWS SDK のサンプルコード](#)」を参照してください。

機能	説明
Lowercase()	文字列を小文字に変換します。
Hex()	16 進数の小文字エンコード。
SHA256Hash()	セキュアハッシュアルゴリズム (SHA) 暗号化ハッシュ関数。
HMAC-SHA256()	指定した署名キーで SHA256 アルゴリズムを使用して HMAC を計算します。これが最後の署名です。
Trim()	先頭または末尾の空白をすべて削除します。
UriEncode()	<p>すべてのバイトが URI でエンコードされます。UriEncode() は、以下のルールを適用する必要があります。</p> <ul style="list-style-type: none"><li>• 非予約文字 (「A」 - 「Z」、 「a」 - 「z」、 「0」 - 「9」、 「-」、 「.」、 「_」 および 「~」) を除くすべてのバイトが URI でエンコードされます。</li><li>• スペース文字は予約文字であるため、「+」ではなく「%20」としてエンコードする必要があります。</li><li>• URI でエンコードされた各バイトは、「%」とそのバイトの 2 桁の 16 進値で構成されています。</li><li>• 16 進値の文字は大文字である必要があります (例: 「%1A」)。</li><li>• オブジェクトキー名以外では、フォワードスラッシュ文字「/」がエンコードされます。例えば、オブジェクトキー名が photos/Jan/sample.jpg の場合、キー名のフォワードスラッシュはエンコードされません。</li></ul>

機能	説明
	<p><b>⚠ Important</b></p> <p>開発プラットフォームが提供する標準の UriEncode 関数は、実装の違いや基礎となる RFC のあいまいさにより動作しない場合があります。エンコードが確実に機能するように、独自のカスタム UriEncode 関数を作成することをお勧めします。</p> <p>Java での UriEncode 関数の例については、GitHub ウェブサイトで「<a href="#">Java Utilities</a>」を参照してください。</p>

**i Note**

リクエストに署名するときは、AWS Signature Version 4 または AWS Signature Version 4A のいずれかを使用できます。これら 2 つの大きな違いを決定づけるのは、署名の計算方法です。AWS Signature Version 4A では、署名にはリージョン固有の情報は含まれず、署名の計算には AWS4-ECDSA-P256-SHA256 アルゴリズムが使用されます。

## 一時的な認証情報

リクエストに署名するために長期的に認証情報を使用する代わりに、AWS Security Token Service (AWS STS) から提供された一時的なセキュリティ認証情報を使用できます。

一時的なセキュリティ認証情報を使用する場合は、認証ヘッダーまたはクエリ文字列にセッショントークンを保持するように X-Amz-Security-Token を追加する必要があります。一部のサービスでは、正規リクエストに X-Amz-Security-Token を追加する必要があります。その他のサービスでは、署名の計算後に、X-Amz-Security-Token を末尾に追加するだけです。詳細については、それぞれ AWS のサービスのドキュメントを確認してください。

## 署名手順の概要

### 手順 1: 正規リクエストを作成する

リクエストのコンテンツ (ホスト、アクション、ヘッダーなど) を標準的な正規形式に変換します。正規リクエストは、署名する文字列を作成するのに使用される入力の 1 つです。詳細については、「[AWS API リクエスト署名の要素](#)」を参照してください。

### 手順 2: 正規リクエストのハッシュを作成する

署名キーは、最初のハッシュオペレーションのキーとして AWS のシークレットアクセスキーを使用して、リクエスト日、リージョン、およびサービスに対する一連のキー付きハッシュオペレーション (HMAC オペレーション) を実行することによって抽出します。

### 手順 2: 署名文字列を作成する

正規リクエストに加えてアルゴリズム、リクエスト日、認証情報スコープ、正規リクエストのダイジェスト (ハッシュ) などの追加情報を使用して、署名する文字列を作成します。

### 手順 4: 署名を計算する

署名キーを取得したら、署名文字列にキー付きハッシュ操作を実行することで、署名を計算します。取得した署名キーを、この操作のハッシュキーとして使用します。

### 手順 5: リクエストヘッダーに署名を追加します。

署名を計算したら、それをリクエストの HTTP ヘッダーまたはクエリ文字列に追加します。

### 手順 1: 正規リクエストを作成する

次の文字列を改行文字で区切って連結し、正規リクエストを作成します。これにより、計算する署名と AWS が計算する署名が一致することが保証されます。

```
<HTTPMethod>\n<CanonicalURI>\n<CanonicalQueryString>\n<CanonicalHeaders>\n<SignedHeaders>\n<HashedPayload>
```

- **HTTPMethod** – GET、PUT、HEAD、DELETE などの HTTP メソッド。

- **CanonicalUri** – 絶対パスコンポーネント URI の URI エンコード版 (ドメイン名の後の「/」から始まり、文字列の末尾まで、またはクエリ文字列パラメータがある場合は疑問符文字 (「?」) まで)。絶対パスが空値の場合は、フォワードスラッシュ文字 (/) を使用します。次の例の URI、/examplebucket/myphoto.jpg は絶対パスであるため、ユーザーは絶対パス内の「/」をエンコードしないでください。

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

- **CanonicalQueryString** – URI エンコードされたクエリ文字列パラメータ。それぞれの名前と値を個別に URI エンコードします。また、正規クエリ文字列内のパラメータをキー名のアルファベット順にソートする必要があります。ソートはエンコード後に行われます。次のサンプル URI のクエリ文字列は次のとおりです。

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=2
```

正規クエリ文字列は次のとおりです (この例では、読みやすいように改行が追加されています)。

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

リクエストがサブリソースをターゲットにしている場合、対応するクエリパラメータ値は空の文字列 ("" ) になります。例えば、次の URI は examplebucket バケット上の ACL サブリソースを特定してします。

```
http://s3.amazonaws.com/examplebucket?acl
```

この場合の CanonicalQueryString は次のとおりです。

```
UriEncode("acl") + "=" + ""
```

URI に「?」が含まれていない場合は、リクエストにクエリ文字列が存在しないため、正規クエリ文字列を空の文字列 ("" ) に設定します。「\n」は引き続き含める必要があります。

- **CanonicalHeaders** – リクエストヘッダーとその値のリスト。個々のヘッダー名と値のペアは改行文字 (「\n」) で区切られます。以下は正規ヘッダーの例です。



```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

CanonicalHeaders リストには以下が含まれている必要があります。

- HTTP host ヘッダー。
- Content-Type ヘッダーがリクエスト内に存在する場合は、それを *CanonicalHeaders* リストに追加する必要があります。
- リクエストに含める予定の x-amz-\* ヘッダーも追加する必要があります。例えば、一時的なセキュリティ認証情報を使用している場合は、リクエストに x-amz-security-token を含める必要があります。このヘッダーを *CanonicalHeaders* リストに追加する必要があります。

#### Note

x-amz-content-sha256 ヘッダーは Amazon S3 AWS リクエストに必要です。これにより、リクエストペイロードのハッシュが指定されます。ペイロードがない場合は、空の文字列のハッシュを指定する必要があります。

各ヘッダー名は次の条件を満たす必要があります。

- 小文字が使用されていること。
- アルファベット順に表示されていること。
- 後にコロン (:) が指定されていること。

値については、次の操作を行う必要があります。

- 先頭または末尾のスペースをすべて削除する。
- 連続するスペースを1つのスペースに変換する。
- 複数值ヘッダーの値をカンマで区切る。
- host ヘッダー (HTTP/1.1) または :authority ヘッダー (HTTP/2)、および任意の x-amz-\* ヘッダーを署名に含める必要があります。署名には、content-type のような他の標準ヘッダーを指定することもできます。

この例で使用されている関数 Lowercase() と Trim() は、前のセクションで説明されています。

以下は CanonicalHeaders 文字列の例です。ヘッダー名は小文字で、ソートされています。

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

#### Note

認証署名を計算する目的では、host ヘッダーと任意の x-amz-\* ヘッダーだけが必要です。ただし、データの改ざんを防ぐために、すべてのヘッダーを署名計算に含めることを検討する必要があります。

- **SignedHeaders** – アルファベット順にソートされ、セミコロンで区切られた小文字のリクエストヘッダー名のリスト。リスト内のリクエストヘッダーは、CanonicalHeaders 文字列に含めたヘッダーと同じです。例えば、前の例の場合、**SignedHeaders** の値は次のようになります。

```
host;x-amz-content-sha256;x-amz-date
```

- **HashedPayload** — HTTP リクエスト本文のペイロードをハッシュ関数への入力として使用して作成された文字列。この文字列には小文字の 16 進数文字を使用します。

```
Hex(SHA256Hash(<payload>))
```

リクエストにペイロードがない場合は、空の文字列のハッシュを次のように計算します。

```
Hex(SHA256Hash(""))
```

ハッシュは次の値を返します。

```
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

例えば、PUT リクエストを使用してオブジェクトをアップロードする場合、本文にオブジェクトデータを指定します。GET リクエストを使用してオブジェクトを取得する場合、空の文字列ハッシュを計算します。

## 手順 2: 正規リクエストのハッシュを作成する

ペイロードのハッシュを作成する際に使用したのと同じアルゴリズムを使用して、正規リクエストのハッシュ (ダイジェスト) を作成します。正規リクエストのハッシュは、小文字の 16 進数文字の文字列として表す必要があります。

## 手順 3: 署名文字列を作成する

次の文字列を改行文字で区切って連結して、文字列を作成します。この文字列は改行文字で終わらせないようにしてください。

```
Algorithm \n
RequestDateTime \n
CredentialScope \n
HashedCanonicalRequest
```

- **Algorithm** — 正規リクエストのハッシュを作成するために使用されるアルゴリズム。SHA-256 では、アルゴリズムは AWS4-HMAC-SHA256 です。
- **RequestDateTime** — 認証情報のスコープで使用される日付と時刻。この値は、ISO 8601 形式の現在の UTC 時刻です (例: 20130524T000000Z)。
- **CredentialScope** — 認証情報のスコープ。これにより、生成される署名は指定されたリージョンとサービスに制限されます。文字列の形式は **YYYYMMDD/region/service/aws4\_request** です。
- **HashedCanonicalRequest** — 正規リクエストのハッシュ。この値はステップ 2 で計算されます。

以下は署名する文字列の例です。

```
"AWS4-HMAC-SHA256" + "\n" +
timestampISO8601Format + "\n" +
<Scope> + "\n" +
Hex(SHA256Hash(<CanonicalRequest>))
```

## 手順 4: 署名を計算する

AWS Signature Version 4 では、リクエストの署名に AWS アクセスキーを使用するのではなく、リクエストに追加する認証情報として、特定のリージョンとサービスにスコープされた署名キーを作成します。

```
DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

リージョン文字列のリストについては、「AWS 全般のリファレンス」の「[リージョンのエンドポイント](#)」を参照してください。

各ステップで、必要なキーとデータを使用してハッシュ関数を呼び出します。ハッシュ関数を呼び出すたびに、その結果はハッシュ関数への次の呼び出しの入力になります。

#### 入力必須

- シークレットアクセスキーを含む文字列 Key
- 認証情報の範囲で使用される日付を含んだ YYYYMMDD 形式の文字列 Date
- リージョンコードを含む文字列 Region (例: us-east-1)
- サービスコードを含む文字列 Service (例: ec2)
- 上記の手順で作成したトピックを選択します。

#### 署名を計算するには

1. 「AWS4」とシークレットアクセスキーを連結します。データとしてのキーおよび日付文字列として、連結した文字列を備えたハッシュ関数を呼び出します。

```
kDate = hash("AWS4" + Key, Date)
```

2. データとしてのキーおよびリージョン文字列として、前回の結果を備えたハッシュ関数を呼び出します。

```
kRegion = hash(kDate, Region)
```

3. データとしてのキーおよびサービス文字列として、前回の結果を備えたハッシュ関数を呼び出します。

```
kService = hash(kRegion, Service)
```

4. データとしてのキーおよび「aws4\_request」として、前回の結果を備えたハッシュ関数を呼び出します

```
kSigning = hash(kService, "aws4_request")
```

5. データとしての署名用キーおよび文字列として、前回の結果を備えたハッシュ関数を呼び出します。その結果、署名はバイナリ値になります。

```
signature = hash(kSigning, string-to-sign)
```

6. 署名を 2 進数から 16 進数表現に、小文字に変換します。

手順 5: リクエストヘッダーに署名を追加します。

Example 例: 認証ヘッダー

以下の例は、DescribeInstances アクションに対する Authorization ヘッダーを示しています。読みやすいように、この例では改行でフォーマットされています。コードでは、これは連続した文字列である必要があります。アルゴリズムと Credential の間にカンマはありません。ただし、他の要素はカンマで区切られている必要があります。

```
Authorization: AWS4-HMAC-SHA256  
Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request,  
SignedHeaders=host;x-amz-date,  
Signature=calculated-signature
```

Example 例: クエリ文字列に認証パラメータを含むリクエスト

次の例は、認証情報を含む DescribeInstances アクションのクエリを示しています。読みやすいように、この例は改行でフォーマットされており、URL はエンコードされていません。コードでは、クエリ文字列は URL エンコードされた連続した文字列である必要があります。

```
https://ec2.amazonaws.com/?  
Action=DescribeInstances&  
Version=2016-11-15&  
X-Amz-Algorithm=AWS4-HMAC-SHA256&  
X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request&  
X-Amz-Date=20220830T123600Z&  
X-Amz-SignedHeaders=host;x-amz-date&  
X-Amz-Signature=calculated-signature
```

## AWS SDK 内のソースコード

AWS SDK には、AWS API リクエストに署名する方法を示す GitHub のソースコードが含まれています。コードの例については、「[AWS サンプルリポジトリ内のプロジェクト例](#)」を参照してください。

- AWS SDK for .NET – [AWS4Signer.cs](#)
- AWS SDK for C++ – [AWSAuthV4Signer.cpp](#)
- AWS SDK for Go – [v4.go](#)
- AWS SDK for Java – [BaseAws4Signer.java](#)
- AWS SDK for JavaScript – [v4.js](#)
- AWS SDK for PHP – [SignatureV4.php](#)
- AWS SDK for Python (Boto) – [signers.py](#)
- AWS SDK for Ruby – [signer.rb](#)

## リクエスト署名の例

以下の AWS 署名リクエストの例は、AWS SDK または AWS コマンドラインツールを使わずに送信されたリクエストに対して SigV4 を使用して署名する方法を示しています。

### HTTP POST を使用したブラウザベースの Amazon S3 アップロード

「[リクエストの認証: ブラウザベースのアップロード](#)」では、Amazon S3 がリクエストを受け取るときに署名を計算するために使用する署名と関連情報について説明します。

「[例: HTTP POST を使用したブラウザベースのアップロード \(AWS Signature Version 4 を使用\)](#)」では、サンプルの POST ポリシーおよびファイルのアップロードに使用できるフォームを含む詳細情報が提供されています。ポリシーと架空の認証情報の例は、ワークフローおよびその結果として生成される署名とポリシーハッシュを示しています。

### VPC Lattice で認証されたリクエスト

「[Signature Version 4 \(SigV4\) で認証されたリクエストの例](#)」では、Python と Java の例が紹介されており、カスタムインターセプターを使用する場合と使用しない場合の両方でリクエスト署名を実行する方法が示されています。

## Amazon Translate での署名バージョン 4 の使用

「[Amazon Translate での Signature Version 4 の使用](#)」では、Python プログラムを使用して Amazon Translate リクエストに認証情報を追加する方法を示しています。この例では、POST リクエストを作成し、翻訳するテキストがリクエストの本文 (ペイロード) に含まれている JSON 構造を作成して、Authorization ヘッダーで認証情報を渡します。

## Neptune での Signature Version 4 の使用

「[例: Signature Version 4 署名付き Python を使用して Neptune に接続する](#)」では、Python を使用して Neptune に署名付きリクエストを行う方法を示しています。この例には、アクセスキーや一時的な認証情報を使用するためのバリエーションが含まれています。

## S3 Glacier への HTTP リクエストへの署名

[ストリーミング API の署名計算例](#) では、S3 Glacier の 2 つのストリーミング API の 1 つであるアップロードアーカイブ (POST アーカイブ) の署名を作成する方法について詳しく説明します。

## Amazon SWF に対する HTTP リクエストの実行

[Amazon SWF への HTTP リクエストの作成](#) では、Amazon SWF への JSON リクエストのヘッダーの内容を示します。

## Amazon OpenSearch Service におけるストリーミング API のシグネチャ計算

「[AWS SDK または PHP バージョン 3 で Amazon OpenSearch Service に署名する](#)」では、署名された HTTP リクエストを Amazon OpenSearch サービスに送信する方法の例を示します。

## AWS サンプルリポジトリ内のプロジェクト例

次のプロジェクト例では、Python、Node.js、Java、C#、Go、Rust などの一般的な言語を使用してリクエストに署名し、AWS サービスに Rest API リクエストを作成する方法を示します。

### 署名バージョン 4a のプロジェクト

[sigv4-signing-examples](#) プロジェクトでは、Python、Node.js、Java、C#、Go、Rust などの一般的な言語を使用して、SigV4A でリクエストに署名して AWS のサービスへの Rest API リクエストを作成する方法の例を示します。

[sigv4a-signing-examples](#) プロジェクトでは、マルチリージョン API リクエストに署名するための例を提供しています。例えば、[Amazon S3 でのマルチリージョンアクセスポイント](#)などです。

## Publish to AWS IoT Core

「[Python code to publish to AWS IoT Core using HTTPs protocol](#)」では、HTTPS プロトコルと AWS SigV4 認証を使用してメッセージを AWS IoT Core にパブリッシュする方法を説明しています。これには 2 つのリファレンス実装があります。1 つは Python で、もう 1 つは NodeJS です。

「[.Net Framework application to publish to AWS IoT Core using HTTPs protocol](#)」では、HTTPS プロトコルと AWS SigV4 認証を使用してメッセージを AWS IoT Core にパブリッシュする方法を説明しています。このプロジェクトには、.NET Core と同等の実装も含まれています。

## AWS API の署名済みリクエストのトラブルシューティング

### Important

AWS SDK または CLI を使用していない限り、リクエストの認証情報を提供する署名を計算するコードを記述する必要があります。SigV4 での署名計算は複雑な作業になる場合があるため、可能な限り AWS SDK または CLI を使用することをお勧めします。

署名リクエストを作成するコードを開発する際に、AWS のサービスが HTTP 403 SignatureDoesNotMatch エラーを表示する場合があります。これらのエラーは、AWS に対する HTTP リクエストの署名値が、AWS のサービスが計算した署名と一致しなかったことを意味します。HTTP 401 Unauthorized エラーは、アクセス許可によって呼び出し元がリクエストを行うことを許可されていない場合に返されます。

API リクエストは次の場合にエラーを返す可能性があります。

- API リクエストは署名されておらず、API リクエストは IAM 認証を使用している場合。
- リクエストの署名に使用された IAM 認証情報が正しくないか、API を呼び出す権限がない場合。
- 署名された API リクエストの署名が、AWS サービスが計算した署名と一致していない場合。
- API リクエストヘッダーが正しくない場合。

### Note

他のエラー解決策を検討する前に、AWS Signature Version 2 (SigV2) から AWS Signature Version 4 (SigV4) に更新してください。Amazon S3 などのサービスやリージョンは SigV2 署名をサポートしなくなりました。



## 考えられる原因

- [認証情報エラー](#)
- [正規リクエストと署名文字列エラー](#)
- [認証情報範囲エラー](#)
- [キー署名エラー](#)

## 認証情報エラー

API リクエストが SigV4 で署名されていることを確認します。API リクエストが署名されていない場合、Missing Authentication Token のエラーが表示されることがあります。[足りない署名を追加して](#)、リクエストを再送信してください。

アクセスキーとシークレットキーの認証情報が正しいかどうか確認してください。アクセスキーが正しくない場合、Unauthorized のエラーが表示されることがあります。リクエストに署名したエンティティがリクエストを行う権限を持っていることを確認してください。詳細については、「[アクセス拒否エラーメッセージのトラブルシューティング](#)」を参照してください。

## 正規リクエストと署名文字列エラー

[手順 2: 正規リクエストのハッシュを作成する](#) または [手順 3: 署名文字列を作成する](#) での正規化リクエストまたは署名する文字列を誤って計算した場合、サービスによって実行される署名の検証手順が失敗し、エラーメッセージが表示されます。

```
The request signature we calculated does not match the signature you provided
```

AWS サービスは署名されたリクエストを受け取ると、署名を再計算します。値に差があると、署名の一致に失敗します。正規リクエストと文字列を、署名付きリクエストとエラーメッセージの値と比較します。相違点がある場合は、署名プロセスを変更してください。

### Note

また、ヘッダーやリクエストを変更するプロキシ経由でリクエストを送信していないことを確認することもできます。

## Example 正規リクエストの例

```
GET
```

```
----- HTTP method
```

```

/ ----- Path. For API stage
  endpoint, it should be /{stage-name}/{resource-path}

----- Query string key-
value pair. Leave it blank if the request doesn't have a query string.
content-type:application/json ----- Header key-value
  pair. One header per line.
host:0123456789.execute-api.us-east-1.amazonaws.com ----- Host and x-amz-date
  are required headers for all signed requests.
x-amz-date:20220806T024003Z

content-type;host;x-amz-date ----- A list of signed
  headers
d167e99c53f15b0c105101d468ae35a3dc9187839ca081095e340f3649a04501 ----- Hash
  of the payload

```

シークレットキーがアクセス キー ID と一致することを確認するには、既知の動作する実装でテストします。例えば、AWS SDK または AWS CLI を使用して AWS へのリクエストを行います。

## API リクエストヘッダー

[手順 4: 署名を計算する](#) に追加した SigV4 認証ヘッダーに、次のような正しい認証情報キーが含まれていることを確認してください。

```

Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=example-generated-signature

```

認証情報キーがないか、正しくないと、Authorization header requires 'Credential' parameter. Authorization header requires 'Signature' parameter. のエラーが表示されることがあります。SigV4 認証要求には、HTTP Date または x-amz-date ヘッダーのいずれかを使用してリクエスト日も記載してください。

## 認証情報範囲エラー

[手順 3: 署名文字列を作成する](#) で作成された認証情報の範囲により、署名は特定の日付、リージョン、およびサービスに制限されます。この文字列は以下の形式になります。

```
YYYYMMDD/region/service/aws4_request
```

**Note**

SigV4a を使用している場合、リージョンは認証情報のスコープに含まれません。

## 日付

認証情報の範囲で x-amz-date ヘッダーと同じ日付が指定されていない場合、署名の検証手順が失敗し、次のエラーメッセージが表示されます。

```
Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from HTTP
```

リクエストで将来の時刻が指定されている場合、署名の検証ステップは次のエラーメッセージで失敗します。

```
Signature not yet current: date is still later than date
```

リクエストの有効期限が切れた場合、署名検証ステップは次のエラーメッセージで失敗します。

```
Signature expired: date is now earlier than date
```

## リージョン

認証情報の範囲でリクエストと同じリージョンが指定されていない場合、署名の検証ステップは次のエラーメッセージで失敗します。

```
Credential should be scoped to a valid Region, not region-code
```

## サービス

認証情報の範囲で host ヘッダーと同じサービスが指定されていない場合、署名の検証ステップは次のエラーメッセージで失敗します。

```
Credential should be scoped to correct service: 'service'
```

## 終了文字列

認証情報の範囲が `aws4_request` で終わっていない場合、署名の検証ステップは次のエラーメッセージで失敗します。

```
Credential should be scoped with a valid terminator: 'aws4_request'
```

## キー署名エラー

署名キーの不正な取得や暗号の不適切な使用に起因するエラーは、トラブルシューティングがさらに困難です。正規文字列と署名文字列が正しいことが検証されたら、次の問題のいずれかを確認することもできます。

- シークレットアクセスキーが、指定したアクセスキー ID と一致しない。
- キー取得コードに問題がある。

シークレットキーがアクセス キー ID と一致することを確認するには、既知の動作する実装でテストします。例えば、AWS SDK または AWS CLI を使用して AWS へのリクエストを行います。例については、「[リクエスト署名の例](#)」を参照してください。

## IAM JSON ポリシーリファレンス

このセクションは、IAM の JSON ポリシーの要素、変数、および評価ロジックの詳細な構文、説明、および例を示します。一般的な的な情報については、「[JSON ポリシー概要](#)」を参照してください

このリファレンスには、次のセクションがあります。

- [IAM JSON ポリシー要素のリファレンス](#) - ポリシーを作成する際に使用できる要素についての詳細は、こちらをご覧ください。さらにポリシーの例をご紹介しますほか、条件、サポートされているデータタイプ、および様々なサービスにおけるそれらの使用方法について説明します。
- [ポリシーの評価論理](#) - このセクションは、AWS リクエストの概要、リクエストがどのように認証されるか、および AWS がポリシーをどのように利用してリソースへのアクセス権限付与を決定するかについて説明します。
- [IAM JSON ポリシー言語の文法](#) - このセクションでは、IAM でポリシーを作成する際に使用する言語の正式な文法を示します。
- [AWSジョブ機能の管理ポリシー](#) - このセクションでは、IT 業界の一般的なジョブ機能に直接マッピングする AWS 管理ポリシーについて説明します。これらのポリシーを使用して、特定のジョブ

機能を持つユーザーによるタスクの実行に必要なアクセス許可を付与します。これらのポリシーによって、多くのサービスのアクセス権限が1つのポリシーに一元化されます。

- [AWS グローバル条件コンテキストキー](#) - このセクションには、IAM ポリシーのアクセス許可を制限するのに使用できるすべての AWS グローバル条件キーのリストが含まれています。
- [IAM および AWS STS の条件コンテキストキー](#) — このセクションには、IAM ポリシーのアクセス許可を制限するのに使用できるすべての IAM 条件キーおよび AWS STS 条件キーのリストが含まれています。
- [AWS サービスのアクション、リソース、および条件キー](#) — このセクションでは、IAMポリシーでアクセス許可として使用できるすべての AWS API オペレーションのリストを示します。また、そのリクエストをさらに絞り込むのに使用できるサービス固有の条件キーも含まれています。

## IAM JSON ポリシー要素のリファレンス

JSON ポリシードキュメントは要素で構成されます。要素は、ポリシーで使用する一般的な順番で記載されています。要素の順番は重要ではありません (たとえば、Resource 要素を Action 要素の前にもってくるなどが可能です)。ポリシーで、あらゆる Condition 要素も特定する必要はありません。JSON ポリシードキュメントの全体構造と目的については「[JSON ポリシー概要](#)」をご覧ください。

一部の JSON ポリシーの要素は相互排他的です。つまり、両方を使用するポリシーを作成することはできません。たとえば、Action と NotAction を同じポリシーステートメントで使用することはできません。相互排他的な他のペアには Principal/NotPrincipal や Resource/NotResource があります。

ポリシーに取り入れる詳細は各サービスによって異なり、サービスで利用可能なアクションやリソースの種類などにより異なります。特定のサービスのポリシーを記述している場合、そのサービスに関するポリシーの例を参照することが役に立ちます。IAM をサポートするすべてのサービスのリスト、およびそれらのサービスの IAM とポリシーについて説明しているドキュメントへのリンクについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

JSON ポリシーを作成または編集するときに、IAM はポリシー検証を実行し、効果的なポリシーを作成するのに役立ちます。IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックを提供します。ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。IAM Access Analyzer のポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」IAM Access Analyzer ポリシーの検証を参照してください。

## トピック

- [IAM JSON ポリシー要素Version](#)
- [IAM JSON ポリシー要素Id](#)
- [IAM JSON ポリシー要素Statement](#)
- [IAM JSON ポリシー要素Sid](#)
- [IAM JSON ポリシー要素Effect](#)
- [AWS JSON ポリシーの要素: Principal](#)
- [AWS JSON ポリシーの要素: NotPrincipal](#)
- [IAM JSON ポリシー要素Action](#)
- [IAM JSON ポリシー要素NotAction](#)
- [IAM JSON ポリシー要素Resource](#)
- [IAM JSON ポリシー要素NotResource](#)
- [IAM JSON ポリシー要素Condition](#)
- [IAM ポリシーの要素: 変数とタグ](#)
- [IAM JSON ポリシー要素: サポートされているデータ型](#)

## IAM JSON ポリシー要素Version

### あいまいさに関する注意

この Version JSON ポリシー要素はポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。一方で、ポリシーバージョンは、IAM でカスタマー管理ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。管理ポリシーに対する複数のバージョンのサポートに関する情報を探している場合は、「[the section called “IAM ポリシーのバージョンニング”](#)」を参照してください。

Version ポリシー要素は、このポリシーを処理するために使用される言語構文ルールを指定します。使用可能なポリシーの機能をすべて使用するには、すべてのポリシーの Statement 要素の外部にある以下の Version 要素を含めます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

IAM は以下の Version 要素値をサポートしています。

- 2012-10-17. これはポリシー言語の現行バージョンであり、常に Version 要素を含め、2012-10-17 に設定する必要があります。このようにしない場合、このバージョンで導入された[ポリシー変数](#)などの機能は使用できません。
- 2008-10-17. これはポリシー言語の旧バージョンです。既存のポリシーで古めのものでは、このバージョンが表示される場合があります。新規ポリシーを作成するときや既存ポリシーを更新するときは、この旧バージョンを使用しないでください。ポリシー変数などの新しい機能は、ポリシーでは機能しません。たとえば、`${aws:username}` などの変数は変数として認識されず、代わりにポリシー内のリテラル文字列として扱われます。

## IAM JSON ポリシー要素 Id

Id 要素は、ポリシーで使用する任意の識別子を特定します。ID の使用方法は、サービスによって異なります。ID は、リソースベースのポリシーでは使用できますが、アイデンティティベースのポリシーでは使用できません。

ID 要素を設定するサービスの場合、UUID (GUID) を値に使用する、または UUID を唯一性を確認するための ID の一部として統合することを推奨します。

```
{
  "Version": "2012-10-17",
  "Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

```
}  
]  
}
```

### Note

AWS のサービス (たとえば、Amazon SQS や Amazon SNS など) には、この要素を要求し、唯一条件を与えるものもあります。ポリシーの記述に関するサービス固有の情報は、お取り扱いのサービス用のドキュメントを参照してください。

## IAM JSON ポリシー要素Statement

Statement 要素は、ポリシーの主要要素です。この要素は必須です。Statement 要素には、単一のステートメントまたは個々のステートメントの配列を含めることができます。個々のステートメントブロックは、中括弧 {} で囲む必要があります。複数のステートメントの場合、配列は角括弧 [] で囲む必要があります。

```
"Statement": [{...},{...},{...}]
```

以下の例は、1つの Statement 要素の中に3つのステートメントの配列を含むポリシーを示しています。(このポリシーにより、Amazon S3 コンソール内の自分の "ホームフォルダー" にアクセスできます)。ポリシーには `aws:username` 変数が含まれ、変数はポリシーの評価時にリクエストからのユーザー名に置き換えられます。詳細については、「[序章](#)」を参照してください。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:ListAllMyBuckets",  
        "s3:GetBucketLocation"  
      ],  
      "Resource": "arn:aws:s3:::*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "s3:ListBucket",  
      "Resource": "arn:aws:s3:::BUCKET-NAME",  
    }  
  ]  
}
```



```
    "Condition": {"StringLike": {"s3:prefix": [
      "",
      "home/",
      "home/${aws:username}/"
    ]}}
  },
  {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
      "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
    ]
  }
]
```

## IAM JSON ポリシー要素Sid

ポリシーステートメントのオプションの識別子として、Sid (ステートメントID) を指定することができます。Sid 値は、ステートメント配列内の各ステートメントに割り当てることができます。Sid の値は、ポリシーステートメントの説明として使用できます。SQS や SNS などの ID 要素を特定するサービスでは、Sid 値はポリシードキュメント ID の副 ID に過ぎません。IAM では、Sid 値は JSON ポリシー内で固有のものでなければいけません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

-Sid要素は、ASCII 大文字 (A～Z)、小文字 (a～z)、および数字 (0～9) をサポートします。

IAM は、IAM API でSidを公開しません。この ID に基づいて、特定のステートメントを復元することはできません。

**Note**

AWS のサービス (例えば、Amazon SQS や Amazon SNS など) には、この要素を要求し、唯一条件を与えるものもあります。ポリシーの記述に関するサービス固有の情報は、お取り扱いのサービス用のドキュメントを参照してください。

## IAM JSON ポリシー要素 Effect

Effect 要素は必須であり、ステートメントの結果を許可または明示的な拒否のどちらにするかを指定します。Effect の有効値は、Allow と Deny です。Effect 値では、大文字と小文字が区別されます。

```
"Effect": "Allow"
```

デフォルト設定では、リソースへのアクセスは拒否されます。リソースへのアクセスを許可するには、Effect 要素を Allow に設定する必要があります。許可を無効にするには (たとえば、本来有効となる許可を無効にするなど)、Effect 要素を Deny に設定します。詳細については、「[ポリシーの評価論理](#)」を参照してください。

## AWS JSON ポリシーの要素: Principal

リソースベースの JSON ポリシーの Principal 要素を使用して、リソースへのアクセスを許可または拒否するプリンシパルを指定します。

[リソースベースポリシー](#) の Principal 要素を使用する必要があります。IAM など、いくつかのサービスが、リソースベースのポリシーをサポートしています。IAM リソースベースのポリシーのタイプは、ロールの信頼ポリシーです。IAM ロールでは、ロールの信頼ポリシー内の Principal 要素を使用して、だれがこのロールを引き受けることができるかを指定します。クロスアカウントアクセスとして、信頼されたアカウントの 12 桁の ID を指定する必要があります。信頼ゾーン (信頼できる組織またはアカウント) 外にあるアカウントのプリンシパルにロールを引き受けるアクセス権があるかどうかについては、「[IAM Access Analyzer とは](#)」を参照してください。

**Note**

ロールを作成した後、アカウントを「\*」に変更すると、誰もがそのロールを引き受けることができます。これを行う場合は、アクセスを特定の IP アドレスのみに制限する Condition

要素など、他の方法でロールへのアクセスを制限することを強くお勧めします。誰でもロールにアクセスできる状態のままにしないでください。

サポートベースのポリシーをサポートする他のサービスの例としては、Amazon S3 や AWS KMS key が挙げられます。

Principal エlementを ID ベースのポリシーで使用することはできません。ID ベースのポリシーは、IAM の ID (ユーザー、グループ、ロール) にアタッチするアクセス許可ポリシーです。これらの場合、プリンシパルは、ポリシーがアタッチされた ID によって暗示されます。

## トピック

- [プリンシパルの指定](#)
- [AWS アカウント プリンシパル](#)
- [IAM ロールプリンシパル](#)
- [ロールセッションプリンシパル](#)
- [IAM ユーザープリンシパル](#)
- [IAM Identity Center のプリンシパル](#)
- [AWS STS フェデレーティッドユーザーセッションプリンシパル](#)
- [AWS サービスプリンシパル](#)
- [オプトインリージョンの AWS サービスプリンシパル](#)
- [すべてのプリンシパル](#)
- [詳細情報](#)

## プリンシパルの指定

リソースベースのポリシーにある Principal 要素が、プリンシパルをサポートする条件キーでプリンシパルを指定します。

ポリシーでは、次のいずれかのプリンシパルを指定できます。

- AWS アカウント およびルートユーザー
- IAM ロール
- ロールセッション

- IAM ユーザー
- フェデレーテッドユーザーセッション
- AWS サービス
- すべてのプリンシパル

ユーザーグループをポリシー (リソースベースのポリシーなど) のプリンシパルとして識別することはできません。これは、グループは 認証ではなくアクセス権限に関連しており、プリンシパルは認証済みの IAM エンティティであるためです。

配列を使用して、以降のセクションの各プリンシパル型に対して複数のプリンシパルを指定できます。配列では 1 つまたは複数の値を使用できます。要素で複数のプリンシパルを指定する場合は、各プリンシパルにアクセス許可を付与します。これは論理 OR であり論理 AND ではありません。一度に 1 つのプリンシパルを認証するためです。複数の値を含める場合は、角括弧 ([ と ]) を使用し、配列の各エントリをカンマで区切ります。次のポリシーの例では、123456789012 アカウントまたは 555555555555 アカウントのアクセス許可を定義します。

```
"Principal" : {
  "AWS": [
    "123456789012",
    "555555555555"
  ]
}
```

#### Note

ワイルドカードとして使用して、プリンシパルの名前または ARN の一部に一致させることはできません。

## AWS アカウント プリンシパル

リソースベースのポリシーにある Principal 要素が、プリンシパルをサポートする条件キーで、AWS アカウント の識別子を指定できます。これにより、権限がアカウントに委譲されます。別のアカウントへのアクセスを許可する場合、そのアカウントの管理者が、そのアカウントの ID (IAM ユーザーまたはロール) へのアクセス許可を付与する必要があります。AWS アカウント を指定するときは、アカウント ARN (aarn:aws:iam::*account-ID*:root、または "AWS": プレフィックスの後に ID を付けた短縮形を使用できます。

例えば、割り当てられたアカウント ID (123456789012) から、次のいずれかのメソッドを使用して、Principal 要素でそのアカウントを指定できます。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": { "AWS": "123456789012" }
```

アカウント ARN と短縮アカウント ID は、同じように動作します。どちらも、アカウントにアクセス許可を委譲します。Principal 要素でアカウント ARN を使用しても、アカウントのルートユーザーだけはアクセス許可を制限しません。

#### Note

短縮アカウント ID を含むリソースベースのポリシーを保存すると、それをサービスがプリンシパル ARN に変換することがあります。これはポリシーの機能は変更しません。

一部の AWS サービスでは、アカウントプリンシパルを指定するための追加オプションがサポートされています。例えば、Amazon S3 では、次の形式を使用して [正規ユーザー ID](#) を指定できます。

```
"Principal": { "CanonicalUser":  
  "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

配列を使用すると、プリンシパルとして AWS アカウント (または正規ユーザー ID) を 1 つ以上指定できます。例えば、3 つのメソッドすべてを使用して、バケットポリシーでプリンシパルを指定できます。

```
"Principal": {  
  "AWS": [  
    "arn:aws:iam::123456789012:root",  
    "999999999999"  
  ],  
  "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"  
}
```

## IAM ロールプリンシパル

リソースベースのポリシーにある Principal 要素が、プリンシパルをサポートする条件キーで、IAM ロールプリンシパル ARN を指定できます。IAM ロールは ID です。IAM では、ID はアクセ

ス許可を割り当てることができるリソースです。ロールは、別の認証済みの ID を信頼して、そのロールを引き受けます。これには、AWS のプリンシパルか、外部 ID プロバイダー (IdP) からユーザーが含まれます。プリンシパルまたは ID がロールを引き受けると、引き受けたロールのアクセス許可を持った、一時的なセキュリティ認証情報を受け取ります。AWS でこれらのセッション認証情報を使用して操作を実行すると、これらはロールセッションプリンシパルになります。

IAM ロールは、IAM に存在する ID です。ロールは、AWS のプリンシパルなどの認証済みの ID か、外部 ID プロバイダー からユーザーを信頼します。プリンシパルまたは ID がロールを引き受けると、一時的なセキュリティ認証情報を受け取ります。その後、これらの認証情報をロールセッションプリンシパルとして使用して、AWS で操作を実行できます。

リソースベースのポリシーでロールプリンシパルを指定すると、ロールのアクセス許可を制限するポリシータイプによって、プリンシパルの有効なアクセス許可が制限されます。これには、セッションポリシーとアクセス許可の境界が含まれます。ロールセッションの有効なアクセス許可の評価方法については、「[ポリシーの評価論理](#)」を参照してください。

Principal 要素でロール ARN を指定する場合は、次の形式を使用します。

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:role/role-name" }
```

#### Important

ロール信頼ポリシーの Principal 要素に、特定の IAM ロールを指し示す ARN が含まれている場合、その ARN はポリシーを保存するときにロールの一意のプリンシパル ID に変換されます。これにより、ロールを削除して再作成することにより、誰かがそのユーザーの特権をエスカレートするリスクを緩和できます。通常、この ID はコンソールには表示されません。これは、信頼ポリシーが表示されるときに、IAM がロール ARN への逆変換を行うためです。ただし、ロールを削除すると、関係が壊れます。ロールを再作成した場合でも、ポリシーは適用されません。これは、新しいロールは信頼ポリシーに保存されている ID と一致しない新しいプリンシパル ID を持っているためです。この場合、プリンシパル ID はリソースベースポリシーに表示されます。これは AWS が有効な ARN に戻って ID をマッピングできなくなるためです。その結果、信頼ポリシーの Principal 要素で参照されているロールを削除して再作成する場合は、ポリシーのロールを編集してプリンシパル ID を正しい ARN に置き換える必要があります。ポリシーを保存するときに、ARN は再びロールの新しいプリンシパル ID に変換されます。

または、ロールプリンシパルをリソースベースのポリシーのプリンシパルとして指定するか、[幅広くアクセスを許可するポリシーを作成](#)して `aws:PrincipalArn` 条件キーを使用します。このキーを使用すると、ロールセッションプリンシパルには、セッションの結果として得られる ARN ではなく、引き受けたロールの ARN に基づくアクセス許可が付与されます。なぜなら、AWS は条件キー ARN を ID に変換せず、ロールを削除してから同じ名前でも新しくロールを作成すると、ロール ARN に付与されたアクセス許可は保持されるからです。アクセス許可の境界やセッションポリシーなどの ID ベースのポリシータイプは、アイデンティティベースのポリシーに明示的な拒否が含まれていない限り、Principal 要素で、ワイルドカード (\*) を含む `aws:PrincipalArn` 条件キーを使用して付与されたアクセス許可を制限することはありません。

## ロールセッションプリンシパル

リソースベースのポリシーにある Principal 要素が、プリンシパルをサポートする条件キーで、ロールセッションを指定できます。プリンシパルまたは ID がロールを引き受けると、引き受けたロールのアクセス許可を持った、一時的なセキュリティ認証情報を受け取ります。AWS でこれらのセッション認証情報を使用して操作を実行すると、これらはロールセッションプリンシパルになります。

ロールセッションプリンシパルに使用する形式は、ロールの引き受けに使用する AWS STS のオペレーションによります。

さらに管理者は、ロールセッションの発行方法を制御するプロセスを設計できます。例えば、ユーザーがワンクリックするだけで、予測可能なセッション名を作成するソリューションを提供できます。これを管理者が行う場合、ポリシーまたは条件キーで、ロールセッションプリンシパルを使用できます。それ以外の場合は、ロール ARN を `aws:PrincipalArn` 条件キーのプリンシパルとして指定することができます。ロールをどのようにプリンシパルとして指定するかによって、セッション完了後の有効なアクセス許可が変わります。詳細については、「[IAM ロールプリンシパル](#)」を参照してください。

## 引き受けたロールのセッションプリンシパル

引き受けたロールのセッションプリンシパルは、AWS STS AssumeRole オペレーションの結果として得られるセッションプリンシパルです。このオペレーションを使用してロールを引き受けられるプリンシパルについては、「[AWS STS API オペレーションの比較](#)」を参照してください。

Principal 要素で引き受けたロールの ARN を指定する場合は、次の形式を使用します。

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```



Principal 要素で引き受けたロールセッションを指定する場合、ワイルドカード (\*) を使用して「すべてのセッション」を意味することはできません。プリンシパルは常に特定のセッションに名前を付ける必要があります。

## OIDC セッションプリンシパル

「OIDC セッションプリンシパル」は、AWS STS AssumeRoleWithWebIdentity オペレーションの結果として得られるセッションプリンシパルです。外部の OIDC プロバイダー (IdP) を使用してサインインし、この操作を使用して IAM ロールを引き受けることができます。これは ID フェデレーションを利用して、ロールセッションを発行します。このオペレーションを使用してロールを引き受けることができるプリンシパルについては、「[AWS STS API オペレーションの比較](#)」を参照してください。

OIDC プロバイダーからロールを発行すると、OIDC プロバイダーに関する情報を含む特別なタイプのセッションプリンシパルが取得されます。

このプリンシパルタイプをポリシーで使用して、信頼済みの Web ID プロバイダーに基づき、アクセスを許可または拒否します。ロール信頼ポリシーの Principal 要素で OIDC ロールセッション ARN を指定する場合は、次の形式を使用します。

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

## SAML セッションプリンシパル

SAML セッションプリンシパルは、AWS STS AssumeRoleWithSAML オペレーションの結果として得られるセッションプリンシパルです。外部の SAML ID プロバイダー (IdP) を使用してサインインし、この操作を使用して IAM ロールを引き受けることができます。これは ID フェデレーションを利用して、ロールセッションを発行します。このオペレーションを使用してロールを引き受けることができるプリンシパルについては、「[AWS STS API オペレーションの比較](#)」を参照してください。

SSAML ID プロバイダーからロールを発行すると、SAML ID プロバイダーの情報が含まれた、特別なタイプのセッションプリンシパルが取得されます。



このプリンシパルタイプをポリシーで使用して、信頼済みの SAML ID プロバイダーに基づき、アクセスを許可または拒否します。ロール信頼ポリシーの Principal 要素で SAML ID ロールセッション ARN を指定する場合は、次の形式を使用します。

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:saml-provider/provider-name" }
```

## IAM ユーザープリンシパル

リソースベースのポリシーの Principal 要素、またはプリンシパルをサポートする条件キーで、IAM ユーザーを指定できます。

### Note

Principal 要素にある [Amazon リソースネーム\(ARN\)](#) のユーザー名の部分では、大文字と小文字を区別します。

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:user/user-name" }
```

```
"Principal": {  
  "AWS": [  
    "arn:aws:iam::AWS-account-ID:user/user-name-1",  
    "arn:aws:iam::AWS-account-ID:user/user-name-2"  
  ]  
}
```

Principal 要素内でユーザーを指定する際に、"すべてのユーザー" の意味でワイルドカード (\*) を使用することはできません。プリンシパルには、常に複数の特定ユーザーを指定する必要があります。

### Important

ロールの信頼ポリシーの Principal 要素に、特定の IAM ユーザーを指し示す ARN が含まれている場合、その ARN をポリシーに保存するときに、IAM がユーザーの一意のプリンシパル ID に変換されます。これにより、ユーザーを削除して再作成することにより、誰かがそのユーザーの特権をエスカレートするリスクを緩和できます。通常、この ID はコンソールには表示されません。これは、信頼ポリシーが表示されるときに、ユーザーの ARN への逆変換が行われるためです。ただし、ユーザーを削除すると、関係が壊れます。ユーザーを

再作成しても、ポリシーが適用されることはありません。これは、新しいユーザーには、信頼ポリシーに保存されている ID と一致しない新しいプリンシパル ID が付与されるためです。この場合、プリンシパル ID はリソースベースポリシーに表示されます。これは AWS が有効な ARN に戻って ID をマッピングできなくなるためです。その結果、信頼ポリシーの Principal 要素で参照されているユーザーを削除して再作成する場合は、ロールを編集して、正しくなくなったプリンシパル ID を正しい ARN に置き換える必要があります。ポリシーを保存するときに、IAM が再び、ARN をユーザーの新しいプリンシパル ID に変換します。

## IAM Identity Center のプリンシパル

IAM Identity Center では、リソースベースのポリシーのプリンシパルが AWS アカウント プリンシパルとして定義されている必要があります。アクセスを指定するには、条件ブロック内のアクセス許可セットのロール ARN を参照してください。詳細については、「IAM Identity Center ユーザーガイド」の「[リソースポリシー、Amazon EKS、および AWS KMS のアクセス許可セットの参照](#)」を参照してください。

## AWS STS フェデレーティッドユーザーセッションプリンシパル

リソースベースのポリシーにある Principal 要素が、プリンシパルをサポートする条件キーで、フェデレーティッドユーザーセッションを指定できます。

### Important

AWS では、[ルートユーザーアクセスが必要](#)などの場合にのみ、AWS STS フェデレーティッドユーザーセッションを使用することをお勧めします。代わりに、[ロールを使用してアクセス許可を委任してください](#)。

AWS STS フェデレーティッドユーザーセッションプリンシパルは、AWS STS GetFederationToken オペレーションの結果として得られるセッションプリンシパルです。この場合は、IAM ロールを使用する代わりに一時的なアクセストークンを取得する方法として、AWS STS が [ID フェデレーション](#) を使用しています。

AWS では、IAM ユーザーまたは AWS アカウントのルートユーザーは、長期的なアクセスキーを使用して認証を行うことができます。このオペレーションで、どのプリンシパルがフェデレーションを行えるかについては、「[AWS STS API オペレーションの比較](#)」を参照してください。

- IAM フェデレーティッドユーザー — GetFederationToken オペレーションを使用して IAM ユーザーがフェデレーションを行った結果、IAM ユーザーのフェデレーティッドユーザーセッションプリンシパルができます。
- フェデレーティッドルートユーザー — GetFederationToken オペレーションを使用してルートユーザーがフェデレーションを行った結果、ルートユーザーのフェデレーティッドユーザーセッションプリンシパルができます。

この操作を使用して、IAM ユーザーまたはルートユーザーが AWS STS から一時的な認証情報をリクエストした場合、一時的なフェデレーティッドユーザーセッションを開始します。このセッションの ARN は、フェデレーション元の ID に基づいています。

Principal 要素でフェデレーティッドユーザーセッション ARN を指定する場合は、次の形式を使用します。

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:federated-user/user-name" }
```

## AWS サービスプリンシパル

リソースベースのポリシーにある Principal 要素か、プリンシパルをサポートする条件キーで、AWS サービスを指定できます。サービスプリンシパルは、サービスの識別子です。

AWS のサービスが引き受けることのできる IAM ロールは、[サービスロール](#)と呼ばれます。サービスロールには信頼ポリシーを含める必要があります。信頼ポリシーは、どのプリンシパルがそのロールを果たすことができるかを定義するロールにアタッチされるリソースベースのポリシーです。一部のサービスロールには事前定義済みの信頼ポリシーがあります。ただし、状況によっては、信頼ポリシーでプリンシパルサービスを指定する必要があります。IAM ポリシーのサービスプリンシパルを "Service": "\*" にすることはできません。

サービスプリンシパルの識別子にはサービス名が含まれ、通常は次の形式になります。

*service-name*.amazonaws.com

サービスプリンシパルはサービスによって定義されます。一部のサービスのサービスプリンシパルを検索するには [IAM と連携する AWS のサービス](#) を開き、サービスの [Service-linked role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているかどうかを確認し、[Yes] (はい) リンクを開いてそのサービスのサービスにリンクされたロールのドキュメントを表示します。サービスプリンシパルを表示するには、そのサービスの [Service-Linked Role Permissions] (サービスにリンクされたロールのアクセス許可) のセクションを探します。

次の例では、サービスロールにアタッチできるポリシーを示します。ポリシーは、Amazon ECS および Elastic Load Balancing の 2 つのサービスを有効にして、ロールを引き受けます。これらのサービスは、ロールに割り当てられた権限ポリシー (表示されていない) によって付与されたタスクを実行できます。複数のサービスプリンシパルを指定する場合に、2 つの Service 要素は指定できません。1 つのみ指定できます。代わりに、1 つの Service 要素の値として複数のサービスのプリンシパルのアレイを使用します。

```
"Principal": {
  "Service": [
    "ecs.amazonaws.com",
    "elasticloadbalancing.amazonaws.com"
  ]
}
```

### オプトインリージョンの AWS サービスプリンシパル

リソースは複数の AWS リージョンで起動できますが、一部のリージョンはオプトインする必要があります。オプトインする必要があるリージョンの一覧については、「AWS 全般のリファレンスガイド」の「[AWS リージョンの管理](#)」を参照してください。

オプトインリージョンの AWS サービスが同じリージョン内でリクエストを行うと、サービスプリンシパル名の形式は、サービスプリンシパル名がリージョン化されないバージョンとして識別されます。

*service-name*.amazonaws.com

オプトインリージョンの AWS サービスが別のリージョンにクロスリージョンリクエストを行うと、サービスプリンシパル名の形式は、サービスプリンシパル名のリージョン化バージョンとして識別されます。

*service-name.{region}*.amazonaws.com

例えば、Amazon SNS トピックが ap-southeast-1 リージョンにあり、Amazon S3 バケットがオプトインリージョン ap-east-1 にあるとします。メッセージを SNS トピックに公開するように、S3 バケット通知を設定する必要があります。S3 サービスが SNS トピックにメッセージを投稿できるようにするには、トピックのリソースベースのアクセスポリシーを使用して S3 サービスのプリンシパル sns:Publish 権限を付与する必要があります。

トピックアクセスポリシーで S3 サービスプリンシパルのリージョン化されないバージョンである s3.amazonaws.com を指定すると、バケットからトピックへの sns:Publish リクエストは失敗

します。次の例では、SNS トピックアクセスポリシーの Principal ポリシー要素に、リージョン化されない S3 サービスプリンシパルを指定しています。

```
"Principal": { "Service": "s3.amazonaws.com" }
```

バケットはオプトインリージョンにあり、リクエストは同じリージョン外で行われるため、S3 サービスプリンシパルはリージョン化されたサービスプリンシパル名 `s3.ap-east-1.amazonaws.com` として表示されます。オプトインリージョンの AWS サービスが別のリージョンにリクエストを行う場合は、リージョン化されたサービスプリンシパル名を使用する必要があります。リージョン化されたサービスプリンシパル名を指定した後に、バケットが別のリージョンにある SNS トピックに `sns:Publish` リクエストを行うと、リクエストは成功します。次の例では、SNS トピックアクセスポリシーの Principal ポリシー要素に、リージョン化された S3 サービスプリンシパルを指定しています。

```
"Principal": { "Service": "s3.ap-east-1.amazonaws.com" }
```

オプトインリージョンから別のリージョンへのクロスリージョンリクエストに対するリソースポリシーまたはサービスプリンシパルベースの許可リストは、リージョン化されたサービスプリンシパル名を指定した場合にのみ成功します。

#### Note

IAM ロールの信頼ポリシーには、リージョン化されないサービスプリンシパル名を使用することをお勧めします。IAM リソースはグローバルであるため、どのリージョンでも同じロールを使用することができます。

## すべてのプリンシパル

ワイルドカード (\*) を使用して Principal リソースベースのポリシーの要素またはプリンシパルをサポートする条件キーにあるすべてのプリンシパルを指定できます。[リソースベースのポリシー許可](#)のアクセス権および[条件キー](#)はポリシーステートメントの条件を制限するために使用されます。

#### Important

パブリックまたは匿名アクセスを許可する意図がない限り、Allow の影響を伴うリソースベースのポリシーの Principal 要素にワイルドカード (\*) を使用しないことを強くお勧めします。それ以外の場合は、Principal 要素で目的のプリンシパル、サービス、または

AWS アカウントを指定して、Condition 要素でさらにアクセスを制限します。これは、他のプリンシパルがアカウントのプリンシパルになることを許可することから、特にIAM ロールの信頼ポリシーに当てはまります。

リソースベースポリシーでは、Allow 効果でワイルドカード (\*) を使用することで、匿名ユーザー (パブリックアクセス) を含むすべてのユーザーへのアクセスが許可されます。アカウント内の IAM ユーザーおよびロールプリンシパルの場合、他のアクセス権は必要ありません。他のアカウントのプリンシパルの場合、アカウント内にリソースへのアクセスを許可する ID ベースのアクセス許可がある必要があります。これは [クロスアカウントアクセス](#) と呼ばれます。

匿名ユーザーの場合、次の要素は同等です。

```
"Principal": "*" 
```

```
"Principal" : { "AWS" : "*" } 
```

ワイルドカードとして使用して、プリンシパルの名前または ARN の一部に一致させることはできません。

次の例は、Condition 要素で指定されたものを除いたすべてのプリンシパルを明確に拒否する [Deny とともに NotPrincipal を指定する](#) の代わりに使用できるリソースベースのポリシーを示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UsePrincipalArnInsteadOfNotPrincipalWithDeny",
      "Effect": "Deny",
      "Action": "s3:*",
      "Principal": "*",
      "Resource": [
        "arn:aws:s3:::BUCKETNAME/*",
        "arn:aws:s3:::BUCKETNAME"
      ],
      "Condition": {
        "ArnNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::444455556666:user/user-name"
        }
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

## 詳細情報

詳細については、次を参照してください:

- 「Amazon Simple Storage Service ユーザーガイド」にある [バケットポリシーの例](#)
- 「Amazon Simple Notification Service 開発者ガイド」にある [Amazon SNS のポリシーの例](#)
- 「Amazon Simple Queue Service 開発者ガイド」にある [Amazon SQS ポリシーの例](#)
- 「AWS Key Management Service 開発者ガイド」にある [主要ポリシー](#)
- 「AWS 全般のリファレンス」の「[アカウント ID](#)」
- [OIDC フェデレーション](#)

## AWS JSON ポリシーの要素: NotPrincipal

NotPrincipal 要素を使用すると、NotPrincipal 要素で指定された IAM ユーザー、フェデレーションユーザー、IAM ロール、AWS アカウント、AWS のサービスなどのプリンシパル以外のすべてのプリンシパルへのアクセスを拒否できます。

これは、VPC エンドポイントなど、一部の AWS のサービスのリソースベースポリシーで使用できます。リソースベースのポリシーは、リソースに直接埋め込むポリシーです。IAM のアイデンティティベースのポリシーでも IAM ロールの信頼ポリシーでも NotPrincipal 要素は使用できません。

NotPrincipal は "Effect": "Deny" とともに使用する必要があります。"Effect": "Allow" とともに使用することはサポートされていません。

### Important

NotPrincipal の使用が必要になるシナリオはほとんどありません。NotPrincipal の使用を決定する前に、他の認証オプションを検討するようお勧めします。NotPrincipal を使用すると、複数のポリシータイプの影響をトラブルシューティングすることが難しい場合があります。代わりに ARN 条件演算子で `aws:PrincipalArn` コンテキストキーを使用することをおすすめします。詳細については、「[すべてのプリンシパル](#)」を参照してください。



## Deny とともに NotPrincipal を指定する

NotPrincipal で Deny を使用する場合は、拒否されていないプリンシパルのアカウントの ARN も設定する必要があります。それ以外の場合、ポリシーでは、そのプリンシパルを含むアカウント全体へのアクセスが拒否されます。ポリシーに含めるサービスに応じて、AWS は最初にアカウントを確認し、次にユーザーを確認する場合があります。引き受けたロールのユーザー (ロールを使用しているユーザー) を評価する場合、AWS は最初にアカウントを確認し、次にロール、最後にロールを引き受けたユーザーを確認する場合があります。ロールを引き受けたユーザーは、ユーザーがロールを引き受けたときに指定されたロールセッション名で識別されます。したがって、ユーザーのアカウントの ARN、またはロールの ARN とそのロールを含むアカウントの ARN の両方を、明示的に含めることを強くお勧めします。

### Important

アクセス許可の境界ポリシーがアタッチされている IAM ユーザーまたはロールに対する Deny 効果を持つ NotPrincipal ポリシー要素を含むリソースベースのポリシーステートメントは使用しないでください。Deny 効果のある NotPrincipal 要素は、NotPrincipal 要素で指定されている値に関係なく、アクセス許可の境界ポリシーがアタッチされている IAM プリンシパルを常に拒否します。これにより、本来であればリソースにアクセスできたはずの IAM ユーザーまたはロールの一部がアクセスを失うことになります。リソースベースのポリシーステートメントを変更して、NotPrincipal 要素ではなく [aws:PrincipalArn](#) コンテキストキーで条件演算子 [ArnNotEquals](#) を使用してアクセスを制限することをおすすめします。アクセス許可の境界の詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。

### Note

ベストプラクティスとして、アカウントの ARN をポリシーに含めます。一部のサービスではアカウント ARN が必要ですが、すべてのケースで必要になるわけではありません。必要な ARN のない既存のポリシーは引き続き機能しますが、これらのサービスを含む新しいポリシーはこの要件を満たす必要があります。IAM はこれらのサービスを追跡しないため、常にアカウント ARN を含めることをお勧めします。

以下の例では、同じポリシーステートメント内で NotPrincipal と "Effect": "Deny" を効果的に使用する方法を示しています。



## Example 同じまたは異なるアカウントの IAM ユーザーの例

以下の例では、444455556666 という AWS アカウント に含まれる Bob という名前のユーザーを除いて、すべてのプリンシパルによるリソースへのアクセスが明示的に拒否されています。ベストプラクティスとして、NotPrincipal 要素には、Bob というユーザーの ARN と、Bob が属する AWS アカウント (arn:aws:iam::444455556666:root) の ARN が両方含まれています。NotPrincipal 要素に含まれているのが Bob の ARN のみであれば、このポリシーの結果として、ユーザー Bob を含む AWS アカウント のアクセスが明示的に拒否されることがあります。場合によっては、ユーザーは、親アカウントを超えるアクセス許可を得ることができないため、Bob のアカウントが明示的にアクセスを拒否されると、Bob もリソースにアクセスできなくなる可能性があります。

この例は、同じまたは異なる AWS アカウント (444455556666 ではない) のリソースにアタッチされているリソーススペースのポリシー内でポリシーステートメントに含まれている場合、意図したとおりに動作します。この例自体は Bob にアクセス許可を付与しておらず、明示的に拒否するプリンシパルのリストから Bob を除外しているだけです。リソースへのアクセスを Bob に許可するには、別のポリシーステートメントで "Effect": "Allow" を使用して明示的にアクセスを許可する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "NotPrincipal": {"AWS": [
      "arn:aws:iam::444455556666:user/Bob",
      "arn:aws:iam::444455556666:root"
    ]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::BUCKETNAME",
      "arn:aws:s3:::BUCKETNAME/*"
    ]
  }]
}
```

## Example 同じまたは異なるアカウントの IAM ロールの例

以下の例では、444455556666 という AWS アカウント に含まれる cross-account-audit-app という名前の引き受けたロールユーザーを除いて、すべてのプリンシパルによるリソースへのアクセスが明示的に拒否されています。ベストプラクティスとして、NotPrincipal 要素には、引き受けたロー

ルユーザー (cross-account-audit-app)、ロール (cross-account-read-only-role)、ロールが属する AWS アカウント (444455556666) の ARN が含まれています。ロールの ARN が NotPrincipal 要素に含まれていなければ、このポリシーの効果としては、このロールに対して、アクセスを明示的に拒否することになります。同様に、ロールが属する AWS アカウントの ARN が NotPrincipal エlement に含まれていなければ、このポリシーの結果として、AWS アカウントとそのアカウントに属するすべてのエンティティへのアクセスは明示的に拒否されることがあります。場合によっては、引き受けたロールユーザーは、親ロールを超えるアクセス許可を得ることができず、ロールは、親である AWS アカウントを超えるアクセス許可を得ることができないため、ロールまたはアカウントが明示的にアクセスを拒否された場合、引き受けたロールユーザーはリソースにアクセスできなくなる可能性があります。

この例は、別の AWS アカウント (444455556666 ではない) のリソースにアタッチされているリソーススペースのポリシー内でポリシーステートメントに含まれている場合、意図したとおりに動作します。この例自体は引き受けたロールユーザー (cross-account-audit-app) にアクセスを許可しておらず、明示的に拒否するプリンシパルのリストから cross-account-audit-app を除外しているだけです。リソースへのアクセス許可を cross-account-audit-app に付与するには、別のポリシーステートメントで "Effect": "Allow" を使用して明示的にアクセスを許可する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "NotPrincipal": {"AWS": [
      "arn:aws:sts::444455556666:assumed-role/cross-account-read-only-role/cross-account-audit-app",
      "arn:aws:iam::444455556666:role/cross-account-read-only-role",
      "arn:aws:iam::444455556666:root"
    ]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::Bucket_AccountAudit",
      "arn:aws:s3:::Bucket_AccountAudit/*"
    ]
  }]
}
```

NotPrincipal 要素で引き受けたロールセッションを指定する場合、ワイルドカード (\*) を使用して「すべてのセッション」を意味することはできません。プリンシパルは常に特定のセッションに名前を付ける必要があります。

## IAM JSON ポリシー要素Action

Action 要素は、許可または拒否される特定のアクションについて説明します。ステートメントには、Action または NotAction 要素を含める必要があります。各 AWS 製品には、そのサービスで行うことができるタスクを記述する独自のアクションセットがあります。例えば、Amazon S3 のアクションのリストは、[Amazon Simple Storage Service ユーザーガイド](#)のポリシーでのアクセス許可の指定にあり、Amazon EC2 のアクションのリストは、[Amazon EC2 API リファレンス](#)にあり、AWS Identity and Access Management のアクションのリストは、[IAM API リファレンス](#)に記載されています。他のサービスのアクションのリストについては、そのサービスの API リファレンス [ドキュメント](#)を参照してください。

値は、サービス名前空間をアクションプレフィックス ( iam、ec2 sqs、sns、s3 など ) として使用し、許可または拒否するアクションの名前を付けて特定します。この名前は、サービスでサポートされているアクションと一致しなければいけません。プレフィックスとアクション名には、大文字と小文字の区別がありません。たとえば、iam:ListAccessKeys は IAM:listaccesskeys と同じです。下記の例は、様々サービスに対するAction要素の例を示します。

### Amazon SQS アクション

```
"Action": "sqs:SendMessage"
```

### Amazon EC2 アクション

```
"Action": "ec2:StartInstances"
```

### IAM アクション

```
"Action": "iam:ChangePassword"
```

### Amazon S3 のアクション

```
"Action": "s3:GetObject"
```

Action要素には複数の値を指定することができます。

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",  
            "iam:ChangePassword", "s3:GetObject" ]
```

特定の AWS 製品が提供するすべてのアクションへのアクセスを許可するには、ワイルドカード (\*) を使用することができます。たとえば、以下の Action 要素はすべての S3 アクションに適用します。

```
"Action": "s3:*"
```

また、アクション名の一部にワイルドカード (\*) を使用できます。たとえば、以下の Action 要素は、AccessKey、CreateAccessKey、DeleteAccessKey、ListAccessKeys などの文字列 UpdateAccessKey を含むすべての IAM アクションに適用されます。

```
"Action": "iam:*AccessKey*"
```

サービスの中には、使用可能なアクションに制限があるものがあります。たとえば、Amazon SQS では、使用可能なすべての Amazon SQS アクションのサブセットだけを使用することができます。この場合、\* ワイルドカードはキューの完全なコントロールを許可せず、共有しているアクションのサブセットだけが許可されます。詳細については、[Amazon Simple Storage Service 開発者ガイド](#)の「アクセス許可を理解する」を参照してください。

## IAM JSON ポリシー要素 NotAction

NotAction は、指定されたアクションリスト以外のすべてを明示的に照合する高度なポリシー要素です。NotAction を使うと、一致するアクションの長いリストではなく、いくつかの一致しないアクションのリストが含まれるため、ポリシーが短くなります。NotAction で指定したアクションは、ポリシーステートメントの Allow または Deny 効果の影響を受けません。これは、Allow 効果を使用する場合、リストされていない該当するすべてのアクションまたはサービスが許可されることを意味します。また、Deny 効果を使用する場合は、リストされていないそのようなアクションやサービスは定義されません。NotAction を Resource 要素と共に使用することで、ポリシーの範囲を指定します。これにより、AWS は該当するアクションまたはサービスを決定します。詳細については、次のポリシー例を参照してください。

### Allow での NotAction の使用

```
"Effect": "Allow" のステートメントで NotAction 要素を使用して、AWS サービス内で、NotAction で指定したアクションを除くすべてのアクションへのアクセスを許可できます。これを Resource 要素と共に使用してポリシーの範囲を提供し、許可されるアクションを、指定されたリソースで実行できるアクションに制限します。
```

次の例では、任意の S3 リソースで実行できる、バケットの削除以外のすべての Amazon S3 アクションにユーザーがアクセスすることを許可します。そのアクションでは "\*" リソースが必要である

ため、このポリシーは ListAllMyBuckets S3 API オペレーションの使用をユーザーに許可しません。また、このポリシーでは、他のサービスのアクションも許可されません。他のサービスのアクションは S3 リソースには適用されないためです。

```
"Effect": "Allow",  
"NotAction": "s3:DeleteBucket",  
"Resource": "arn:aws:s3:::*",
```

多数のアクションへのアクセスを許可することが必要になる場合があります。NotAction 要素を使用してそのステートメントを効果的に反転させることで、アクションのリストを短くすることができます。たとえば、AWS には多数のサービスがあるため、IAM へのアクセスを除くすべてのアクションの実行をユーザーに許可するポリシーを作成することが必要になる場合があります。

以下の例では、IAM を除くすべての AWS サービスですべてのアクションへのアクセスをユーザーに許可します。

```
"Effect": "Allow",  
"NotAction": "iam:*",  
"Resource": "*"
```

NotAction 要素と "Effect": "Allow" をポリシー内の同じステートメントで使用したり、別のステートメントで使用したりすることに注意してください。NotAction は、指定したリソースに対して明示的に列挙または適用されないすべてのサービスおよびアクションと一致するため、意図した以上のアクセス許可をユーザーに付与する結果になる場合があります。

## Deny での NotAction の使用

NotAction のステートメントで "Effect": "Deny" 要素を使用すると、NotAction 要素で指定されているアクションを除いて、リストされたすべてのリソースへのアクセスを拒否できます。この組み合わせでは、リストされた項目は許可されませんが、リストされていないアクションは明示的に拒否されます。許可したいアクションを許可する必要があります。

次の条件付きの例では、ユーザーが MFA の使用にサインインしていない場合、非 IAM アクションへのアクセスを拒否しています。ユーザーが MFA でサインインした場合は、"Condition" テストは失敗し、最終的な "Deny" ステートメントは無効になります。ただし、これにより、ユーザーがアクションにアクセスすることは許可されないため、IAM アクションを除くアクションはすべて明示的に拒否されます。

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [{
  "Sid": "DenyAllUsersNotUsingMFA",
  "Effect": "Deny",
  "NotAction": "iam:*",
  "Resource": "*",
  "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}}
}]
}
```

特定のサービスからのアクションを除き、特定のリージョン以外のアクションへのアクセスを拒否するポリシーの例については、「[AWS: リクエストされたリージョンに基づいて、AWS へのアクセスを拒否する](#)」を参照してください。

## IAM JSON ポリシー要素 Resource

Resource 要素は、ステートメントで取り扱う一連のオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ARN を使用して、リソースを特定します。ARN の形式についての詳細は、[IAM ARN](#) を参照してください。

各サービスには、それぞれ一連のリソースがあります。リソースを特定するためには必ず ARN を使用しますが、それぞれのリソースの ARN の詳細は、そのサービスおよびリソースによって異なります。リソースの指定方法についての情報は、お客様が記述しているステートメントを適用するリソースのサービスに関するドキュメントを参照してください。

### Note

一部のサービスでは、個々のリソースに対してアクションを指定することができず、代わりに Action または NotAction 要素でリストしたアクションがサービス内のすべてのリソースに適用されます。その場合、\* 要素内でワイルドカード (Resource) を使用してください。

以下の例は、特定の Amazon SQS キューを示しています。

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

以下の例は、AWS アカウント 内のボブという名の IAM ユーザーを示しています。

### Note

Resource 要素では、IAM ユーザー名の大文字と小文字が区別されます。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

## リソース ARN でのワイルドカードの使用

リソース ARN の一部にワイルドカードを使用することができます。ARN セグメント (コロンで区切られた部分) 内でワイルドカード文字 (\* と ?) を使用し、アスタリスク (\*) の付いた文字と疑問符 (?) が付いた任意の 1 文字の任意の組み合わせを表すことができます。複数の \* または ? 各セグメントの文字。ワイルドカード (\*) がリソース ARN セグメントの最後の文字である場合は、コロンの境界を越えて一致するように拡張できます。コロンで区切られた ARN セグメント内ではワイルドカード (\* と ?) を使用することをお勧めします。

### Note

AWS 製品を識別するためにサービスセグメントでワイルドカードを使用することはできません。ARN セグメントの詳細については、「[Amazon リソースネーム \(ARN\)](#)」を参照してください。

以下の例は、/accounting というパスを持つすべての IAM ユーザーを示しています。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```

以下の例は、特定の Amazon S3 バケット内のすべての項目を示しています。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

アスタリスク (\*) 文字を展開して、セグメント内のすべてを置き換えることができます。スラッシュ (/) などの文字は、特定のサービス名前空間内で区切り文字のように見える場合があります。たとえば、次のような Amazon S3 ARN を考えてみます。同じワイルドカード拡張ロジックがすべてのサービスに適用されるためです。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/test/*"
```

ARN のワイルドカードは、リストされた最初のオブジェクトだけでなく、バケット内の次のすべてのオブジェクトに適用されます。

```
DOC-EXAMPLE-BUCKET/1/test/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/test/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/test/3/object.jpg
```



```
DOC-EXAMPLE-BUCKET/1/2/3/test/4/object.jpg
DOC-EXAMPLE-BUCKET/1///test///object.jpg
DOC-EXAMPLE-BUCKET/1/test/.jpg
DOC-EXAMPLE-BUCKET//test/object.jpg
DOC-EXAMPLE-BUCKET/1/test/
```

前のリストの最後の 2 つのオブジェクトを考えてみましょう。Amazon S3 オブジェクト名は、従来の区切り文字のスラッシュ (/) 文字で開始または終了することができます。「/」は区切り文字として機能しますが、リソース ARN 内でこの文字を使用する場合は特に意味がありません。これは、他の有効な文字と同じように扱われます。ARN は次のオブジェクトと一致しません。

```
DOC-EXAMPLE-BUCKET/1-test/object.jpg
DOC-EXAMPLE-BUCKET/test/object.jpg
DOC-EXAMPLE-BUCKET/1/2/test.jpg
```

## 複数のリソースの指定

複数のリソースを特定することができます。以下の例は、2 つの DynamoDB テーブルを示しています。

```
"Resource": [
  "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",
  "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"
]
```

## リソース ARN でのポリシー変数の使用

Resource 要素では、特定のリソースを示す ARN の一部 (つまり、ARN の末尾部分) で JSON [ポリシー変数](#)を使用できます。たとえば、リソース ARN の一部としてキー {aws:username} を使用することで、現在のユーザー名をリソースの名前的一部分として含める必要があることを示すことができます。以下の例は、{aws:username} 要素内での Resource キーの使用方法を示します。ポリシーは、現在のユーザー名に一致する Amazon DynamoDB テーブルへのアクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:account-id:table/${aws:username}"
  }
}
```



JSON ポリシー変数の詳細については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

## IAM JSON ポリシー要素NotResource

NotResource は、指定されたリソースを除くすべてのリソースを明示的に照合する高度なポリシー要素です。NotResource を使うと、一致する予定のリソースのリストを含めるのではなく、一致する必要がないアクションがいくつかリストアップされ、リソースが短くなります。これは、単一の AWS のサービス内で適用されるポリシーで特に便利です。

たとえば、HRPayroll という名前のグループがあるとしましょう。HRPayroll のメンバーは、Payroll バケット内の HRBucket フォルダ以外のすべての Amazon S3 リソースにアクセスできません。次のポリシーは、リストされたリソース以外のすべての Amazon S3 リソースへのアクセスを明示的に拒否します。ただし、このポリシーは、すべてのリソースにユーザーのアクセス権を与えるものではないことに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "s3:*",
    "NotResource": [
      "arn:aws:s3:::HRBucket/Payroll",
      "arn:aws:s3:::HRBucket/Payroll/*"
    ]
  }
}
```

通常、リソースへのアクセスを明示的に拒否するには、「Effect」:"Deny" を使用し、各フォルダを個別にリストする Resource 要素を含むポリシーを作成します。ただし、その場合には、HRBucket にフォルダを追加したり、アクセスすべきでない Amazon S3 にリソースを追加するたびに、Resource のリストにその名前を追加する必要があります。代わりに NotResource 要素を使用すると、ユーザーはフォルダ名を NotResource 要素に追加しない限り、新しいフォルダへのアクセスは自動的に拒否されます。

NotResource を使用する場合は、この要素に指定されているリソースは制限されていないリソースのみであることに注意してください。これにより、アクションに適用されるすべてのリソースが制限されます。上記の例では、ポリシーは Amazon S3 アクションにのみ影響し、そのため、Amazon S3 リソースにのみ影響します。アクションに Amazon EC2 アクションも含まれている場合、ポリシーは EC2 リソースへのアクセスを拒否しません。サービス内のどのアクションでリソースの ARN

を指定できるかについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## NotResource とその他の要素

"Effect": "Allow"、"Action": "\*"、および "NotResource":

"arn:aws:s3:::HRBucket" 要素と一緒に使用しないでください。このステートメントは、HRBucket S3 バケットを除くすべてのリソースで AWS のすべてのアクションを許可するため、非常に危険です。これにより、ユーザーは自身に HRBucket へのアクセスを許可するポリシーを追加することもできます。この操作はしないでください。

NotResource 要素と "Effect": "Allow" をポリシー内の同じステートメントで使用したり、別のステートメントで使用したりすることに注意してください。NotResource は、明示的に列挙されないすべてのサービスおよびリソースを許可するため、意図した以上のアクセス許可をユーザーに付与する結果になる場合があります。同じステートメントで NotResource 要素と "Effect": "Deny" を使用すると、明示的にリストされていないサービスとリソースが拒否されます。

## IAM JSON ポリシー要素 Condition

Condition 要素 (または Condition block) は、ポリシーを実行するタイミングの条件を指定することができます。Condition 要素はオプションです。Condition 要素に、[条件演算子](#) (等しい、より小さい、など) を使用して、ポリシーのコンテキストキーバリューをリクエストコンテキストのキーバリューに一致させる式を構築します。リクエストコンテキストの詳細については、「[リクエスト](#)」を参照してください。

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

ポリシー条件で指定できるコンテキストキーは、[グローバル条件コンテキストキー](#)またはサービス固有のコンテキストキーです。グローバル条件コンテキストキーには、aws: というプレフィックスが付いています。サービス固有のコンテキストキーには、サービスのプレフィックスがあります。例えば Amazon EC2 では、ec2:InstanceType コンテキストキーを使用して、そのサービスに固有の条件を記述できます。iam: プレフィックスが付いたサービス固有の IAM コンテキストキーを表示するには、「[IAM および AWS STS の条件コンテキストキー](#)」を参照してください。

コンテキストキーでは、名前の大文字と小文字が区別されません。例えば、aws:SourceIP コンテキストキーを含めることは、AWS:SourceIp をテストすることと同じです。コンテキストキーの値の大文字と小文字の区別は、使用する[条件演算子](#)によって異なります。例えば、次の条件には、johndoe によって行われたリクエストのみが一致するようにする StringEquals 演算子が含まれています。JohnDoe という名前のユーザーはアクセスを拒否されます。

```
"Condition" : { "StringEquals" : { "aws:username" : "johndoe" } }
```

次の条件では、[StringEqualsIgnoreCase](#) 演算子を使用して、johndoe または JohnDoe という名前のユーザーに一致させます。

```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "johndoe" } }
```

一部のコンテキストキーでは、キー名の一部を指定することができるキーバリューのペアをサポートしています。この例には、[aws:RequestTag/tag-key](#) コンテキストキー、AWS KMS [kms:EncryptionContext:encryption\\_context\\_key](#)、および複数のサービスでサポートされている [ResourceTag/tag-key](#) コンテキストキーが含まれます。

- [Amazon EC2](#) などのサービスに [ResourceTag/tag-key](#) コンテキストキーを使用する場合は、tag-key のキー名を指定する必要があります。
- キー名では大文字と小文字が区別されません。つまり、ポリシーの条件要素で "aws:ResourceTag/TagKey1": "Value1" で指定した場合、その条件は TagKey1 または tagkey1 という名前のリソースタグキーに一致しますが、その両方には一致しません。
- これらの属性をサポートする AWS サービスでは、大文字と小文字だけが異なる複数のキー名を作成することができる場合があります。例えば、ec2=test1 および EC2=test2 を使用して Amazon EC2 インスタンスにタグ付けします。"aws:ResourceTag/EC2": "test1" などの条件を使用して、そのリソースへのアクセスを許可すると、キー名は両方のタグと一致しますが、1つの値のみが一致します。これにより、予期しない障害が発生することがあります。

#### Important

ベストプラクティスとして、キーバリューのペア属性に名前を付けるときは、アカウントのメンバーが一貫した命名規則に従うようにします。例としては、AWS KMS タグや暗号化コンテキストなどがあります。これを強制するには、タグ付けに [aws:TagKeys](#) コンテキストキーを使用するか、AWS KMS 暗号化コンテキストに [kms:EncryptionContextKeys](#) を使用します。

- 条件演算子の一覧と演算子の動作の説明については、「[条件演算子](#)」を参照してください。
- 他に特定の無い限り、すべてのコンテキストキーには複数の値を含むことができます。複数の値を持つコンテキストキーを処理する方法の説明については、「[複数値のコンテキストキー](#)」を参照してください。

- グローバルに利用できるコンテキストキーのすべてのリストについては、[AWS グローバル条件コンテキストキー](#) を参照してください。
- 各サービスで定義されるコンテキストキーについては、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。

## リクエストのコンテキスト

[プリンシパル](#)が AWS に [リクエスト](#) を行うと、AWS はリクエスト情報をリクエストコンテキストに収集します。この情報は、リクエストの評価と承認に使用されます。JSON ポリシーの Condition 要素を使用して、リクエストコンテキストに対して特定のコンテキストキーをテストできます。例えば、[aws:CurrentTime](#) コンテキストキーを使用するポリシーを作成して、[特定の日付範囲内でのみアクションの実行をユーザーに許可できます](#)。

リクエストが送信されると、AWS はポリシーの各コンテキストキーを評価し、true、false、not present、場合によっては null (空のデータ文字列) の値を返します。リクエストに存在しないコンテキストキーは、不一致と見なされます。例えば、次のポリシーでは、過去 1 時間 (3,600 秒) に MFA を使用してサインインした場合に限り、独自の多要素認証 (MFA) デバイスの削除を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AllowRemoveMfaOnlyIfRecentMfa",
    "Effect": "Allow",
    "Action": [
      "iam:DeactivateMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}",
    "Condition": {
      "NumericLessThanEquals": {"aws:MultiFactorAuthAge": "3600"}
    }
  }
}
```

リクエストコンテキストは次の値を返すことができます。

- True – リクエストが過去 1 時間以内に MFA を使用してサインインした場合、条件は true を返します。
- False – リクエストが MFA を使用して 1 時間以上前にサインインした場合、条件は false を返します。

- Not present – AWS CLI リクエストがまたは AWS API の IAM ユーザーアクセスキーを使用してリクエストを行った場合、キーは存在しません。この場合、キーは存在せず、一致しません。
- Null – リクエストでタグを渡すなど、ユーザーが定義したコンテキストキーの場合、空の文字列を含めることができます。この場合、リクエストコンテキストの値はnullです。null 値は、場合によっては true を返すことがあります。例えば、[aws:TagKeys](#) コンテキストキーで複数値 [ForAllValues](#) 条件演算子を使用する場合、リクエストコンテキストが null を返すと、予期しない結果が発生する可能性があります。詳細については、「[aws:TagKeys](#)」および「[複数値のコンテキストキー](#)」を参照してください。

## 条件ブロック

以下の例は、Condition要素の基本フォーマットを示します。

```
"Condition": {"StringLike": {"s3:prefix": ["janedoe/*"]}}
```

リクエストからの値は、コンテキストキーによって表現されます。この場合は s3:prefix です。コンテキストキーバリューは、janedoe/\* などのリテラル値として指定した値と比較されます。比較の種類は、[条件演算子](#)によって指定されます (ここでは StringLike)。等号、大なり記号、小なり記号といった一般的なブール演算子を使用して、文字列、日付、数値などを比較する条件を作成できます。[文字列演算子](#)または [ARN 演算子](#)を使用する場合は、コンテキストキーの値に[ポリシー変数](#)を使用することもできます。次の例では、aws:username 変数が含まれています。

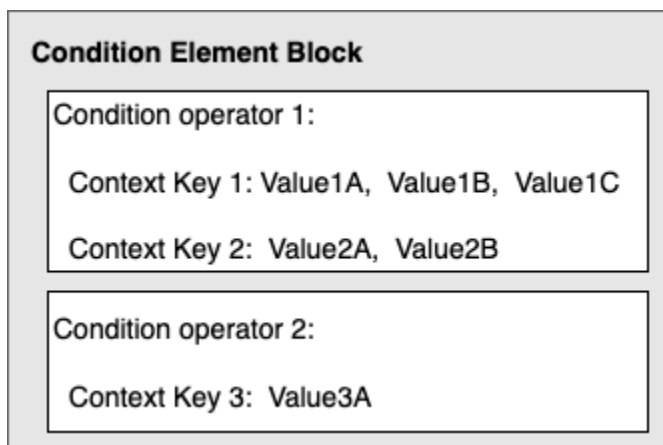
```
"Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
```

一部の環境では、コンテキストキーに複数の値が含まれる可能性があります。たとえば、Amazon DynamoDB へのリクエストによって、テーブルの複数の属性を返すまたは更新することが要求される場合があります。DynamoDB テーブルへのアクセスのポリシーには、リクエスト内のすべての属性を含む dynamodb:Attributes コンテキストキーを追加できます。Condition 要素で設定演算子を使用することで、リクエスト内のすべての属性を、ポリシー内の許可された属性のリストと照合できます。詳細については、「[複数値のコンテキストキー](#)」を参照してください。

リクエスト中にポリシーが評価される際、AWS はキーをリクエストからの対応する値に置き換えます。(この例では、AWS はリクエストの日時を使用します。)条件が評価された上で「true (真)」または「false (偽)」が返され、それを考慮に入れてポリシー全体がリクエストを許可または拒否します。

## 条件内の複数の値

Condition 要素は複数の条件演算子を含むことができ、各条件演算子は複数のキーと値のペアを含むことができます。以下の図が解説したものです。



詳細については、「[複数値のコンテキストキー](#)」を参照してください。

### IAM JSON ポリシー要素: 条件演算子

Condition 要素で条件演算子を使用して、ポリシーの条件キーバリューをリクエストコンテキストの値と一致させます。Condition 要素の詳細については、「[IAM JSON ポリシー要素Condition](#)」を参照してください。

ポリシーで使用できる条件演算子は、選択する条件キーによって異なります。グローバル条件キーまたはサービス固有の条件キーを選択できます。グローバル条件キーに使用できる条件演算子については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。サービスのサービス固有の条件キーを表示するには、「[AWS サービスのアクション、リソース、および条件キー](#)」を参照し、キーを表示するサービスを選択します。

#### **⚠ Important**

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致せず条件は false になります。StringNotLike または ArnNotLike などのように、キーの一致を必要としないポリシー条件であり、かつ正しいキーが存在していない場合、条件は true となります。このロジックは、[...IfExists](#) および [Null check](#) を除くすべての条件演算子に適用されます。これらの演算子は、キーがリクエストコンテキストにある（存在する）かどうかをテストします。

条件演算子は次のカテゴリに分類できます。


- [文字列](#)
- [数値](#)
- [日付および時間](#)
- [ブール値](#)
- [バイナリ](#)
- [IP アドレス](#)
- [Amazon リソースネーム \(ARN\)](#) (一部のサービスでのみ使用可能)
- [...IfExists](#) (別のチェックの一部としてキーバリューが存在するかを確認)
- [Null check](#) (スタンドアロンチェックとしてキーが存在するかを確認)

## 文字列条件演算子

文字列条件演算子では、キーと文字列値の比較に基づいてアクセスを制限する Condition 要素を構築できます。

条件演算子	説明
StringEquals	完全一致、大文字と小文字の区別あり。
StringNotEquals	符号反転の一致
StringEqualsIgnoreCase	完全一致、大文字と小文字の区別なし。
StringNotEqualsIgnoreCase	符号反転の一致、大文字と小文字の区別なし。
StringLike	大文字と小文字の区別がある一致。値には、複数文字一致のワイルドカード (*) および 1 文字一致のワイルドカード (?) を文字列のどこにでも含めることができます。文字列の部分一致検索を行うには、ワイルドカードを指定する必要があります。



条件演算子	説明
	<p> <b>Note</b></p> <p>キーに複数の値が含まれる場合、設定演算子 (StringLike および ForAllValues:StringLike ) を使用して ForAnyValue:StringLike を修飾できます。詳細については、「<a href="#">複数値のコンテキストキー</a>」を参照してください。</p>
StringNotLike	<p>符号反転の一致には、大文字と小文字の区別があります。値には、複数文字一致のワイルドカード (*) または 1 文字一致のワイルドカード (?) を文字列のどこにでも含めることができます。</p>

たとえば、次のステートメントに含まれている Condition 要素では、[aws:PrincipalTag](#) キーの使用により、リクエストを行うプリンシパルに iamuser-admin ジョブカテゴリのタグ付けが必要であることを指定しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/job-category": "iamuser-admin"}}
  }
}
```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。この例では、タグがアタッチされた IAM ユーザーをプリンシパルが使用している場合、aws:PrincipalTag/job-category キーがリクエストコンテキストに存在します。これは、タグまたはセッションタグがアタッチされた IAM ロールを使用するプリンシパルのために含まれます。タグがないユーザーがアクセスキーを表示または編集しようとする、条件により false が返され、リクエストはこのステートメントによって暗黙的に拒否されます。

次を使用できます。[ポリシー変数](#)とString条件演算子を使用します。



以下の例では、StringLike 条件演算子を使用して[ポリシー変数](#)による文字列一致を実行して、IAM ユーザーが Amazon S3 コンソールを使用して Amazon S3 バケット内の自らの「ホームディレクトリ」を管理できるようにするポリシーを作成します。このポリシーは、s3:prefix が指定されたパターンのいずれかに一致する限り、指定されたアクションを S3 バケットに対して実行することを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

OIDC フェデレーションによるアプリケーション ID とユーザー ID に基づいてリソースへのアクセスを制限する Condition 要素の使用法を示すポリシーの例については、「[Amazon S3: Amazon Cognito ユーザーにバケット内のオブジェクトへのアクセスを許可する](#)」を参照してください。

## ワイルドカードによる一致

文字列条件演算子は、事前に定義された形式を適用しないパターンレスマッチングを実行します。ARN 条件演算子と日付条件演算子は、条件キー値に構造体を適用する文字列演算子のサブセットです。ARN または日付の文字列の部分一致に StringLike または StringNotLike 演算子を使用すると、マッチングではワイルドカードで指定されている構造体の部分が無視されます。

例えば、次の条件では、さまざまな条件演算子を使用して ARN の部分一致を検索します。

ArnLike を使用する場合、ARN のパーティション、サービス、アカウント ID、リソースタイプ、および部分的なリソース ID 部分が、リクエストコンテキストの ARN と完全に一致する必要があります。部分一致が許可されるのは、リージョンとリソースパスのみです。

```
"Condition": {"ArnLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/*"}}
```

ArnLike の代わりに StringLike を使用すると、マッチングでは ARN 構造が無視され、ワイルドカードで指定された部分に関係なく部分一致が可能になります。

```
"Condition": {"StringLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/*"}}
```

ARN	ArnLike	StringLike
arn:aws:cloudtrail:us-west-2:111122223333:trail/finance	一致	一致
arn:aws:cloudtrail:us-east-2:111122223333:trail/finance/archive	一致	一致
arn:aws:cloudtrail:us-east-2:444455556666:user/111122223333:trail/finance	一致なし	一致

## 数値条件演算子

数値条件演算子では、キーと整数または 10 進値の比較に基づいてアクセスを制限する Condition 要素を構築できます。

条件演算子	説明
NumericEquals	一致
NumericNotEquals	符号反転の一致
NumericLessThan	「未満」の部分一致
NumericLessThanEquals	「未満と等しい」の部分一致
NumericGreaterThan	「上回る」の部分一致
NumericGreaterThanEquals	「上回るまたは等しい」の部分一致

たとえば、以下のステートメントに含まれる Condition 要素は、NumericLessThanEquals 条件演算子を s3:max-keys キーと合わせて使用して、リクエスタが の中で一度に最大 example\_bucket 10 個のオブジェクトを列挙できることを指定しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket",
    "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
  }
}
```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。この例では、ListBucket オペレーションを実行すると、s3:max-keys キーは常にリクエストに存在します。このポリシーですべての Amazon S3 オペレーションが許可されている場合、10 以下の値を持つ max-keys コンテキストキーを含むオペレーションのみが許可されます。

[ポリシー変数](#)とNumeric条件演算子を使用することはできません。

## 日付条件演算子

日付条件演算子では、キーと日付/時刻値の比較に基づいてアクセスを制限する Condition 要素を構築できます。これらの条件演算子は、[aws:CurrentTime](#) キーまたは [aws:EpochTime](#) キーと合わせて使用します。日付/時間値と共に、[W3C implementations of the ISO 8601 date formats](#) またはエポック (UNIX) 時間のどれか 1 つを特定しなければいけません。

### Note

ワイルドカードは日付条件演算子では使用できません。

条件演算子	説明
DateEquals	特定の日付との一致
DateNotEquals	符号反転の一致
DateLessThan	特定の日時よりも前の日時との一致
DateLessThanEquals	特定の日時またはそれよりも前の日時との一致
DateGreaterThan	特定の日時よりも後の日時との一致
DateGreaterThanEquals	特定の日時またはそれよりも後の日時との一致

たとえば、次のステートメントには、DateGreaterThan 条件演算子を [aws:TokenIssueTime](#) キーとともに使用する Condition 要素が含まれています。この条件は、リクエストの作成に使用された一時的なセキュリティ認証情報が 2020 年に発行されたことを示します。このポリシーは、毎日プログラムによって更新され、アカウントメンバーが最新の資格情報を使用するようにできます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
```

```

    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"DateGreaterThan": {"aws:TokenIssueTime": "2020-01-01T00:00:01Z"}}
  }
}

```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。プリンシパルがリクエストを行うために一時的な認証情報を使用する場合に限り、リクエストコンテキストで `aws:TokenIssueTime` キーが表示されます。このキーは、アクセスキーを使用して行われた AWS CLI、AWS API、または AWS SDK リクエストには存在しません。この例では、IAM ユーザーがアクセスキーを表示または編集しようとする、リクエストは拒否されます。

Date 条件演算子で [ポリシー変数](#) を使用することはできません。

## ブール条件演算子

ブール条件演算子では、キーと "true (真)" または "false (偽)" の比較に基づいてアクセスを制限する Condition 要素を構築できます。

条件演算子	説明
Bool	ブールの一致

例えば、この ID ベースのポリシーでは [aws:SecureTransport](#) キーを持つ Bool 条件演算子が使用されるため、リクエストが SSL 経由でない場合に、オブジェクトとオブジェクトタグを送信先バケットとそのコンテンツに複製することを拒否できます。

### Important

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BooleanExample",
      "Action": "s3:ReplicateObject",
      "Effect": "Deny",

```

```
"Resource": [
  "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
  "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
],
"Condition": {
  "Bool": {
    "aws:SecureTransport": "false"
  }
}
]
```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。aws:SecureTransport リクエストコンテキストは true または false を返します。

次を使用できます。[ポリシー変数](#)と Boolean 条件演算子を使用します。

### バイナリ条件演算子

BinaryEquals 条件演算子では、バイナリ形式のキーバリューをテストする Condition 要素を構築できます。これは、指定されたキーの値を、ポリシー内の値を [base-64](#) エンコードした表現に対してバイト単位で比較します。

```
"Condition" : {
  "BinaryEquals": {
    "key" : "Qm1uYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。

[条件演算子](#)で Binary ポリシー変数を使用することはできません。

### IP アドレス条件演算子

IP アドレス条件演算子では、キーと IPv4 または IPv6 アドレスまたは IP アドレス範囲の比較に基づいてアクセスを制限する Condition 要素を構築できます。これらを [aws:SourceIp](#) キーと合わせて使用します。値は、標準的な CIDR 形式でなければいけません (例: 203.0.113.0/24 または 2001:DB8:1234:5678::/64)。IP アドレスの指定時に関連付けられたルーティングプレフィックスを使用しないと、IAM ではデフォルトのプレフィックス値 /32 を使用します。

IPv6 をサポートしている AWS のサービスでは、0 の範囲を :: で表します。サービスで IPv6 がサポートされているかどうかは、そのサービスのドキュメントを参照してください。

条件演算子	説明
IpAddress	所定の IP アドレスまたは範囲
NotIpAddress	所定の IP アドレスまたは範囲以外のすべての IP アドレス

たとえば、以下のステートメントでは、IpAddress 条件を aws:SourceIp キーと合わせて使用して、リクエストが 203.0.113.0 から 203.0.113.255 までの IP 範囲から送られてこなければいけないことを指定しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"IpAddress": {"aws:SourceIp": "203.0.113.0/24"}}
  }
}
```

aws:SourceIp 条件キーは、リクエストの送信元である IP アドレスに解決します。リクエストが Amazon EC2 インスタンスから発信された場合、aws:SourceIp はインスタンスのパブリック IP アドレスに評価されます。

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。aws:SourceIp キーは、リクエストが VPC エンドポイントを使用してリクエストを行う場合を除き、リクエストコンテキストに常に表示されます。この場合、条件は false を返し、リクエストはこのステートメントによって暗黙的に拒否されます。

IpAddress 条件演算子で [ポリシー変数](#) を使用することはできません。

次の例では、組織内の有効な IP アドレスすべてを含めるために、IPv4 と IPv6 アドレスを混在させる方法を示しています。IPv6 への移行に合わせてポリシーが引き続き機能することを確認するため、すでにある IPv4 の範囲に追加する IPv6 アドレスの範囲の組織のポリシーを更新することをお勧めします。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
}
```

aws:SourceIp 条件キーは、テストされた API をユーザーとして直接呼び出す場合に JSON ポリシーでのみ機能します。代わりにサービスを使用してターゲットサービスを呼び出した場合、ターゲットサービスは元のユーザーの IP アドレスではなく呼び出し元サービスの IP アドレスを認識します。これは、AWS CloudFormation を使用して Amazon EC2 を呼び出すことでインスタンスを自動的に作成した場合などに生じることがあります。現在のところ、JSON ポリシーで評価を行うために、発信元サービスを通じて元の IP アドレスをターゲットサービスに渡す方法はありません。これらのタイプのサービス API 呼び出しでは、aws:SourceIp 条件キーを使用しないでください。

### Amazon リソースネーム (ARN) の条件演算子

Amazon Resource Name (ARN) 条件演算子では、キーと ARN の比較に基づいてアクセスを制限する Condition 要素を構築できます。ARN は文字列として見なされます。

条件演算子	説明
ArnEquals , ArnLike	ARN の大文字と小文字を区別した一致。ARN のコロンで分割された 6 個の各構成要素は個別に確認され、それぞれ複数文字一致のワイルドカード (*) または 1 文字一致のワイルドカード (?) を含むことができます。ArnEquals および ArnLike 条件演算子は、同じように動作します。
ArnNotEquals , ArnNotLike	ARN の符号反転の一致。ArnNotEquals および ArnNotLike 条件演算子は、同じように動作します。



次を使用できます。[ポリシー変数](#)と ARN 条件演算子を使用します。

次のリソースベースのポリシーの例は、SNS メッセージの送信先となる Amazon SQS キューにアタッチされたポリシーを示しています。この例では、サービスが 1 つまたは複数の特定の Amazon SNS トピックのためにメッセージを送る場合に限り、1 つまたは複数のキューにメッセージを送る Amazon SNS 許可を付与しています。Resource フィールドのキューを指定し、SourceArn キーの値として Amazon SNS トピックを指定します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "123456789012"},
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:REGION:123456789012:QUEUE-ID",
    "Condition": {"ArnEquals": {"aws:SourceArn":
"arn:aws:sns:REGION:123456789012:TOPIC-ID"}}
  }
}
```

ポリシー条件で指定したキーがリクエストコンテキストに存在しない場合、値は一致しません。[aws:SourceArn](#) キーは、リソース所有者に代わって別のサービス呼び出すようにリソースがトリガーした場合にのみ、リクエストコンテキストに表示されます。IAM ユーザーがこのオペレーションを直接実行しようとする、条件によ false が返され、リクエストはこのステートメントによって暗黙的に拒否されます。

### IfExists 条件演算子

Null 条件 (StringLikeIfExists など) を除く任意の条件演算子名の末尾に IfExists を追加できます。「ポリシーキーがリクエストのコンテキストで存在する場合、ポリシーで指定されたとおりにキーを処理します。キーが存在しない場合、条件要素は true と評価されます。」...IfExists でチェックすると、ステートメント内の別の Condition 要素は一致なしの結果となることもありますが、キーが見つからないことはありません。StringNotEqualsIfExists のような否定条件演算子を持つ "Effect": "Deny" 要素を使用している場合は、タグがなくとも要求が拒否されます。

### IfExists の使用例

多くの条件キーは特定のタイプのリソースに関する情報を示し、そのタイプのリソースにアクセスしている場合にのみ存在します。これらの条件キーはその他のタイプのリソースにはありません。ポリシーステートメントが 1 種類のリソースのみに適用される場合には、これで問題はありません。ところが、ポリシーステートメントが複数のサービスからアクションを参照する場合や、サービス内の

特定のアクションが同じサービス内の異なるタイプのリソースにアクセスする場合などのように、1つのステートメントが複数のタイプのリソースに適用される場合があります。このような場合、ポリシーステートメント内の1つのリソースのみに適用される条件キーを含めると、ポリシーステートメントの Condition 要素が失敗し、ステートメントの "Effect" は適用されません。

たとえば、次のポリシーの例を考えてみます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "THISPOLICYDOESNOTWORK",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

前述のポリシーの目的は、ユーザーが t1、t2 および m3 タイプのインスタンスを起動できるようにすることです。ところが、インスタンスを起動する場合には、インスタンス自体に加えて、イメージ、キーペア、セキュリティグループおよびそれ以上のさまざまなリソースにアクセスする必要があります。ステートメント全体が、インスタンスを起動するために必要なすべてのリソースに対して評価されます。これらの追加のリソースには ec2:InstanceType 条件キーがないため、StringLike のチェックは失敗し、ユーザーはいずれのタイプのインスタンスも起動できません。

これに対応するには、StringLikeIfExists 条件演算子を代わりに使用します。そうすれば、条件キーが存在する場合のみにテストが行われます。以下のコードは次のように解釈できます。「チェックされるリソースには「ec2:InstanceType」条件キーがあり、キーバリューが t1.、t2.、または m3. で始まる場合にのみ、アクションを許可します。チェックされるリソースにこの条件キーがなくても問題ありません。」条件キー値のアスタリスク (\*) を StringLikeIfExists 条件演算子と併用すると、ワイルドカードとして解釈され、文字列の一部が一致します。この DescribeActions 文には、コンソールでインスタンスを表示するために必要なアクションが含まれます。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "RunInstance",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {
      "StringLikeIfExists": {
        "ec2:InstanceType": [
          "t1.*",
          "t2.*",
          "m3.*"
        ]
      }
    }
  },
  {
    "Sid": "DescribeActions",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages",
      "ec2:DescribeInstances",
      "ec2:DescribeVpcs",
      "ec2:DescribeKeyPairs",
      "ec2:DescribeSubnets",
      "ec2:DescribeSecurityGroups"
    ],
    "Resource": "*"
  }
]
```

### 条件キーの有無をチェックする条件演算子

Null 条件演算子を使用して、認証時に条件キーが存在していないかどうかを確認します。ポリシーステートメントで、true ( キーは存在しません-nullです ) または false ( キーが存在し、その値は nullではありません ) を使用します。

Null 条件演算子で[ポリシー変数](#)を使用することはできません。

たとえば、ユーザーが操作または一時的な認証情報に独自の認証情報を使用しているかどうかを判断するために、この条件演算子を使用することができます。ユーザーが一時的な認証情報を使用している場合、キー `aws:TokenIssueTime` が存在し、このキーには値があります。以下の例の場合、ユーザーが Amazon EC2 API を使うために一時的な認証情報を使用することはできない ( キーは存在してはならない ) という条件を示しています。

```
{
  "Version": "2012-10-17",
  "Statement":{
    "Action":"ec2:*",
    "Effect":"Allow",
    "Resource":"*",
    "Condition":{"Null":{"aws:TokenIssueTime":"true"}}
  }
}
```

## 複数のコンテキストキーまたは値による条件

ポリシーの Condition 要素を使用して、リクエスト内の 1 つのコンテキストキーに対して複数のコンテキストキーまたは値をテストできます。プログラムまたは AWS を通じて AWS Management Console にリクエストを行うと、リクエストにはプリンシパル、オペレーション、タグなどに関する情報が含まれます。コンテキストキーを使用すると、ポリシー条件で指定したコンテキストキーで、リクエスト内の一致するコンテキストキーの値をテストできます。リクエストに含まれる情報とデータについては、「[リクエストのコンテキスト](#)」を参照してください。

### トピック

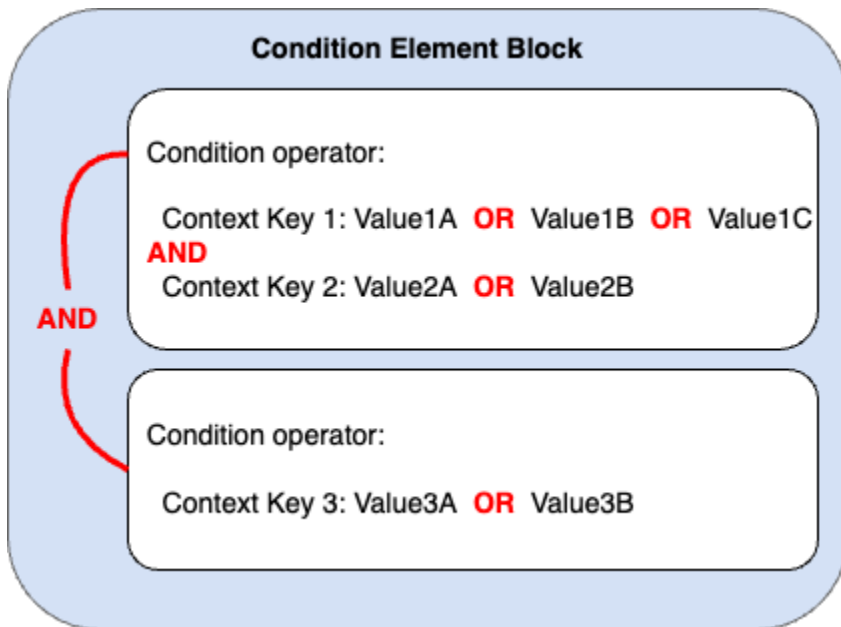
- [複数のキーまたは値の評価ロジック](#)
- [否定された一致条件演算子の評価ロジック](#)

## 複数のキーまたは値の評価ロジック

Condition 要素は複数の条件演算子を含むことができ、各条件演算子は複数のキーと値のペアを含むことができます。特に指定のない限り、ほとんどのコンテキストキーで複数の値を使用できます。

- ポリシーステートメントに複数の[条件演算子](#)がある場合、条件演算子は論理 AND を使用して評価されます。
- 1 つの条件演算子にアタッチされている複数のコンテキストキーがポリシーステートメントにある場合、コンテキストキーは論理 AND を使用して評価されます。
- 1 つの条件演算子に 1 つのコンテキストキーの値が複数含まれる場合、それらの値は論理 OR を使用して評価されます。
- 1 つの否定された一致条件演算子に 1 つのコンテキストキーの値が複数含まれる場合、それらの値は論理 NOR を使用して評価されます。

Condition 要素のブロックのすべてのコンテキストキーを true に変換して、目的の Allow または Deny 効果を呼び出す必要があります。次の図は、複数の条件演算子およびコンテキストのキーと値のペアを含む条件の評価ロジックを示しています。



例えば、次の S3 バケットポリシーは、上記の図がポリシーでどのように表されるかを示しています。条件ブロックは、条件演算子 StringEquals および ArnLike、コンテキストキー aws:PrincipalTag および aws:PrincipalArn を含んでいます。目的の Allow または Deny 効果を呼び出すには、条件ブロックのすべてのコンテキストキーが true に変換される必要があります。リクエストを行うユーザーには、ポリシーで指定されているタグキー値のいずれかを含む、department と role の両方のプリンシパルタグキーが必要です。また、リクエストを行うユーザーのプリンシパル ARN は、true と評価されるポリシーで指定される aws:PrincipalArn の値のいずれかに一致する必要があります。

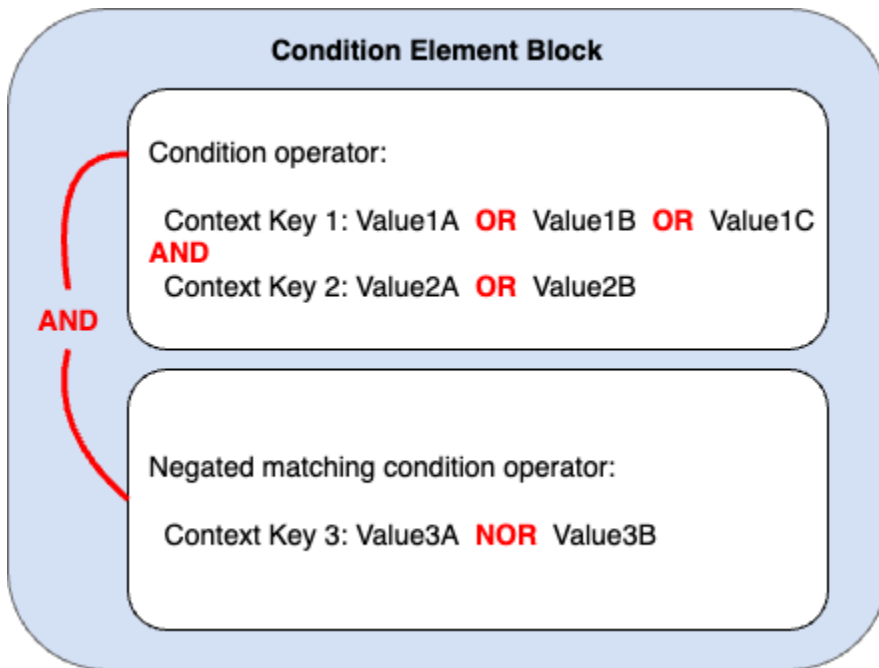
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
```

```
    "aws:PrincipalTag/department": [
      "finance",
      "hr",
      "legal"
    ],
    "aws:PrincipalTag/role": [
      "audit",
      "security"
    ]
  },
  "ArnLike": {
    "aws:PrincipalArn": [
      "arn:aws:iam::222222222222:user/Ana",
      "arn:aws:iam::222222222222:user/Mary"
    ]
  }
}
```

### 否定された一致条件演算子の評価ロジック

`StringNotEquals` や `ArnNotLike` などの一部の[条件演算子](#)は、否定された一致を使用して、ポリシー内のコンテキストのキーと値のペアをリクエスト内のそれと比較します。否定された一致条件演算子を使用するポリシーで、1つのコンテキストキーに複数の値が指定されている場合、有効なアクセス許可は論理 NOR であるかのように機能します。否定された一致では、論理 NOR または NOT OR はすべての値が `false` と評価された場合のみ `true` を返します。

次の図は、複数の条件演算子およびコンテキストのキーと値のペアを含む条件の評価ロジックを示しています。この図には、コンテキストキー 3 用の否定された一致条件演算子が含まれています。



例えば、次の S3 バケットポリシーは、上記の図がポリシーでどのように表されるかを示しています。条件ブロックは、条件演算子 `StringEquals` および `ArnNotLike`、コンテキストキー `aws:PrincipalTag` および `aws:PrincipalArn` を含んでいます。目的の Allow または Deny 効果を呼び出すには、条件ブロックのすべてのコンテキストキーが true に変換される必要があります。リクエストを行うユーザーには、ポリシーで指定されているタグキー値のいずれかを含む、`department` と `role` の両方のプリンシパルタグキーが必要です。`ArnNotLike` 条件演算子で否定された一致を使用するため、リクエストを行うユーザーのプリンシパル ARN は true と評価されるポリシーで指定される `aws:PrincipalArn` の値のいずれとも一致してはいけません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",

```

```
        "hr",
        "legal"
    ],
    "aws:PrincipalTag/role": [
        "audit",
        "security"
    ]
},
"ArnNotLike": {
    "aws:PrincipalArn": [
        "arn:aws:iam::222222222222:user/Ana",
        "arn:aws:iam::222222222222:user/Mary"
    ]
}
}
}
]
```

## 単一値と複数値のコンテキストキー

単一値と複数値のコンテキストキーの違いは、ポリシー条件の値の数ではなく、[リクエストコンテキスト](#)の値の数によります。

- 単一値の条件コンテキストキーはリクエストコンテキストに、最大で1つの値を持ちます。たとえば、AWSでリソースにタグを付けることができます。リソースタグは、タグキー値のペアとして格納されます。リソースタグキーは単一のタグ値を持つことができます。したがって、[the section called "ResourceTag"](#) は単一値のコンテキストキーです。単一値のコンテキストキーで条件集合演算子を使用しないでください。
- 複数値の条件コンテキストキーは、リクエストコンテキストに複数の値を持つことができます。たとえば、AWS内のリソースにタグを付けることができ、リクエストに複数のタグキーと値のペアを含めることができます。したがって、[the section called "TagKeys"](#) は複数の値を持つコンテキストキーです。複数値のコンテキストキーには条件集合演算子が必要です。

### Important

複数値のコンテキストキーには条件集合演算子が必要です。単一値のコンテキストキーで、条件集合演算子の `ForAllValues` または `ForAnyValue` を使用しないでください。条件集合演算子の詳細については、「[複数値のコンテキストキー](#)」を参照してください。



単一値と複数値の分類については、条件コンテキストキーの値の種類として、[AWS グローバル条件コンテキストキー](#) トピックでそれぞれ説明されています。[サービス認証リファレンス](#)では、異なる方法で複数値のコンテキストキーを分類しています。ArrayOfプレフィックスのあとに、条件演算子のカテゴリタイプが続くという形式です。例えば、ArrayOfString、ArrayOfARN などです。

例えば、リクエストは最大でも1つのVPCエンドポイントから発信されるため、[the section called "SourceVpce"](#) は単一値のコンテキストキーです。サービスには、そのサービスに属するサービスプリンシパル名を複数持つことができるため、[aws:PrincipalServiceNamesList](#) は複数値のコンテキストキーです。

どの単一値のコンテキストキーでも、ポリシー変数として使用できます。複数値のコンテキストキーは、ポリシー変数としては使用できません。ポリシー変数の詳細については、「[IAM ポリシーの要素: 変数とタグ](#)」を参照してください。

複数値のコンテキストキーは、条件集合演算子 ForAllValues または ForAnyValue が必要です。[the section called "RequestTag"](#) や [the section called "ResourceTag"](#) のようなキーバリューペアを含むコンテキストキーは複数の *tag-key* 値を持つため、混乱を招く可能性があります。しかし、それぞれの *tag-key* が持つことのできる値は1つであるため、aws:RequestTag とaws:ResourceTag はどちらも単一値のコンテキストキーです。単一値のコンテキストキーで条件集合演算子を使用すると、過度にポリシーが許容される可能性があります。

## 複数値のコンテキストキー

条件コンテキストキーを、複数の値を持つ[リクエストコンテキストキー](#)と比較するには、ForAllValues または ForAnyValue 集合演算子を使用する必要があります。これらの集合演算子は、リクエストにあるタグのセットと、ポリシー条件にあるタグのセットなど、2つの値のセットを比較する場合に使われます。

ForAllValues および ForAnyValue の限定子によって条件演算子にセット演算機能が追加されるため、複数のリクエストコンテキストキー値を、ポリシー条件にある複数のコンテキストキー値と照合できます。さらに、ワイルドカードまたは変数を使用してポリシーに複数値の文字列コンテキストキーを含める場合は、StringLike [条件演算子](#)も使用する必要があります。複数の条件キー値は、[配列](#)のように括弧で囲む必要があります。例えば、"Key2":["Value2A", "Value2B"] です。

- ForAllValues この限定子は、リクエストセットのすべてのメンバーの値が条件コンテキストキーセットのサブセットであるかどうかをテストします。リクエストのすべてのコンテキストキーバリューが、ポリシーの1つ以上のコンテキストキーの値と一致する場合、条件は true を返します。また、リクエストにコンテキストキーがない場合、またはキーバリューが空の文字列など

の null データセットに解決される場合は true を返します。欠落したコンテキストキーや、値のないコンテキストキーが true と評価されないようにするには、ポリシーの中に [Null](#) 条件演算子を false となるような値で含めておき、コンテキストキーが存在することと、その値が null でないことがチェックできるようにします。

#### Important

リクエストコンテキストで、コンテキストキーが欠落していたり、値が空であるコンテキストキーが予期せず存在したりすると、許容範囲が広すぎる場合があります。そのため、ForAllValues を Allow 効果で使用する場合は注意してください。ポリシーにある Null 条件演算子に false 値を指定すると、コンテキストキーが存在するかどうかと、その値が null でないかどうかを確認できます。例については、「[タグキーに基づいたアクセスの制御](#)」を参照してください。

- ForAnyValue この限定子は、リクエストコンテキストキーの値のセットの、少なくとも 1 つのメンバーが、ポリシー条件にあるコンテキストキーバリューのセットの、少なくとも 1 つのメンバーと一致するかどうかをテストします。リクエスト内のコンテキストキーバリューのいずれかが、ポリシーのコンテキストキーの値のいずれかと一致する場合に true が返されます。一致するコンテキストキーまたは空のデータセットがない場合、条件は false を返します。

#### Note

単一値と複数値のコンテキストキーの違いは、ポリシー条件の値の数ではなく、リクエストコンテキストの値の数によります。

## 条件ポリシーの例

IAM ポリシーでは、単一値のコンテキストキーと複数値のコンテキストキー、その両方に複数の値を指定して、リクエストコンテキストと比較できます。次の一連のポリシーの例では、複数のコンテキストキーと値を使用しているポリシー条件を示しています。

#### Note

このリファレンスガイドに含めるポリシーを送信する場合は、このページの下部にある [フィードバック] ボタンを使用します。IAM ID ベースのポリシーの例については、「[IAM アイデンティティベースのポリシーの例](#)」を参照してください。

### 条件ポリシーの例：単一値のコンテキストキー

- 単一値のコンテキストキーを持つ、複数の条件ブロック。(例を表示します。)
- 単一値のコンテキストキーバリューを複数持つ、1つの条件ブロック。(例を表示します。)

### 条件ポリシーの例：複数値のコンテキストキー

- 条件集合演算子の ForAllValues を使用している拒否ポリシー。(例を表示します。)
- 条件集合演算子の ForAnyValue を使用している拒否ポリシー。(例を表示します。)

### 複数値のコンテキストキーの主な例

次の一連のポリシーの例では、複数値のコンテキストキーを使用してポリシー条件を作成する方法を示しています。

#### 例：条件集合演算子の ForAllValues を使用している拒否ポリシー

次の ID ベースのポリシーの例では、特定のタグキーのプレフィックスがリクエストに含まれている場合に、IAM タグ付けアクションの使用を拒否します。コンテキストキーの `aws:TagKeys` には、文字列を部分一致させるために、ワイルドカード (\*) が含まれています。リクエストコンテキストキーには複数の値を含められるため、コンテキストキー `aws:TagKeys` を使用しているセット演算子 `ForAllValues` がポリシーに含まれています。コンテキストキー `aws:TagKeys` が `true` を返すようにするには、リクエストのすべての値がポリシーの 1 つ以上の値と一致する必要があります。

また `ForAllValues` セット演算子は、リクエストにキーがない場合、またはコンテキストのキーバリューが `null` データセット (空の文字列など) に解決される場合、`true` を返します。欠落したコンテキストキーや、値のないコンテキストキーが `true` と評価されないようにするには、ポリシーの中に `Null` 条件演算子を `false` となるような値で含めておき、コンテキストキーが存在することと、その値が `null` でないことがチェックできるようにします。

#### Important

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "DenyRestrictedTags",
    "Effect": "Deny",
    "Action": [
      "iam:Tag*",
      "iam:Untag*"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "Null": {
        "aws:TagKeys": "false"
      },
      "ForAllValues:StringLike": {
        "aws:TagKeys": [
          "key1*",
          "key2*",
          "key3*"
        ]
      }
    }
  }
]
```

例：条件集合演算子の ForAnyValue を使用している拒否ポリシー

次の ID ベースのポリシーの例では、ポリシー、environment または webserver で指定されているタグキーのいずれかでタグ付けされたスナップショットがある場合、EC2 インスタンスボリュームのスナップショットの作成を拒否します。リクエストコンテキストキーには複数の値を含められるため、コンテキストキー aws:TagKeys を使用しているセット演算子 ForAnyValue がポリシーに含まれています。ポリシーで指定されたタグキーの値が、タグ付けリクエストに 1 つでも含まれている場合、aws:TagKeys コンテキストキーは true を返し、拒否ポリシーの効果をもたらします。

**⚠ Important**

このポリシーでは、アクションを許可しません。特定のアクションを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:CreateSnapshot",
        "ec2:CreateSnapshots"
      ],
      "Resource": "arn:aws:ec2:us-west-2::snapshot/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["environment", "webserver"]
        }
      }
    }
  ]
}
```

### 単一値のコンテキストキーポリシーの例

次の一連のポリシーの例では、単一値のコンテキストキーを使用してポリシー条件を作成する方法を示しています。

例：単一値のコンテキストキーを持つ、複数の条件ブロック

条件ブロックに複数の条件があり、それぞれにコンテキストキーが1つしかない場合、目的の Allow または Deny 効果を呼び出すには、すべてのコンテキストキーが true になる必要があります。否定一致条件演算子を使用すると、条件の値を評価するロジックが反転します。

次の例では、ユーザーが EC2 ボリュームを作成し、ボリューム作成時にボリュームにタグを適用しています。リクエストコンテキストには、コンテキストキー `aws:RequestTag/project` の値を含める必要があります。本稼働環境以外ならば、コンテキストキー `aws:ResourceTag/environment` の値はどの値でもかまいません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVolume",
      "Resource": "*"
    }
  ]
}
```

```
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:::volume/*",
    "Condition": {
      "StringLike": {
        "aws:RequestTag/project": "*"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:region:account:*/*",
    "Condition": {
      "StringNotEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  }
]
```

リクエストコンテキストには、プロジェクトのタグ値が含まれている必要があり、本稼働環境のリソース用にリクエストコンテキストを作成して Allow 効果を呼び出すことはできません。プロジェクト名に QA リソースタグがついて Feature3 となっているため、次の EC2 ボリュームは正常に作成されました。

```
aws ec2 create-volume \
  --availability-zone us-east-1a \
  --volume-type gp2 \
  --size 80 \
  --tag-specifications 'ResourceType=volume,Tags=[{Key=project,Value=Feature3},
{Key=environment,Value=QA}]'
```

例：単一値のコンテキストキーと値が複数ある、1つの条件ブロック

条件ブロックに複数のコンテキストキーが含まれていて、それぞれのコンテキストキーに複数の値がある場合、目的の Allow または Deny 効果を呼び出すには、少なくとも1つのキーバリューが true になる必要があります。否定一致条件演算子を使用すると、コンテキストキーの値を評価するロジックが反転します。

次の例では、ユーザーが Amazon Elastic Container Service クラスターで、タスクを開始して実行できるようにします。

- リクエストコンテキストには、AND `aws:RequestTag/environment` コンテキストキー用に、`production` または `pre-prod` を含める必要があります。
- `ecs:cluster` コンテキストキーにより、`default1` または `default2` の ARN ECS クラスターでタスクが実行されたことが確実にになります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask",
        "ecs:StartTask"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": [
            "production",
            "prod-backup"
          ]
        },
        "ArnEquals": {
          "ecs:cluster": [
            "arn:aws:ecs:us-east-1:111122223333:cluster/default1",
            "arn:aws:ecs:us-east-1:111122223333:cluster/default2"
          ]
        }
      }
    }
  ]
}
```

## IAM ポリシーの要素: 変数とタグ

ポリシーを記述するときにリソースや条件キーの正確な値がわからない場合、AWS Identity and Access Management (IAM) のポリシー変数をプレースホルダーとして使用します。

### Note

AWS が変数を解決できない場合は、これによってステートメント全体が無効になる可能性があります。たとえば、`aws:TokenIssueTime` 変数を使用する場合は、リクエストが一時的な認証情報 (IAM ロール) を使用して認証した場合にのみ、変数は値に解決されます。変数が無効なステートメントを発生させないようにするには、[...IfExists 条件演算子を使用します。](#)

### トピック

- [序章](#)
- [ポリシーでの変数の使用](#)
- [ポリシー変数としてのタグ](#)
- [ポリシー変数を使用する場所](#)
- [値のないポリシー変数](#)
- [ポリシー変数に使用可能なリクエスト情報](#)
- [デフォルト値の指定](#)
- [詳細情報](#)

### 序章

IAM ポリシーでは、アクセスを制御する特定のリソースに対して名前を指定することが多くのアクセシオンで許可されています。例えば、以下のポリシーでは、ユーザーは、marketing プロジェクトの S3 バケット DOC-EXAMPLE-BUCKET 内のオブジェクトの一覧表示、読み取り、および書き込みができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```



```
    "Action": ["s3:ListBucket"],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
    "Condition": {"StringLike": {"s3:prefix": ["marketing/*"]}}
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject"
    ],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/marketing/*"]
  }
]
```

場合によっては、ポリシーを書く際に対象リソースの正確な名前が分からないときがあります。ポリシーを汎用化して多くのユーザーが共用できるようにすると、ユーザーごとにポリシーの一意なコピーを作成する必要がなくなります。ユーザーごとの個別のポリシーを作成するのではなく、そのグループに属するすべてのユーザーに対して機能する単一のグループポリシーを作成することをお勧めします。

## ポリシーでの変数の使用

ポリシーにプレースホルダーを設定するポリシー変数を使用して、ポリシー内で動的な値を定義できます。

変数は、\$ プレフィックスを使用してマークされ、その後に、リクエストに値がある変数名が、中括弧 ({ }) で囲まれて続きます。

ポリシーの評価時に、ポリシー変数はリクエスト自体の条件コンテキストから取得された値に置き換えられます。変数は、[ID ベースのポリシー、リソースポリシー、サービスコントロールポリシー、セッションポリシー、VPC エンドポイントポリシーで使用できます](#)。アクセス許可の境界として使用される ID ベースのポリシーは、ポリシー変数もサポートします。

グローバル条件コンテキストキーは、AWS サービス間のリクエストの変数として使用できます。サービス固有の条件キーは、AWS リソースを操作するときの変数としても使用できますが、それらをサポートするリソースに対してリクエストが行われた場合にのみ使用できます。各 AWS サービスとリソースで使用可能なコンテキストキーのリストについては、[サービス認証リファレンスをご覧ください](#)。特定の状況では、グローバル条件コンテキストキーに値を設定できません。詳細については、「[AWS グローバル条件コンテキストキー](#)」を参照してください。

**⚠ Important**

- キー名では大文字と小文字は区別されません。たとえば、aws:CurrentTime と AWS:currenttime は同じです。
- 単一値の条件キーは、変数として使用できます。複数値の条件キーは、変数としては使用できません。

次の例は、特定のリソース名をポリシー変数に置き換える IAM ロールまたはユーザー用のポリシーを示しています。このポリシーは、aws:PrincipalTag 条件キーを利用して再利用できます。このポリシーが評価されると、`${aws:PrincipalTag/team}` では、バケット名が team プリンシパルタグのチーム名で終わる場合にのみアクションが許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:PrincipalTag/team}/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/team}/*"]
    }
  ]
}
```

変数は、\$ プレフィックスの後に波かっこ ( { } ) を付けることでマークできます。\${ } 文字の内部には、ポリシーで使用する値の名前をリクエストから取得して含めることができます。使用できる値については、このページの後半で説明します。

このグローバル条件キーの詳細については、グローバル条件キーのリストの「[aws:PrincipalTag/tag-key](#)」を参照してください。

**Note**

ポリシー変数を使用するには、ステートメントに `Version` 要素を含めて、ポリシー変数をサポートするバージョンに設定する必要があります。変数はバージョン 2012-10-17 から導入されました。旧バージョンでのポリシー言語では、ポリシー変数をサポートしていません。Version 要素を含めず、適切なバージョンの日付に設定しない場合、`${aws:username}` などの変数はポリシー内のリテラル文字列として扱われます。Version ポリシー要素は、ポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。一方で、ポリシーバージョンは、IAM でカスタマー管理ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。Version ポリシー要素の詳細については、「[the section called “Version”](#)」を参照してください。ポリシーのバージョンの詳細については、「[the section called “IAM ポリシーのバージョンニング”](#)」を参照してください。

プリンシパルが S3 バケットの `/David` パスからオブジェクトを取得することを許可するポリシーは次のようになります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/David/*"]
  }]
}
```

このポリシーがユーザー David にアタッチされている場合、そのユーザーは自身の S3 バケットからオブジェクトを取得しますが、このユーザー名を含む各ユーザーには別のポリシーを作成する必要があります。その後、各ポリシーを個々のユーザーに関連付ける必要があります。

ポリシー変数を使用することで、再利用できるポリシーを作成できます。次のポリシーでは、`aws:PrincipalTag` のタグキー値がリクエストで渡されたタグキーの `owner` 値と一致する場合、ユーザーは Amazon S3 バケットからオブジェクトを取得できます。

```
{
  "Version": "2012-10-17",
  "Statement": [{
```

```
"Sid": "AllowUnlessOwnedBySomeoneElse",
"Effect": "Allow",
"Action": ["s3:GetObject"],
"Resource": ["*"],
"Condition": {
  "StringEquals": {
    "s3:ExistingObjectTag/owner": "${aws:PrincipalTag/owner}"
  }
}
]
```

このようにユーザーのところにポリシー変数を使用することで、ユーザーごとに別々のポリシーを作成する必要がなくなります。次の例では、プロダクトマネージャーが一時的なセキュリティ認証情報を使用して引き受ける IAM ロールにポリシーをアタッチしています。ユーザーが Amazon S3 オブジェクトへのアクセスをリクエストした場合、IAM では `${aws:PrincipalTag}` 変数を現在のリクエストから取得した dept タグ値に置き換えてポリシーを評価します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowOnlyDeptS3Prefix",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/dept}/*"],
  }
]
```

## ポリシー変数としてのタグ

一部の AWS サービスでは、それらのサービスによって作成されたリソースに独自のカスタム属性をアタッチできます。例えば、Amazon S3 バケットや IAM ユーザーにタグを適用できます。これらのタグはキーバリューのペアです。タグキー名と、そのキー名に関連付けられた値とを定義します。たとえば、**department** キーと **Human Resources** 値を持つタグを作成します。IAM エンティティのタグ付けの詳細については、「[IAM リソースのタグ付け](#)」を参照してください。他の AWS サービスによって作成されるリソースのタグ付けについては、そのサービスのドキュメントを参照してください。タグエディタの使用については、「AWS Management Console ユーザーガイド」の「[タグエディタの使用](#)」を参照してください。

IAM リソースにタグ付けすることで、IAM リソースの検出、整理、および追跡を簡単に行うことができます。また、IAM アイデンティティにタグ付けして、リソースまたはタグ付け自体へのアクセスを制御することもできます。タグを使用したアクセスの制御については、「[タグを使用した IAM ユーザーおよびロールへのアクセスとそのユーザーおよびロールのアクセスの制御](#)」を参照してください。

## ポリシー変数を使用する場所

Resource 要素のポリシー変数、および Condition 要素の文字列比較を活用できます。

## リソースの要素

ポリシー変数は Resource 要素で使用できますが、ARN のリソース部分でのみ使用できます。ARN のこの部分は、5 番目のコロン (:) の後に表示されます。サービスやアカウントなど 5 番目のコロンよりも前の ARN の部分は、変数を使用して置き換えることはできません。ARN 形式の詳細については、[IAM ARN](#)を参照してください。

ARN の一部をタグ値で置き換えるには、プレフィックスとキー名を `${ }` で囲みます。例えば、次のリソース要素は、リクエストしているユーザーの部門タグの値と同じ名前のバケットのみを参照します。

```
"Resource": ["arn:aws::s3:::bucket/${aws:PrincipalTag/department}"]
```

多くの AWS リソースは、ユーザーが作成した名前を含む ARN を使用します。以下の IAM ポリシーでは、アクセスプロジェクト、アクセスアプリケーション、アクセス環境のタグ値が一致する対象ユーザーのみがリソースを変更できるようにしています。さらに、[\\*ワイルドカードマッチ](#)を使用すると、カスタムリソース名のサフィックスを指定できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessBasedOnArnMatching",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns>DeleteTopic"
      ],
      "Resource": ["arn:aws:sns:*:*:${aws:PrincipalTag/access-project}-
${aws:PrincipalTag/access-application}-${aws:PrincipalTag/access-environment}-*"]
    }
  ]
}
```

```
}
```

## 条件の要素

ポリシー変数は、文字列演算子または ARN 演算子を含む任意の条件の Condition の値に使用できます。文字列演算子には StringEquals、StringLike、StringNotLike などが含まれます。ARN 演算子には ArnEquals と ArnLike が含まれます。ポリシー変数は Numeric、Date、Boolean、Binary、IP Address、または Null など、他の演算子では使用できません。条件演算子の詳細については、「[IAM JSON ポリシー要素: 条件演算子](#)」を参照してください。

Condition 要素式のタグを参照するときは、関連する接頭辞とキー名を条件キーとして使用します。次に、条件値でテストする値を使用します。

例えば、次のポリシー例では、ユーザーにフルアクセスが許可されていますが、タグ costCenter がユーザーにアタッチされている場合に限りです。タグの値は、12345 または 67890 である必要があります。タグ値が設定されていない場合、またはそれ以外の値の場合、リクエストは失敗します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:*user*"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:ResourceTag/costCenter": [ "12345", "67890" ]
        }
      }
    }
  ]
}
```

## 値のないポリシー変数

ポリシー変数が、値のない、またはリクエストの承認コンテキストに存在しない条件コンテキストキーを参照する場合、その値は事実上 null になります。同等または同等の価値はありません。以下の場合、条件コンテキストキーが認可コンテキストに存在しない可能性があります。

- その条件キーをサポートしていないリソースへのリクエストで、サービス固有の条件コンテキストキーを使用しています。
- IAM プリンシパル、セッション、リソース、またはリクエストのタグは存在しません。
- [AWS グローバル条件コンテキストキー](#) 内の各グローバル条件コンテキストキーにリストされているその他の状況。

IAM ポリシーの条件要素に値のない変数を使用すると ( 「[IAM JSON ポリシー要素: 条件演算子](#)いいね」 `StringEquals` または 「`StringLike`一致しない」 )、ポリシーステートメントが有効になりません。

`StringNotEquals` または `StringNotLike` などの逆条件演算子は、テスト対象の条件キーの値が実際の NULL 値と等しくないか、等しくないため、null 値とマッチします。

例えば、アクセスを許可するには `aws:principaltag/Team` が `s3:ExistingObjectTag/Team` と等しい必要があります。`aws:principaltag/Team` が設定されていない場合、アクセスは明示的に拒否されます。承認コンテキストに値のない変数をポリシーの `Resource` または `NotResource` 要素の一部として使用すると、値のないポリシー変数を含むリソースはどのリソースとも一致しません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::/example-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:ExistingObjectTag/Team": "${aws:PrincipalTag/Team}"
        }
      }
    }
  ]
}
```

### ポリシー変数に使用可能なリクエスト情報

JSON ポリシーの `Condition` 要素を使用して、[リクエストコンテキスト](#)のキーを、ポリシーで指定したキーバリューと比較できます。ポリシー変数を使用する場合、AWS は、ポリシー内の変数の代わりにリクエストコンテキストキーの値を置換します。

## プリンシパルキーの値

aws:username、aws:userid、および aws:PrincipalType の値は、リクエストを開始したプリンシパルの値によって異なります。例えば、IAM ユーザー、IAM ロール、または AWS アカウントのルートユーザー の認証情報を使用してリクエストを行うことができます。以下のリストは、様々な種類のプリンシパルに対するキーの値を示しています。

- AWS アカウントのルートユーザー
  - aws:username: ( なし )
  - aws:userid: AWS アカウント ID
  - aws:PrincipalType: Account
- IAM ユーザー
  - aws:username: *IAM-user-name*
  - aws:userid: [一意の ID](#)
  - aws:PrincipalType: User
- 委任ユーザー
  - aws:username: ( なし )
  - aws:userid: *account:caller-specified-name*
  - aws:PrincipalType: FederatedUser
- ウェブフェデレーティッドユーザーおよび SAML フェデレーティッドユーザー

### Note

OIDC フェデレーションを使用するときに利用可能なポリシーキーの詳細については、「[???](#)」を参照してください。

- aws:username: ( なし )
- aws:userid: ( なし )
- aws:PrincipalType: AssumedRole
- 委任されたロール
  - aws:username: ( なし )
  - aws:userid: *role-id:caller-specified-role-name*
  - aws:PrincipalType: Assumed role



- Amazon EC2 インスタンスに割り当てられたロール
  - `aws:username:` ( なし )
  - `aws:user-id:` *role-id:ec2-instance-id*
  - `aws:PrincipalType:` Assumed role
- 匿名の発信者 ( Amazon SQS Amazon SNS および Amazon S3 のみ )
  - `aws:username:` ( なし )
  - `aws:user-id:` ( なし )
  - `aws:PrincipalType:` Anonymous

このリストについては、次の点に注意してください。

- なしである場合、値が現在のリクエスト情報内に存在しないために値を一致させる試みが失敗し、ステートメントが無効になることを意味します。
- *role-id* は、作成時に各ロールに割り当てられる一意の識別子です。次の AWS CLI コマンドでロール ID を表示できます: `aws iam get-role --role-name rolename`
- *caller-specified-name* および *caller-specified-role-name* は、一時的認証情報を取得するための呼び出し元プロセス (アプリケーションまたはサービスなど) によって渡された名前です。
- *ec2-instance-id* は、インスタンスにその起動時に割り当てられ、Amazon EC2 コンソールの [Instances (インスタンス)] ページに表示される値です。インスタンス ID を表示するには、次の AWS CLI コマンドを実行します: `aws ec2 describe-instances`

## フェデレーテッドユーザーがリクエストで利用可能な情報

フェデレーションユーザーとは、IAM 以外のシステムを使用して認証されたユーザーです。たとえば、ある企業が AWS に発信する社内用のアプリケーションを使用しているとします。このアプリケーションを使用する社内すべてのユーザーに IAM ID を発行することは実際的ではありません。代わりに、単一の IAM ID を持つプロキシ (中間層) アプリケーションを使用するか、SAML ID プロバイダー (IdP) を使用することができます。プロキシアプリケーションまたは SAML IdP は、企業ネットワークを使用して個々のユーザーを認証します。プロキシアプリケーションは、その IAM ID を使用して、個々のユーザーの一時的セキュリティ認証情報を取得できます。SAML IdP は、実際には AWS 一時的セキュリティ認証情報の ID に置き換えることができます。次に、一時的な認証情報を使用して、AWS リソースにアクセスできます。

同様に、AWS リソースへのアクセスが必要なモバイルデバイス用アプリを作成することもできます。この場合、「OIDC フェデレーション」を使用すると、アプリケーションで、Login with Amazon、Amazon Cognito、Facebook、Google などのよく知られた ID プロバイダーを使用するユーザーを認証できます。その後、アプリはこれらのプロバイダーから取得したユーザーの認証情報を用いて、AWS リソースにアクセスするための一時認証情報を取得します。

OIDC フェデレーションを使用する場合は、Amazon Cognito と AWS Mobile SDK を利用することをお勧めします。詳細については、次を参照してください:

- [Amazon Cognito ユーザーガイド](#)
- [一時的な認証情報の一般的なシナリオ](#)

## 特殊文字

それ以外の場合は特別な意味を持つ文字を表すことのできる、固定値を持つ事前定義された特殊なポリシー変数がいくつかあります。これらの特殊文字が、一致を試みている文字列、またリテラルに挿入した文字列の一部である場合、これらは誤って解釈されます。たとえば、文字列に挿入されたアスタリスク (\*) は、リテラルな \* としてではなく、任意の文字列を指定するワイルドカードとして解釈されます。この場合、事前定義した次のポリシー変数を使用できます。

- `{*}` - \* (アスタリスク) が必要な場合に使用します。
- `{?}` - ? (疑問符) が必要な場合に使用します。
- `{$}` - \$ (ドル記号) が必要な場合に使用します。

これらの事前定義されたポリシー変数は、正規のポリシー変数を使用できるすべての文字列で使用できます。

## デフォルト値の指定

変数にデフォルト値を追加するには、デフォルト値を一重引用符 ( ' ' ) で囲み、変数テキストとデフォルト値をコンマとスペース ( , ) で区切ります。

例えば、プリンシパルが `team=yellow` でタグ付けされている場合、`DOC-EXAMPLE-BUCKET-yellow` という名前で ExampleCorp's の Amazon S3 バケットにアクセスできます。このリソースを使用するポリシーでは、チームメンバーがチームのバケットにアクセスすることは許可されますが、他のチームのリソースにアクセスすることはできません。チームタグのないユーザーの場合、`company-wide` のデフォルト値をバケットの名前として設定します。これらのユーザーは

DOC-EXAMPLE-BUCKET-company-wide バケットにのみアクセスでき、チームへの参加方法などの一般的な情報を確認できます。

```
"Resource": "arn:aws:s3::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

## 詳細情報

ポリシーの詳細については、以下を参照してください。

- [IAM でのポリシーとアクセス許可](#)
- [IAM アイデンティティベースのポリシーの例](#)
- [IAM JSON ポリシー要素のリファレンス](#)
- [ポリシーの評価論理](#)
- [OIDC フェデレーション](#)

## IAM JSON ポリシー要素: サポートされているデータ型

このセクションでは、JSON ポリシーで値を指定するときにサポートされているデータの種類の一覧を掲載します。ポリシー言語では、各ポリシー要素のすべての種類がサポートされているわけではありません。各要素の詳細については、前のセクションを参照してください。

- 文字列
- 数字 ( 整数および浮動小数点数 )
- ブール値
- Null
- リスト
- マップ
- 構造体 ( 入れ子形式のマップ )

以下の表は、各データの種類のシリアル化してまとめたものです。すべてのポリシーは UTF-8 形式でなければいけないことに注意してください。JSON データの種類については、[RFC 4627](#) を参照してください。

タイプ	JSON
文字列	文字列
整数	数値
浮動小数点	数値
ブール値	true または false
Null	null
日付	<a href="#">ISO 8601 の W3C プロファイル</a> に準拠する文字列
IP アドレス	<a href="#">RFC 4632</a> に準拠する文字列
リスト	配列
オブジェクト	オブジェクト

## ポリシーの評価論理

プリンシパルが AWS Management Console、AWS API、または AWS CLI を使用しようとする  
と、プリンシパルは AWS にリクエストを送信します。AWS サービスが、リクエストを受け取る  
と、AWS は、いくつかのステップを実行してリクエストの許可または拒否を決定します。

1. **認証** – AWS は、必要に応じて、最初にリクエストを行ったプリンシパルを認証します。このステップは、匿名ユーザーからの一部のリクエストを許可するいくつかのサービス (Amazon S3 など) では不要です。
2. **[リクエストコンテキストの処理](#)** – AWS は、リクエスト内で収集した情報を処理し、リクエストに適用するポリシーを決定します。
3. **[単一アカウント内のポリシーを評価する](#)** – AWS は、すべてのポリシータイプを評価します。ポリシータイプは、ポリシーを評価する順序に影響を与えます。
4. **[アカウント内でのリクエストの許可または拒否の決定](#)** – AWS 次にリクエストコンテキストに対してポリシーを処理して、リクエストが許可されているか拒否されているかを判断します。

## リクエストコンテキストの処理

AWS はリクエストを処理して、以下の情報をリクエストコンテキスト内に取り込みます。

- アクション (またはオペレーション) – プリンシパルが実行するアクションまたはオペレーション。
- リソース – アクションまたはオペレーションを実行する対象の AWS リソースオブジェクト。
- プリンシパル – リクエストの送信元のユーザー、ロール、フェデレーティッドユーザー、またはアプリケーション。プリンシパルに関する情報には、そのプリンシパルに関連付けられたポリシーが含まれます。
- 環境データ – IP アドレス、ユーザーエージェント、SSL 有効化ステータス、または時刻に関する情報。
- リソースデータ – リクエストされているリソースに関連するデータ。これには、DynamoDB テーブル名、Amazon EC2 インスタンスのタグなどの情報が含まれる場合があります。

次に、AWS は以上の情報を使用してリクエストコンテキストに適用するポリシーを見つけます。

### 単一アカウント内のポリシーを評価する

AWS によるポリシーの評価方法は、リクエストコンテキストに適用するポリシーのタイプによって異なります。以下のポリシータイプ (使用頻度の高い順に表示) は、単一の AWS アカウントで使用することができます。これらのポリシータイプの詳細については、「[IAM でのポリシーとアクセス許可](#)」をご参照ください。AWS がクロスアカウントアクセスのポリシーを評価する方法については、「[クロスアカウントポリシーの評価論理](#)」を参照してください。

1. ID ベースのポリシー – アイデンティティベースのポリシーは IAM ID (ユーザー、ユーザーのグループ、またはロール) にアタッチされており、アクセス許可を IAM エンティティ (ユーザーおよびロール) にアタッチします。リクエストにアイデンティティベースのポリシーのみ、リクエストに適用された場合、AWS では、それらのすべてのポリシーに少なくとも 1 つ Allow がないか確認します。
2. リソースベースのポリシー – リソースベースのポリシーは、プリンシパルとして指定されたプリンシパル (アカウント、ユーザー、ロール、ロールセッションなどのセッションプリンシパル、IAM フェデレーションユーザー) にアクセス許可を付与します。このアクセス許可では、ポリシーがアタッチされているリソースに対してプリンシパルが実行できる操作を定義します。リソースベースのポリシーとアイデンティティのポリシーの両方がリクエストに適用された場合、AWS では、それらのすべてのポリシーに少なくとも 1 つ Allow がないか確認します。リソースベースのポリシーが評価されると、ポリシーで指定されたプリンシパル ARN によって、他のポリシータイプの暗黙的な拒否が、最終的な決定に適用されるかどうかが決まります。

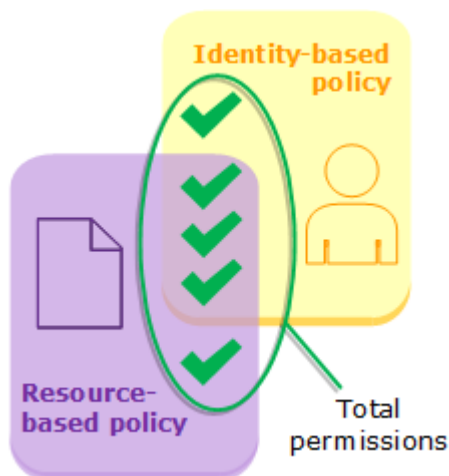
3. IAM アクセス許可の境界 – アクセス許可の境界は、アイデンティティベースのポリシーが IAM エンティティ (ユーザーまたはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティのアクセス許可の境界を設定した場合、エンティティは、アイデンティティベースのポリシーとそのアクセス許可の境界の両方で許可されているアクセス許可のみ実行できます。アクセス許可の境界で暗黙的に拒否しても、リソースベースのポリシーによって付与されるアクセス許可は制限される場合もあります。詳細については、このトピックの後半の「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。
4. AWS Organizations サービスコントロールポリシー (SCP) – Organizations SCP は、組織または組織単位 (OU) のアクセス許可の上限を指定します。最大 SCP はメンバーアカウントのプリンシパルに適用されます (各 AWS アカウントのルートユーザー など)。SCP が存在する場合、アイデンティティのポリシーおよびリソースベースのポリシーは、それらのポリシーと SCP によってアクションが許可されている場合にのみ、メンバーアカウントのプリンシパルにアクセス許可を付与します。アクセス許可の境界と SCP が両方存在する場合は、アクセス許可の境界、SCP、およびアイデンティティのポリシーによって、すべてのアクションが許可されます。
5. セッションポリシー – セッションポリシーは、ロールまたはフェデレーテッドユーザーの一時セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。ロールセッションをプログラムで作成するには、AssumeRole\* API オペレーションのいずれかを使用します。これを行い、セッションポリシーに合格すると、結果として得られるセッションのアクセス許可は、IAM エンティティの ID ベースのポリシーとセッションポリシーの共通部分です。フェデレーションユーザーのセッションを作成するには、IAM ユーザーのアクセスキーを使用して、API の GetFederationToken オペレーションをプログラムで呼び出します。リソースベースのポリシーは、セッションポリシーのアクセス許可ポリシーの評価に異なる影響を与えます。この違いは、ユーザーまたはロールの ARN またはセッションの ARN がリソースベースのポリシーでプリンシパルとして表示されているかどうかによって異なります。詳細については、「[セッションポリシー](#)」を参照してください。

これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になる点に注意してください。

#### リソースベースのポリシーを使用したアイデンティティベースのポリシーの評価

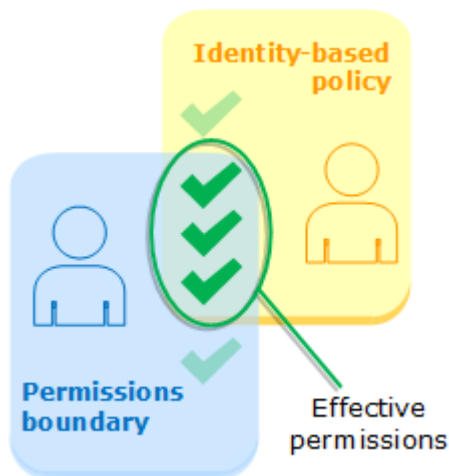
アイデンティティのポリシーとリソースベースのポリシーは、それらが関連付けられているアイデンティティまたはリソースにアクセス許可を付与します。IAM エンティティ (ユーザーまたはロール) が同じアカウントのリソースへのアクセスをリクエストした場合、AWS は、アイデンティティのポリシーおよびリソースベースのポリシーによって付与されているすべてのアクセス許可を評価します。結果として、合計 2 種類のアクセス許可が付与されます。アクションがアイデンティティのポ

リソースベースのポリシー、または両方で許可されている場合、AWS はそのアクションを許可します。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。



### アクセス許可の境界を使用したアイデンティティベースのポリシーの評価

AWS でユーザーのアイデンティティベースのポリシーとアクセス許可の境界を評価する場合は、2つのカテゴリの共通部分に基づいてアクセス許可が決定されます。つまり、既存のアイデンティティベースのポリシーを使用してアクセス許可の境界をユーザーに追加すると、ユーザーが実行できるアクションが制限される場合があります。逆に、ユーザーからアクセス許可の境界を削除すると、ユーザーが実行できるアクションが増える場合があります。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。他のポリシータイプがアクセス許可の境界で評価される方法について確認するには、「[境界を設定した場合の有効なアクセス許可の評価](#)」を参照してください。



### SCP を使用したアイデンティティベースのポリシーの評価

ユーザーが組織のメンバーであるアカウントに所属している場合、結果として得られるアクセス許可はユーザーのポリシーと SCP の共通部分です。つまり、アクションは、アイデンティティベースの



ポリシーと SCP の両方で許可されます。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。



[自分のアカウントが AWS Organizations の組織のメンバーであるかどうか](#)を確認できます。組織のメンバーは、SCP による影響を受ける可能性があります。AWS CLI コマンドまたは AWS API オペレーションを使用してこのデータを表示するには、Organizations エンティティに対する `organizations:DescribeOrganization` アクションのアクセス許可が必要です。Organizations コンソールでオペレーションを実行するには、追加のアクセス許可が必要です。SCP が特定のリクエストへのアクセスを拒否しているかどうかを確認する、または有効なアクセス権限を変更するには、AWS Organizations 管理者に連絡してください。

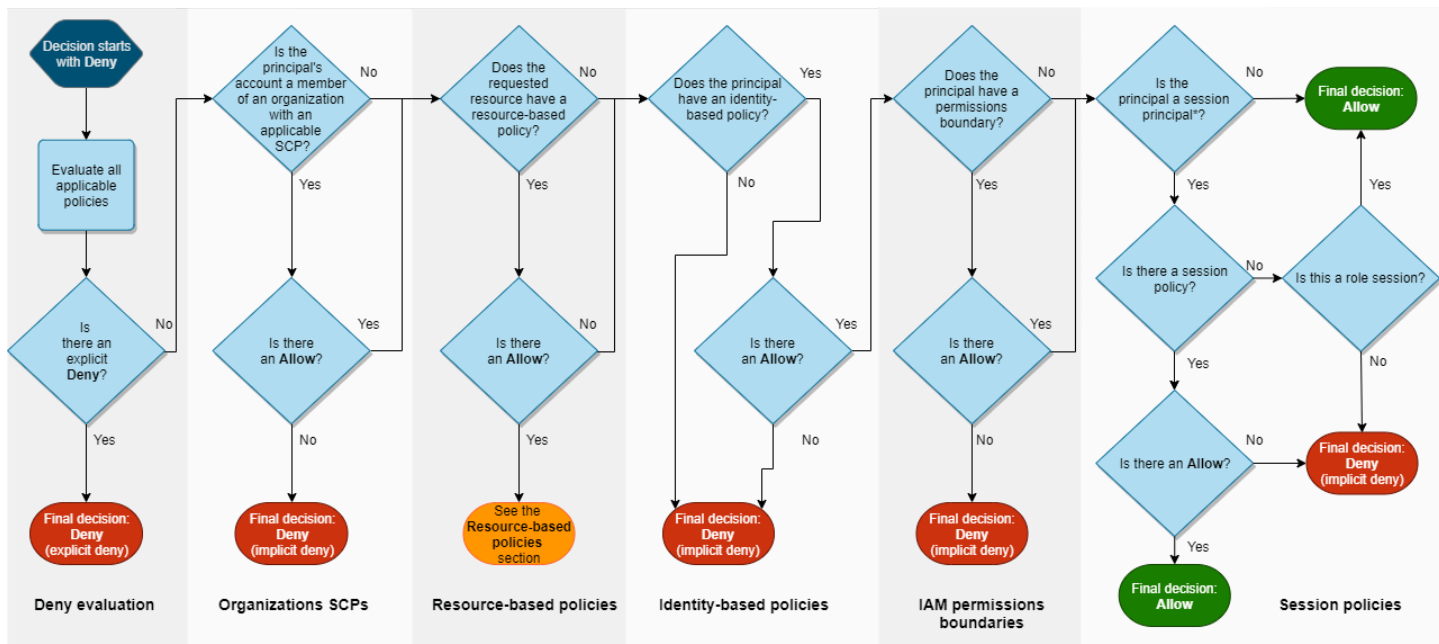
## アカウント内でのリクエストの許可または拒否の決定

プリンシパルが、プリンシパルのエンティティと同じアカウントのリソースにアクセスするためのリクエストを AWS に送信するとします。AWS エンフォースメントコードは、リクエストを許可するか拒否するかどうかを決定します。AWS は、リクエストコンテキストに適用されるすべてのポリシーを評価します。単一アカウントでのポリシーの AWS 評価ロジックの概要を次に示します。

- デフォルトでは、フルアクセスを持つ AWS アカウントのルートユーザー 以外は、すべてのリクエストが暗黙的に拒否されます。
- アイデンティティベースのポリシーまたはリソースベースのポリシーに明示的な許可が含まれている場合、このデフォルト設定は上書きされます。
- アクセス許可の境界、Organizations SCP、またはセッションポリシーがある場合、この許可は明示的な拒否で上書きされる場合があります。
- ポリシー内の明示的な拒否は、すべての許可に優先します。



以下のフローチャートは、決定が下されるまでの詳細な流れを示しています。このフローチャートでは、リソースベースのポリシーと他のタイプのポリシーでの暗黙的な拒否による影響はカバーしていません。



\*A session principal is either a role session or an IAM federated user session.

- 拒否の評価 – デフォルトでは、すべてのリクエストが拒否されます。これは**暗黙的な拒否**と呼ばれます。AWS エンフォースメントコードでは、リクエストに適用されるアカウント内のすべてのポリシーを評価します。このポリシータイプには、AWS Organizations SCP、リソースベースのポリシー、ID ベースのポリシー、IAM 許可の境界、セッションポリシーなどがあります。エンフォースメントコードでは、すべての該当するポリシーでリクエストに適用される Deny ステートメントを探します。このプロセスは**明示的な拒否**と呼ばれています。エンフォースメントコードにより、適用される明示的な拒否が 1 つでも見つかり、拒否の最終決定が返されます。明示的な拒否がない場合は、エンフォースメントコードの評価は続行されます。
- Organizations SCP - その後、エンフォースメントコードは、リクエストに適用する AWS Organizations サービスコントロールポリシー (SCP) を評価します。SCP は、SCP がアタッチされているアカウントのプリンシパルに適用されます。エンフォースメントコードで SCP に Allow ステートメントが見つからなかった場合、例えば暗黙の否定であっても、リクエストは明示的に拒否されます。エンフォースメントコードにより、拒否の最終決定が返ります。SCP がいない場合、または SCP により、リクエストされたアクションが許可された場合は、エンフォースメントコードが続行されます。
- リソースベースのポリシー - 同じアカウント内でも、リソースにアクセスするプリンシパルのタイプと、リソースベースのポリシーで許可されるプリンシパルに応じて、リソースベースのポリシーはポリシーの評価に異なる影響を与えます。リソースベースのポリシーの Allow は、ID

ベースのポリシー、アクセス許可の境界、またはセッションポリシーに潜在的な拒否があったとしても、プリンシパルのタイプに応じて最終的には Allow に決定されます。

ほとんどのリソースでは、ID ベースのポリシーまたはリソースベースのポリシーのいずれかでプリンシパルを明示的に許可するだけで、アクセス権を付与できます。[IAM ロール信頼ポリシー](#)と[KMS キーポリシー](#)は、[プリンシパル](#)へのアクセスを明示的に許可する必要があるため、このロジックの例外です。

指定されたプリンシパルが IAM ユーザー、IAM ロール、またはセッションプリンシパルである場合、リソースベースのポリシーのロジックは他のポリシータイプとは異なります。セッションプリンシパルには、IAM [ロールセッション](#)または[IAM フェデレーテッドユーザーセッション](#)が含まれます。リソースベースのポリシーが、リクエストを作成している IAM ユーザーまたはセッションプリンシパルに直接アクセス許可を付与する場合、ID ベースのポリシー、アクセス許可の境界、またはセッションポリシーで暗黙的な拒否が最終的な決定に影響を与えることはありません。

次の表は、ID ベースのポリシー、アクセス許可の境界、セッションポリシーで暗黙的な拒否が存在する場合に、さまざまなプリンシパルタイプに対する、リソースベースのポリシーの影響を理解するのに役立ちます。

## リソースベースのポリシーと他のポリシータイプでの暗黙的な拒否 (同じアカウント)

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
[IAM role] (IAM ロール)	該当しない	該当しない	該当しない	該当しない	該当しない	ロール自体がリクエストを行うことはできません。リクエストは、ロールを引き受けた後に、ロールセッションで行われます。

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
IAM ロールセッション	ロール ARN を許可する	暗黙的な拒否	暗黙的な拒否	暗黙的な拒否	DENY	アクセス許可の境界とセッションポリシーは、最終決定の一部として評価されます。いずれかのポリシーで暗黙的な拒否があると、DENY と決定されます。
IAM ロールセッション	ロールセッション ARN を許可する	暗黙的な拒否	暗黙的な拒否	暗黙的な拒否	許可	アクセス許可は、セッションに直接付与されます。他のポリシータイプは、決定に影響しません。

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
IAM ユーザー	IAM ユーザーの ARN を許可する	暗黙的な拒否	暗黙的な拒否	該当しない	許可	アクセス許可は、ユーザーに直接付与されます。他のポリシータイプは、決定に影響しません。
IAM フェデレーティッドユーザー (GetFederationToken)	IAM ユーザーの ARN を許可する	暗黙的な拒否	暗黙的な拒否	暗黙的な拒否	DENY	アクセス許可の境界またはセッションポリシーで暗黙的な拒否があると、DENY と決定されます。

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
IAM フェデレーティッドユーザー (GetFederationToken)	IAM フェデレーティッドユーザーセッション ARN を許可する	暗黙的な拒否	暗黙的な拒否	暗黙的な拒否	許可	アクセス許可は、セッションに直接付与されます。他のポリシータイプは、決定に影響しません。

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
ルートユーザー	ルートユーザーの ARN を許可する	該当しない	該当しない	該当しない	許可	ルートユーザーは、AWS アカウントのすべてのリソースに完全かつ無制限にアクセスできます。AWS Organizations のアカウントのルートユーザーへのアクセスを制御する方法については、「 <a href="#">Organizations ユーザーガイド</a> 」の「 <a href="#">サービスコントロールポリシー</a> 」

リクエストを実行するプリンシパル	リソースベースのポリシー	ID ベースのポリシー	アクセス許可の境界	セッションポリシー	結果	理由
						<a href="#">(SCP)</a> 」を参照してください。
AWS サービスプリンシパル	AWS サービスプリンシパルを許可する	該当しない	該当しない	該当しない	許可	リソースベースのポリシーが、 <a href="#">AWS サービスプリンシパル</a> にアクセス許可を直接付与する場合、他のポリシータイプは決定に影響しません。

- IAM ロール— IAM ロール ARN にアクセス許可を与えるリソースベースのポリシーは、アクセス許可の境界またはセッションポリシーの暗黙的な拒否によって制限されます。

#### ロール ARN の例

```
arn:aws:iam::111122223333:role/examplerole
```

- IAM ロールセッション — 同じアカウント内で、IAM ロールセッション ARN にアクセス許可を与えるリソースベースのポリシーは、引き受けたロールセッションに、直接アクセス許可を与えます。セッションに直接付与されるアクセス許可は、ID ベースのポリシー、アクセス許可の



境界、またはセッションポリシーの暗黙的な拒否によって制限されません。ロールを引き受けてリクエストを行う場合、リクエストを行うプリンシパルは IAM ロールセッション ARN であり、ロールそれ自体の ARN ではありません。

#### ロールセッション ARN の例

```
arn:aws:sts::111122223333:assumed-role/examplerole/examplerolesessionname
```

- IAM ユーザー — 同じアカウント内では、IAM ユーザー ARN (フェデレーティッドユーザーセッションではない) へのアクセス許可を与えているリソーススペースのポリシーは、ID ベースのポリシーまたはアクセス許可の境界による、暗黙的な拒否によって制限されません。

#### IAM ユーザー ARN の例

```
arn:aws:iam::111122223333:user/exampleuser
```

- IAM フェデレーティッドユーザーセッション — IAM フェデレーティッドユーザーセッションは、[GetFederationToken](#) を呼び出して作成されたセッションです。フェデレーティッドユーザーがリクエストを行う場合、リクエストを行うプリンシパルはフェデレーティッドユーザー ARN であり、フェデレーションした IAM ユーザーの ARN ではありません。同じアカウント内で、フェデレーティッドユーザー ARN にアクセス許可を与えるリソーススペースのポリシーは、セッションに直接アクセス許可を与えます。セッションに直接付与されるアクセス許可は、ID ベースのポリシー、アクセス許可の境界、またはセッションポリシーの暗黙的な拒否によって制限されません。

ただし、リソーススペースのポリシーがフェデレーションした IAM ユーザーの ARN にアクセス許可を与える場合、セッション中にフェデレーティッドユーザーによって行われたリクエストは、アクセス許可の境界またはセッションポリシーの、暗黙的な拒否によって制限されます。

#### IAM フェデレーティッドユーザーセッション ARN の例

```
arn:aws:sts::111122223333:federated-user/exampleuser
```

4. アイデンティティベースのポリシー — コードを使用して、プリンシパルのアイデンティティベースのポリシーを確認します。IAM ユーザーの場合は、ユーザーポリシーや、ユーザーが属するグループのポリシーが含まれます。ID ベースのポリシーが無いが、ID ベースのポリシーにリクエストされたアクションを許可するステートメントがない場合、そのリクエストは暗黙的に拒否され、拒否の最終決定がコードで返ります。該当する ID ベースのポリシーのステートメントで、リクエストされたアクションが許可されている場合は、コードが継続します。

5. IAM アクセス許可の境界 – その後コードで、プリンシパルによって使用されている IAM エンティティにアクセス許可の境界があるかどうかを確認されます。アクセス許可の境界の設定に使用されているポリシーで、リクエストされたアクションが許可されていない場合、リクエストは明示的に拒否されます。この場合、拒否の最終決定が返ります。アクセス許可の境界がない場合、またはアクセス許可の境界で、リクエストされたアクションが許可されている場合、コードは続行されます。
6. セッションポリシー – その後、コードは、プリンシパルがセッションプリンシパルかどうかをチェックします。セッションプリンシパルには、IAM ロールセッションか IAM フェデレーティッドユーザーセッションが含まれます。プリンシパルがセッションプリンシパルでない場合、エンフォースメントコードは最終決定で許可を返します。

セッションプリンシパルの場合、コードは、セッションポリシーがリクエストによって渡されたかどうかをチェックします。セッションポリシーは、AWS CLI または AWS API を使用して、ロールまたは IAM フェデレーティッドユーザーの一時認証情報を取得する場合に渡すことができます。

  - セッションポリシーが存在し、リクエストされたアクションが許可されていない場合、そのリクエストは暗示的に拒否されます。この場合、拒否の最終決定が返ります。
  - セッションポリシーがない場合、プリンシパルがロールセッションかどうかをコードがチェックします。プリンシパルがロールセッションだった場合、リクエストは許可になります。それ以外の場合は、リクエストは暗黙的に拒否され、コードは最終決定で拒否を返します。
  - セッションポリシーが存在し、リクエストされたアクションが許可されている場合は、エンフォースメントコードは最終決定で許可を返します。
7. エラー – 評価中に AWS エンフォースメントコードでエラーが発生すると、例外が生成され、終了します。

## アイデンティティベースのポリシーおよびリソースベースのポリシーの評価の例

最も一般的なポリシータイプは、アイデンティティベースのポリシーとリソースベースのポリシーです。リソースへのアクセスがリクエストされると、同じアカウント内の少なくとも1つの許可について、AWS はポリシーによって付与されたすべてのアクセス許可を評価します。これらのポリシーのいずれかが明示的に拒否された場合、許可は無効になります。

### Important

同じアカウント内の ID ベースのポリシーまたはリソースベースのポリシーの一方がリクエストを許可し、他方が許可しない場合でも、リクエストは許可されます。

Carlos のユーザー名が carlossalazar で、ファイルを Amazon S3 バケット carlossalazar-logs に保存するとします。

また、次のポリシーを IAM ユーザー carlossalazar にアタッチするとします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3ListRead",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3::*:"
    },
    {
      "Sid": "AllowS3Self",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::carlossalazar/*",
        "arn:aws:s3:::carlossalazar"
      ]
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": "arn:aws:s3::*:log*"
    }
  ]
}
```

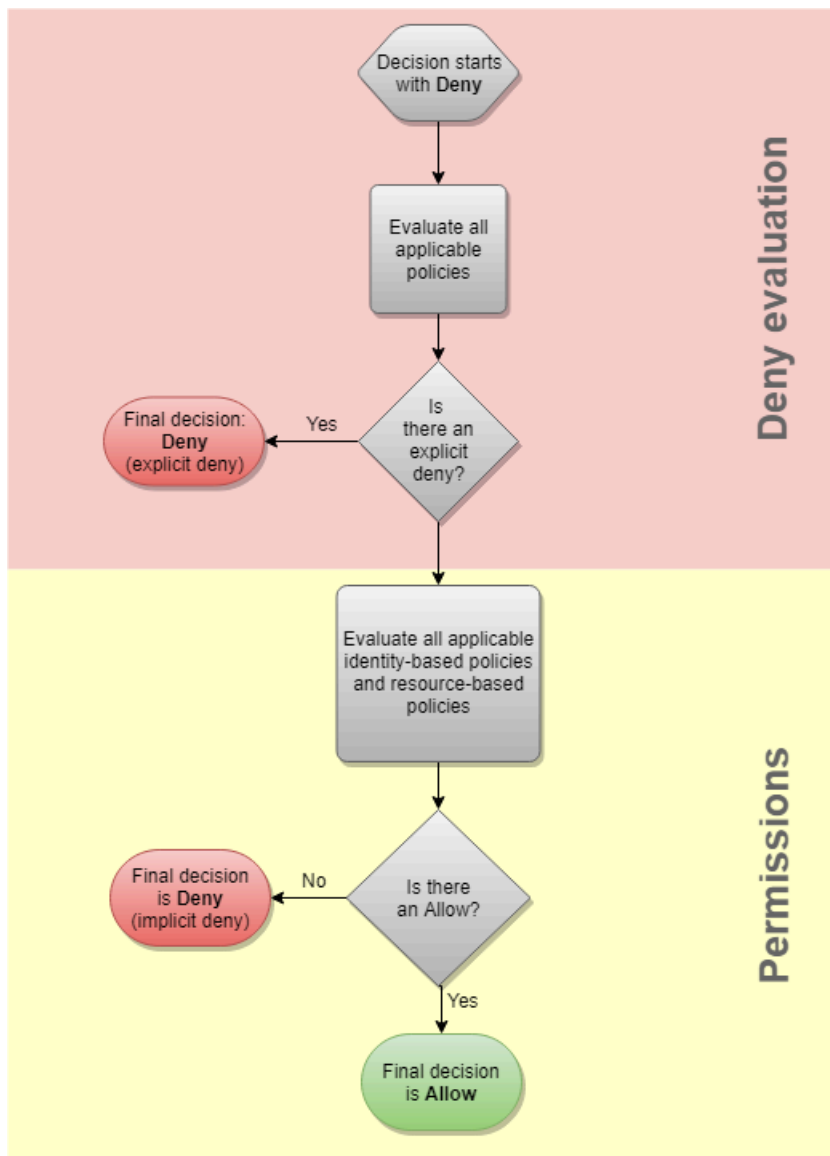
このポリシーの AllowS3ListRead ステートメントでは、アカウント内のすべてのバケットを一覧表示することを Carlos に許可します。AllowS3Self ステートメントでは、Carlos のユーザー名と同じ名前のバケットに対するフルアクセスを Carlos に許可します。DenyS3Logs ステートメントでは、名前に log が含まれている S3 バケットへのアクセスを Carlos に拒否します。

さらに、次のリソースベースのポリシー (バケットポリシー) を `carlossalazar` バケットにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/carlossalazar"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::carlossalazar/*",
        "arn:aws:s3:::carlossalazar"
      ]
    }
  ]
}
```

このポリシーでは、`carlossalazar` ユーザーのみが `carlossalazar` バケットにアクセスできることを指定します。

Carlos がファイルを `carlossalazar-logs` バケットに保存することをリクエストすると、AWS はこのリクエストに適用されるポリシーを決定します。この例では、アイデンティティベースのポリシーとリソースベースのポリシーのみが適用されます。これらは両方ともアクセス許可ポリシーです。アクセス許可の境界は適用されないため、評価ロジックは次のロジックに限定されます。



AWS は、最初にリクエストのコンテキストに適用される Deny ステートメントの有無を確認します。アイデンティティベースのポリシーでは、Carlos に対してログ記録用の S3 バケットへのアクセスが明示的に拒否されるため、該当するステートメントが見つかります。Carlos はアクセスが拒否されます。

彼は間違いに気が付いて、ファイルを carlossalazar バケットに保存しようとしています。AWS は Deny ステートメントを探しますが、1 つも見つかりません。次に、アクセス許可ポリシーを確認します。ID ベースのポリシーとリソースベースのポリシーの両方で、リクエストが許可されます。従って、AWS でリクエストが許可されます。どちらでもステートメントが明示的に拒否された場合、リクエストは拒否されます。いずれかのポリシータイプでリクエストが許可され、もう一方では許可されない場合でも、リクエストは許可されます。

## 明示的な拒否と暗黙的な拒否の違い

該当するポリシーに Deny ステートメントが含まれている場合、リクエストは明示的に拒否されます。リクエストに適用されるポリシーに Allow ステートメントと Deny ステートメントが含まれている場合は、Deny ステートメントが Allow ステートメントより優先されます。リクエストは明示的に拒否されます。

該当する Deny ステートメントがなく、該当する Allow ステートメントもない場合は、暗黙的な拒否が発生します。IAM プリンシパルは、デフォルトでアクセスが拒否されるため、アクションを実行するには明示的に許可を受ける必要があります。そうでない場合は、アクセスは暗黙的に拒否されます。

承認戦略を設計する場合、プリンシパルのリクエストを成功させるには、作成するポリシーに Allow ステートメントを含める必要があります。ただし、明示的な拒否と暗黙的な拒否の任意の組み合わせを選択できます。

例えば、許可されたアクション、暗黙的に拒否されたアクション、および明示的に拒否されたアクションを含む、次のポリシーを作成できます。AllowGetList ステートメントは、Get または List プレフィックスで始まる IAM アクションに対して、読み取り専用のアクセスを許可します。iam:CreatePolicy など、IAM の他のすべてのアクションは、暗黙的に拒否されます。DenyReports ステートメントは、iam:GetOrganizationsAccessReport などの Report サフィックスを含むアクションへのアクセスを拒否することで、IAM レポートへのアクセスを明示的に拒否します。誰かがこのプリンシパルに別のポリシーを追加して、iam:GenerateCredentialReport などの IAM レポートへのアクセス許可を付与した場合は、この明示的拒否のために、レポート関連のリクエストは引き続き拒否されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetList",
      "Effect": "Allow",
      "Action": [
        "iam:Get*",
        "iam:List*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DenyReports",
      "Effect": "Deny",
```

```
        "Action": "iam:*Report",
        "Resource": "*"
    }
]
}
```

## クロスアカウントポリシーの評価論理

1つのアカウントのプリンシパルが別のアカウントのリソースにアクセスすることを許可できます。これはクロスアカウントアクセスと呼ばれます。クロスアカウントアクセスを許可する場合、プリンシパルが存在するアカウントを信頼されたアカウントと呼びます。リソースが存在するアカウントは、信頼するアカウントです。

クロスアカウントアクセスを許可するには、共有するリソースにリソースベースのポリシーをアタッチします。また、リクエストのプリンシパルとして機能する ID に ID ベースポリシーをアタッチする必要があります。信頼するアカウントのリソースベースのポリシーは、リソースにアクセスできる信頼されるアカウントのプリンシパルを指定する必要があります。アカウント全体を指定することも、その IAM ユーザー、フェデレーテッドユーザー、IAM ロール、または引き受けたロールセッションを指定することもできます。AWS サービスをプリンシパルとして指定することもできます。詳細については、「[プリンシパルの指定](#)」を参照してください。

プリンシパルの ID ベースのポリシーは、信頼するサービスにあるリソースへのリクエストされたアクセスを許可する必要があります。これを行うには、リソースの ARN を指定するか、すべてのリソースへのアクセスを許可します (\*)。

IAM では、リソースベースのポリシーをロールにアタッチして、他のアカウントのプリンシパルがその IAM ロールを引き受けることを許可できます。ロールのリソースベースのポリシーは、ロールの信頼ポリシーと呼ばれます。そのロールを引き受けると、許可されたプリンシパルは結果として生じる一時的な認証情報を使用して、アカウント内の複数のリソースにアクセスできます。このアクセスは、ロールの ID ベースのアクセス許可ポリシーで定義されます。ロールを使用したクロスアカウントアクセスの許可と、他のリソースベースのポリシーを使用したクロスアカウントアクセスの許可の違いについては、「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。

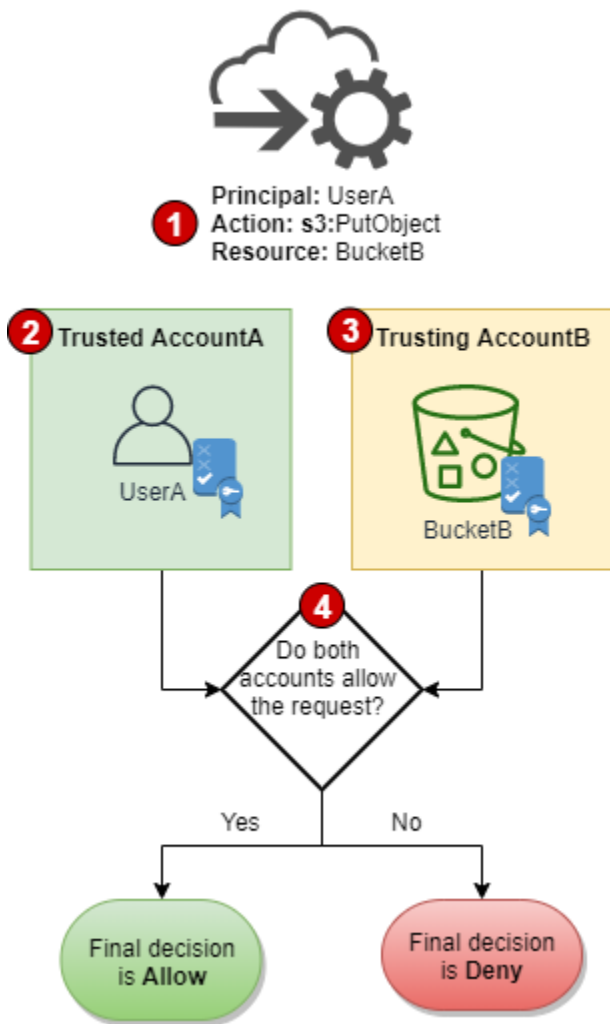
### Important

ポリシー評価ロジックに影響を与えるサービスもあります。たとえば、AWS Organizations は、1つ以上のアカウントのプリンシパルに適用できる[サービスコントロールポリシー](#)をサポートします。AWS Resource Access Manager は、プリンシパルが共有されるリソースに対して実行できるアクションを制御する[ポリシーフラグメント](#)をサポートします。

## クロスアカウントリクエストが許可されているかどうかを確認

クロスアカウントリクエストの場合、信頼済み AccountA のリクエストには ID ベースのポリシーが必要です。このポリシーでは、信頼する AccountB のリソースへのリクエストを許可する必要があります。また、AccountB のリソースベースのポリシーを使用して、AccountA のリクエストによるリソースへのアクセスを許可する必要があります。

クロスアカウントリクエストを行うと、AWS は 2 つの評価を実行します。AWS は、信頼するアカウントのリクエストと信頼されるアカウントを評価します。1 つのアカウント内でリクエストが評価される方法の詳細については、「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。リクエストは、両方の評価が Allow の決定を返す場合にのみ許可されます。



1. あるアカウントのプリンシパルが別のアカウントのリソースにアクセスするためのリクエストを行う場合のリクエストが、クロスアカウントリクエストです。
2. リクエスト元のプリンシパルは、信頼されたアカウント (AccountA) に存在します。AWS がこのアカウントを評価すると、ID ベースのポリシー、および ID ベースのポリシーを制限できるす



すべてのポリシーがチェックされます。詳細については、「[単一アカウント内のポリシーを評価する](#)」を参照してください。

3. リクエストされたリソースは、信頼するアカウント (AccountB) に存在します。AWS は、このアカウントを評価するときに、リクエストされたリソースにアタッチされているリソースベースのポリシーと、リソースベースのポリシーを制限できるすべてのポリシーがチェックされます。詳細については、「[単一アカウント内のポリシーを評価する](#)」を参照してください。
4. AWS は、両方のアカウントポリシー評価でリクエストが許可されている場合にのみリクエストを許可します。

### クロスアカウントポリシー評価の例

次の例は、あるアカウントのユーザーが、別のアカウントのリソースベースのポリシーによってアクセス許可が付与されるシナリオを示しています。

Carlos は、アカウント 111111111111 で carlossalazar という名前の IAM ユーザーを持つ開発者であるとしてします。アカウント 222222222222 の Production-logs Amazon S3 バケットにファイルを保存したいと考えています。

また、次のポリシーを IAM ユーザー carlossalazar にアタッチするとしてします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3ListRead",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "AllowS3ProductionObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object*",
      "Resource": "arn:aws:s3:::Production/*"
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
```

```
        "arn:aws:s3::*log*",
        "arn:aws:s3::*log/*"
    ]
}
]
```

このポリシーの AllowS3ListRead ステートメントでは、Amazon S3 内のすべてのバケットを一覧表示することを Carlos に許可します。この AllowS3ProductionObjectActions ステートメントにより、Carlos に Production バケットへのフルアクセスが許可されます。DenyS3Logs ステートメントでは、名前に log が含まれている S3 バケットへのアクセスを Carlos に拒否します。また、これらのバケット内のすべてのオブジェクトへのアクセスを拒否します。

さらに、次のリソースベースのポリシー (バケットポリシー) を 222222222222 アカウントの Production バケットにアタッチします。

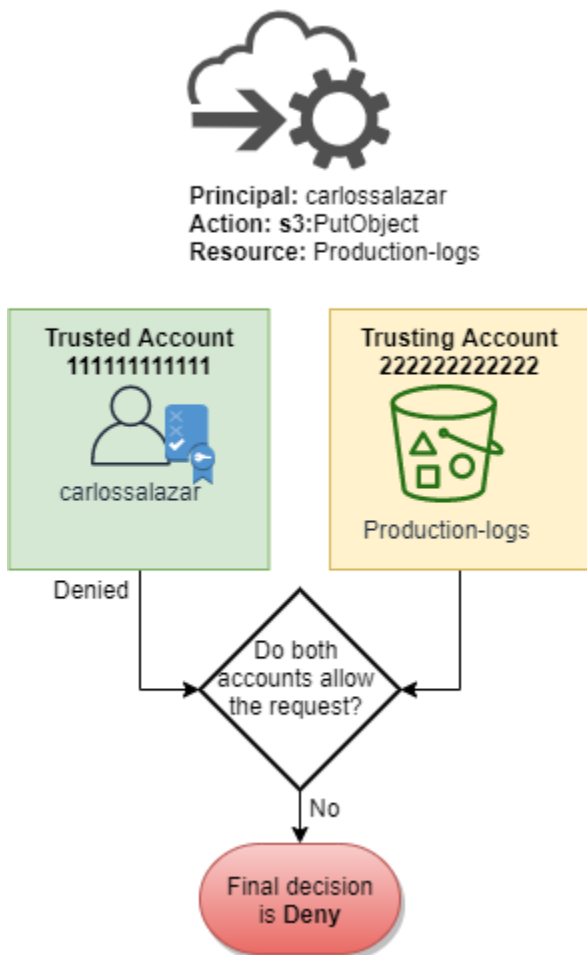
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "s3:PutObject*",
        "s3:ReplicateObject",
        "s3:RestoreObject"
      ],
      "Principal": { "AWS": "arn:aws:iam::111111111111:user/carlossalazar" },
      "Resource": "arn:aws:s3::*Production/*"
    }
  ]
}
```

このポリシーにより、carlossalazar ユーザーは Production バケット内のオブジェクトにアクセスできます。バケット内のオブジェクトを作成および編集することはできますが、削除することはできません。バケット自体を管理することはできません。

Carlos がファイルを Production-logs バケットに保存することをリクエストすると、AWS はこのリクエストに適用されるポリシーを決定します。この場合、アカウント 111111111111 に適用される唯一のポリシーは、carlossalazar ユーザーにアタッチされた ID ベースのポリシーです。アカウント 222222222222 には、Production-logs バケットにアタッチされたリソースベースの

ポリシーはありません。AWS がアカウント 111111111111 を評価するとき、Deny の決定を返します。これは、ID ベースのポリシーの DenyS3Logs ステートメントが、ログバケットへのアクセスを明示的に拒否するためです。1 つのアカウント内でリクエストが評価される方法の詳細については、「[アカウント内でのリクエストの許可または拒否の決定](#)」を参照してください。

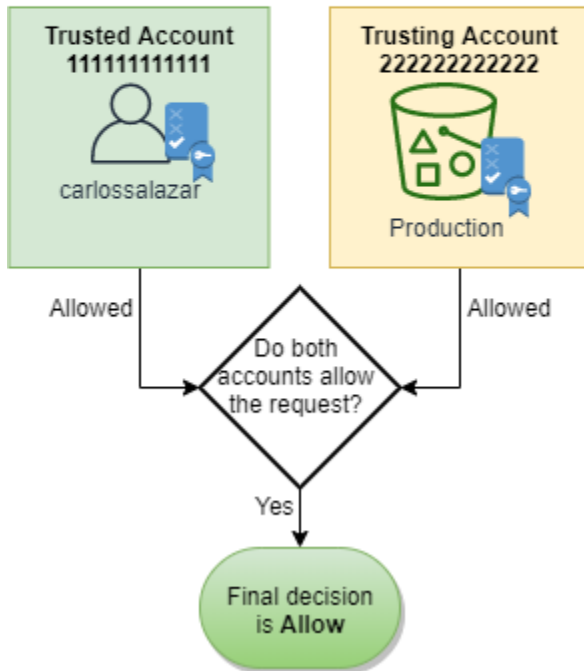
リクエストはいずれかのアカウントで明示的に拒否されるため、最終的な決定はリクエストを拒否することです。



Carlos は、その後、間違いに気づき、Production バケットにファイルを保存しようとするとしてします。AWS は、最初にアカウント 111111111111 をチェックし、リクエストが許可されるかどうかを判断します。ID ベースのポリシーのみが適用され、リクエストが許可されます。次に、AWS はアカウント 222222222222 をチェックします。Production バケットにアタッチされたリソースベースのポリシーのみが適用され、リクエストが許可されます。両方のアカウントがリクエストを許可するため、最終的な決定はリクエストを許可することです。



Principal: carlossalazar  
Action: s3:PutObject  
Resource: Production



## IAM JSON ポリシー言語の文法

このページでは、IAM で JSON ポリシーを作成する際に使用する言語の正式な文法を示します。ポリシーを構築および検証する方法を理解できるようにするために、この文法を示しています。

ポリシーの例については、以下のトピックを参照してください。

- [IAM でのポリシーとアクセス許可](#)
- [IAM アイデンティティベースのポリシーの例](#)
- 「Amazon EC2 ユーザーガイド」の「[Amazon EC2 コンソールで機能するサンプル ポリシー](#)」と「[AWS CLI \(Amazon EC2 CLI\) または AWS SDK で使用するサンプルポリシー](#)」。
- [Amazon Simple Storage Service ユーザーガイド](#)のバケットポリシーの例と[ユーザーポリシーの例](#)。

その他の AWS サービスで使用されるポリシーの例については、各サービスのドキュメントを参照してください。

## トピック

- [ポリシー言語と JSON](#)
- [この文法で使用される表記規則](#)
- [文法](#)
- [ポリシーの文法に関する注意事項](#)

## ポリシー言語と JSON

ポリシーは、JSON 形式で表されます。JSON ポリシーを作成または編集するときに、IAM はポリシー検証を実行し、効果的なポリシーを作成するのに役立ちます。IAM は JSON 構文エラーを識別します。一方、IAM Access Analyzer は、ポリシーをさらに絞り込むのに役立つ推奨事項を含む追加のポリシーチェックを提供します。ポリシーの検証の詳細については、「[IAM ポリシーの検証](#)」を参照してください。。IAM Access Analyzer ポリシーチェックと実用的な推奨事項の詳細については、「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。

このドキュメントでは、有効な JSON の構成内容については詳しく説明しません。ただし、基本的な JSON のルールをいくつか紹介します。

- 個々のエンティティ間の空白文字は許可されています。
- 値は引用符で囲みます。数値やブール値の場合、引用符は省略できます。
- 多くの要素 (たとえば、`action_string_list` や `resource_string_list`) で、値として JSON 配列を使用できます。配列では 1 つまたは複数の値を使用できます。複数の値が含まれている場合、配列は次の例のように、角括弧 ([ と ]) で囲まれ、カンマで区切られます。

```
"Action" : ["ec2:Describe*", "ec2:List*"]
```

- 基本的な JSON のデータ型 (ブール値、数値、文字列) は [RFC 7159](#) で定義されています。

## この文法で使用される表記規則

この文法では、以下の表記規則が使用されます。

- 以下の文字は JSON のトークンであり、ポリシーに含まれます。

```
{ } [ ] " , :
```

- 以下の文字は文法の特殊文字であり、ポリシーには含まれません。

= < > ( ) |

- 要素で複数の値を使用できる場合、繰り返し値、カンマ区切り文字、および省略符号 (...) で示されます。例:

```
[<action_string>, <action_string>, ...]
```

```
<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
```

複数の値が許可されている場合、値が 1 つだけ含むことも有効です。値が 1 つだけである場合、末尾のカンマは省略する必要があります。要素で配列 ([と] で表される) を使用する場合、含まれている値が 1 つだけであるときは、角括弧を省略できます。例:

```
"Action": [<action_string>]
```

```
"Action": <action_string>
```

- 要素に続く疑問符 (?) は、その要素が省略できることを示します。例:

```
<version_block?>
```

ただし、省略可能な要素の詳細については、文法の一覧の後に続く注意事項を参照してください。

- 要素間の縦棒 (|) は選択肢を示します。この文法では、括弧は選択肢の範囲を定義します。例:

```
("Principal" | "NotPrincipal")
```

- リテラル文字列にする必要がある要素は二重引用符 (") で囲まれます。例:

```
<version_block> = "Version" : ("2008-10-17" | "2012-10-17")
```

その他の注意事項については、文法の一覧の後にある「[ポリシーの文法に関する注意事項](#)」を参照してください。

## 文法

以下の一覧では、ポリシー言語の文法について説明します。この一覧で使用されている表記規則については、前のセクションを参照してください。詳細については、後に示されている注意事項を参照してください。

**Note**

この文法では、2008-10-17 と 2012-10-17 のバージョンでマークされているポリシーについて説明します。Version ポリシー要素は、ポリシーバージョンとは異なります。Version ポリシー要素は、ポリシー内で使用され、ポリシー言語のバージョンを定義します。一方で、ポリシーバージョンは、IAM でカスタマー管理ポリシーを変更すると作成されます。変更されたポリシーによって既存のポリシーが上書きされることはありません。代わりに、IAM は管理ポリシーの新しいバージョンを作成します。Version ポリシー要素の詳細については、「[IAM JSON ポリシー要素Version](#)」を参照してください。ポリシーのバージョンの詳細については、「[the section called "IAM ポリシーのバージョンング"](#)」を参照してください。

```

policy = {
  <version_block?>
  <id_block?>
  <statement_block>
}

<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
  <sid_block?>,
  <principal_block?>,
  <effect_block>,
  <action_block>,
  <resource_block>,
  <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

```

```
<principal_map_entry> = ("AWS" | "Federated" | "Service" | "CanonicalUser") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    : ("*" | <resource_string> | [<resource_string>, <resource_string>, ...])

<condition_block> = "Condition" : { <condition_map> }
<condition_map> = {
    <condition_type_string> : { <condition_key_string> : <condition_value_list> },
    <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = (<condition_value_string> | <condition_value_string> |
    <condition_value_string>)
```

## ポリシーの文法に関する注意事項

- 1つのポリシーにステートメントの配列を含めることができます。
- ポリシーの最大サイズは 2,048 ~ 10,240 文字で、ポリシーがアタッチされるエンティティによって異なります。詳細については、「[IAM と AWS STS クォータ](#)」を参照してください。ポリシーのサイズの計算には、空白文字は含まれません。
- 個々の要素に、同じキーの複数のインスタンスを含めることはできません。たとえば、同じステートメントに Effect ブロックを 2 回含めることはできません。
- ブロックは任意の順序で記述できます。たとえば、ポリシー内で version\_block が id\_block の後にあってもかまいません。同様に、ステートメント内で effect\_block、principal\_block、action\_block は任意の順序で記述できます。
- リソースベースのポリシーでは、id\_block はオプションです。アイデンティティベースのポリシーに含めることはできません。
- principal\_block 要素は、リソースベースのポリシー (Amazon S3 のバケットポリシーなど) および IAM ロールの信頼ポリシーでは必須です。アイデンティティベースのポリシーに含めることはできません。
- Amazon S3 バケットポリシーの principal\_map 要素には、CanonicalUser ID を含めることができます。ほとんどのリソースベースのポリシーは、このマッピングをサポートしていません。



バケットポリシーでの正規ユーザー ID の使用の詳細については、[Amazon Simple Storage Service ユーザーガイド](#)の「ポリシーでのプリンシパルの指定の」を参照してください。

- 各文字列値

( `policy_id_string`、`sid_string`、`principal_id_string`、`action_string`、`resource_str` および文字列バージョンの `condition_value` ) では、それぞれの最小長と最大長の制限、特定の許容値、または必要な内部形式が決まっている場合があります。

## 文字列値に関する注意事項

このセクションでは、ポリシー内の複数の要素で使用される文字列値に関する追加情報を示します。

### `action_string`

サービス名前空間、コロン、およびアクション名で構成されます。アクション名にはワイルドカードを含めることができます。例:

```
"Action": "ec2:StartInstances"

"Action": [
  "ec2:StartInstances",
  "ec2:StopInstances"
]

"Action": "cloudformation:*"

"Action": "*"

"Action": [
  "s3:Get*",
  "s3:List*"
]
```

### `policy_id_string`

ポリシーに関する情報をまとめて含める方法を提供します。Amazon SQS や Amazon SNS などの一部のサービスでは、Id 要素は予約された方法で使用されます。個々のサービスで制限されていない限り、`policy_id_string` には空白文字を含めることができます。一部のサービスでは、この値が AWS アカウント内で一意である必要があります。

**Note**

`id_block` は、リソースベースのポリシーでは使用できますが、アイデンティティベースのポリシーでは使用できません。

長さに制限はありません。ただし、この文字列はポリシー全体の長さにカウントされ、全体の長さは制限されます。

```
"Id": "Admin_Policy"
```

```
"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

**sid\_string**

個々のステートメントに関する情報含める方法を提供します。IAM ポリシーでは、基本的な英数字 (A~Z、a~z、0~9) のみを Sid 値に使用できます。リソースポリシーをサポートするその他の AWS サービスでは、Sid 値に関して他の要件がある場合があります。例えば、一部のサービスには AWS アカウント 内でこの値が一意であること、スペースなどの追加の文字を Sid 値で使用できることです。

```
"Sid": "1"
```

```
"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

**principal\_id\_string**

AWS アカウントの [Amazon リソースネーム \(ARN\)](#)、IAM ユーザー、IAM ロール、フェデレーションユーザー、または引き受けたロールユーザーを使用してプリンシパルを指定する方法を提供します。AWS アカウントでは、完全な ARN の代わりに短縮形 `AWS:accountnumber` を使用できます。AWS のサービス、割り当てられたロールなどを含むすべてのオプションについては、「[プリンシパルの指定](#)」を参照してください。

"すべてのユーザー/匿名ユーザー" を指定するには、\* のみを使用できることに注意してください。これを使用して名前または ARN の一部を指定することはできません。

**resource\_string**

多くの場合、[Amazon リソースネーム \(ARN\)](#) で構成されます。

```
"Resource": "arn:aws:iam::123456789012:user/Bob"

"Resource": "arn:aws:s3:::examplebucket/*"
```

## condition\_type\_string

StringEquals、StringLike、NumericLessThan、DateGreaterThanOrEquals、Bool、BinaryE など、テストされる条件のタイプを指定します。条件タイプの詳細なリストについては、「[IAM JSON ポリシー要素: 条件演算子](#)」を参照してください。

```
"Condition": {
  "NumericLessThanEquals": {
    "s3:max-keys": "10"
  }
}

"Condition": {
  "Bool": {
    "aws:SecureTransport": "true"
  }
}

"Condition": {
  "StringEquals": {
    "s3:x-amz-server-side-encryption": "AES256"
  }
}
```

## condition\_key\_string

条件が満たされているかどうかを判断するために値がテストされる条件キーを識別します。AWS は、AWS、aws:PrincipalType、aws:SecureTransport、aws:userid を含むすべてのサービスで使用できる条件キーのセットを定義します。

AWS の条件キーのリストについては、「[AWS グローバル条件コンテキストキー](#)」を参照してください。サービスに固有の条件キーについては、以下のようなそのサービスのドキュメントを参照してください。

- Amazon Simple Storage Service ユーザーガイドの[ポリシーでの条件の指定](#)
- 「Amazon EC2 ユーザーガイド」の「[Amazon EC2 の IAM ポリシー](#)」。

```
"Condition":{
```

```
"Bool": {
  "aws:SecureTransport": "true"
}

"Condition": {
  "StringNotEquals": {
    "s3:x-amz-server-side-encryption": "AES256"
  }
}

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/purpose": "test"
  }
}
```

### condition\_value\_string

条件が満たされているかどうかを判断する `condition_key_string` の値を識別します。条件タイプの有効な値の一覧については、「[IAM JSON ポリシー要素: 条件演算子](#)」を参照してください。

```
"Condition":{
  "ForAnyValue:StringEquals": {
    "dynamodb:Attributes": [
      "ID",
      "PostDateTime"
    ]
  }
}
```

## AWSジョブ機能の管理ポリシー

[最小権限を付与する](#) ポリシーを使用するか、タスクの実行に必要な権限のみを付与することをお勧めします。最小限の権限を付与する最も安全な方法は、チームに必要な権限のみを使用してカスタムポリシーを作成することです。必要に応じて、チームがより多くの権限を要求できるようにプロセスを作成する必要があります。チームに必要な許可のみを提供する [IAM カスタマーマネージドポリシー](#) を作成するには、時間と専門知識が必要です。

[AWS マネージドポリシー](#) を使用して、IAM ID (ユーザー、ユーザーのグループ、およびロール) へのアクセス許可の追加を開始。AWS 管理ポリシーは一般的な使用例をカバーし、AWS アカウント

で利用できます。AWS 管理ポリシーは、最小特権のアクセス許可を付与しません。プリンシパルにジョブに必要な以上のアクセス許可を付与すると、セキュリティ上のリスクを考慮する必要があります。

AWS管理ポリシー（ジョブ機能を含む）を任意の IAM ID にアタッチすることができます。最小権限権限に切り替えるには、AWS Identity and Access ManagementAnalyzerにアクセスしてプリンシパルをモニタリングするAWS管理ポリシーを使用します。どの権限を使用しているかを学習したら、カスタムポリシーを作成するか、チームに必要な権限のみを持つポリシーを生成できます。これは安全性は低くなりますが、チームが AWS をどのように使用しているかを学習するにつれて柔軟性が高まります。

AWSジョブ機能の管理ポリシーは、IT 業界の一般的なジョブ機能と密接に連携するように設計されています。これらのポリシーを使用すると、特定のジョブ機能を持つ人によるタスクの実行に必要な権限を簡単に付与することができます。これらのポリシーは、多くのサービスの権限を一つのポリシーに統合しているため、権限が様々なポリシーに分散している場合に比べてより作業しやすくなっています。

### ロールを使用してサービスを連動する

他の AWS サービスにある機能の活用を促すために IAM サービスのロールを使用するポリシーもあります。これらのポリシーは、iam:passrole にアクセス許可を付与します。これによりポリシーが適用されるユーザーは AWS のサービスにロールを渡すことができます。このロールは、お客様に代わってアクションを実行するための IAM アクセス許可を AWS のサービスに委任します。

必要に応じてロールを作成する必要があります。たとえば、ネットワーク管理者のポリシーでは、ポリシーを利用するユーザーは「flow-logs-vpc」というロールを Amazon CloudWatch サービスに渡すことができます。CloudWatch は、そのロールを使用して、ユーザーが作成した VPC の IP トラフィックをログに記録し、キャプチャします。

セキュリティのベスト プラクティスに従うは、渡すことができる有効なロール名を制限するフィルタを、ジョブ機能のポリシーに含めます。これは不要な権限の付与を避けるのに役立ちます。ご自分のユーザーにオプション サービス ロールが必要な場合は、ポリシーで指定されている命名規則に従うロールを作成する必要があります。その後、ロールに権限を付与します。この処理が終わると、ユーザーは、ロールを使用するサービスを設定して、そのロールが提供する権限を付与することができます。

次のセクションでは、各ポリシーの名前は AWS Management Console のポリシー詳細ページへのリンクです。ここでは、ポリシードキュメントを表示し、そのポリシーによって付与されるアクセス許可を確認できます。

## 管理者ジョブ関数

AWS 管理ポリシー名: [AdministratorAccess](#)

ユースケース: このユーザーはフルアクセスが許可され、AWS のあらゆるサービスおよびリソースにアクセス許可を委任できます。

ポリシーの更新 AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、アカウントの AWS のすべてのサービスおよびリソースに対するすべてのアクションを許可します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[AdministratorAccess](#)」を参照してください。

### Note

IAM ユーザーまたはロールが、このポリシーのアクセス許可を使用して AWS Billing and Cost Management コンソールにアクセスするには、まず IAM ユーザーおよびロールのアクセスをアクティブ化する必要があります。そのためには、[請求コンソールへのアクセスの委任に関するチュートリアル](#)のステップ 1 の手順に従ってください。

## 請求ジョブ関数

AWS 管理ポリシー名: [Billing](#)

ユースケース: このユーザーは請求情報の確認、支払いの設定、支払いの承認を行う必要があります。ユーザーは、AWS のサービス全体の累計されたコストをモニタリングできます。

ポリシーの更新 AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、請求、コスト、支払い方法、予算、レポートを管理するためのフルアクセス許可を付与します。その他のコスト管理ポリシーの例については、「AWS Billing and Cost Management ユーザーガイド」の「[AWS Billing ポリシーの例](#)」を参照してください。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[Billing](#)」を参照してください。

**Note**

IAM ユーザーまたはロールが、このポリシーのアクセス許可を使用して AWS Billing and Cost Management コンソールにアクセスするには、まず IAM ユーザーおよびロールのアクセスをアクティブ化する必要があります。そのためには、[請求コンソールへのアクセスの委任に関するチュートリアル](#)のステップ 1 の手順に従ってください。

## データベース管理者のジョブ機能

AWS 管理ポリシー名: [DatabaseAdministrator](#)

ユースケース: このユーザーは AWS クラウドのデータベースのセットアップ、設定、メンテナンスを行います。

ポリシーの更新 AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、データベースの作成、設定、メンテナンスを行うためのアクセス許可を付与します。これは、へのアクセスが含まれていますAWSAmazon DynamoDB、Amazon Relational Database Service (RDS)、Amazon Redshift などのデータベースサービスを利用できます。このポリシーがサポートしているデータベースサービスの詳細なリストに対するポリシーを表示します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[DatabaseAdministrator](#)」を参照してください。

このジョブ機能ポリシーは、ロールを AWS サービスへ渡す機能をサポートしています。このポリシーは、次の表で示されるロールに対してのみ iam:PassRole アクションを許可します。詳細については、このトピックで後述する「[ロールの作成とポリシーのタッチ \(コンソール\)](#)」を参照してください。

### データベース管理者のジョブ機能のオプション IAM サービスロール

ユースケース	ロール名 (* はワイルドカードです)	選択するサービスロールの種類	この AWS 管理ポリシーを選択します。
ユーザーに RDS データベースのモニタリングを許可します	<a href="#">rds-monitoring-role</a>	拡張モニタリング用 Amazon RDS ロール	<a href="#">AmazonRDS EnhancedMonitoring Role</a>

ユースケース	ロール名 (* はワイルドカードです)	選択するサービスロールの種類	この AWS 管理ポリシーを選択します。
AWS Lambda に、データベースのモニタリングと外部データベースへのアクセスを許可します	<a href="#">rdbms-lambda-access</a>	Amazon EC2	<a href="#">AWSLambda_FullAccess</a>
Lambda が DynamoDB を使用して Amazon S3 および Amazon Redshift クラスターにファイルをアップロードすることを許可する	<a href="#">lambda_exec_role</a>	AWS Lambda	<a href="#">AWS ビッグデータブ</a> <a href="#">ログ</a> で定義されているように新しい管理ポリシーを作成します
Lambda 関数に、DynamoDB テーブルのトリガーとしての動作を許可します	<a href="#">lambda-dynamodb-*</a>	AWS Lambda	<a href="#">AWSLambda</a> <a href="#">DynamoDBExecutionRole</a>
Lambda 関数が VPC 内の Amazon RDS へのアクセスを許可する	<a href="#">lambda-vpc-execution-role</a>	<a href="#">AWS Lambda 開発者ガイド</a> に定義されているように、信頼ポリシーを適用したロールを作成します	<a href="#">AWSLambda</a> <a href="#">VPCAccessExecutionRole</a>
AWS Data Pipeline に、AWS へのアクセスを許可します	<a href="#">DataPipelineDefaultRole</a>	<a href="#">AWS Data Pipeline 開発者ガイド</a> に定義されているように、信頼ポリシーを適用したロールを作成します	AWS Data Pipeline のドキュメントには、このユースケースに必要なアクセス許可が記載されています。「 <a href="#">AWS Data Pipeline の IAM ロール</a> 」を参照してください。



ユースケース	ロール名 (* はワイルドカードです)	選択するサービスロールの種類	この AWS 管理ポリシーを選択します。
Amazon EC2 インスタンスで実行されるアプリケーションに、AWS リソースへのアクセスを許可します	<a href="#">DataPipelineDefaultResourceRole</a>	<a href="#">AWS Data Pipeline 開発者ガイド</a> に定義されているように、信頼ポリシーを適用したロールを作成します	<a href="#">AmazonEC2RoleforDataPipelineRole</a>

## データサイエンティストジョブ関数

AWS 管理ポリシー名: [DataScientist](#)

ユースケース: このユーザーは Hadoop ジョブおよびクエリを実行します。このユーザーは、データ分析やビジネスインテリジェンス用の情報にアクセスして分析も行います。

ポリシーの更新 AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、Amazon EMR クラスターでクエリを作成、管理、実行し、Amazon QuickSight などのツールでデータを分析するアクセス許可を付与します。このポリシーには、追加のデータサイエンティストサービス (AWS Data Pipeline、Amazon EC2、Amazon Kinesis、Amazon Machine Learning、および SageMaker に導入されている。このポリシーがサポートしているデータサイエンティストサービスの詳細なリストに対するポリシーを表示します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[DataScientist](#)」を参照してください。

このジョブ機能ポリシーは、ロールを AWS サービスへ渡す機能をサポートしています。一方のステートメントは、任意のロールを SageMaker に渡すことを許可します。別のステートメントは、次の表で示されるロールに対してのみ iam:PassRole アクションを許可します。詳細については、このトピックで後述する「[ロールの作成とポリシーのタッチ \(コンソール\)](#)」を参照してください。

## データサイエンティストのジョブ機能に対するオプション IAM サービスロール

ユースケース	ロール名 (* はワールドカードです)	選択するサービスロールの種類	選択する AWS 管理ポリシー
Amazon EC2 インスタンスの、クラスターに適したサービスおよびリソースへのアクセスを許可します。	<a href="#">EMR-EC2_DefaultRole</a>	EC2 の Amazon EMR	<a href="#">AmazonElasticMapReduceforEC2Role</a>
Amazon EMR アクセスを許可して Amazon EC2 サービスおよびクラスターのリソースにアクセスする	<a href="#">EMR_DefaultRole</a>	Amazon EMR	<a href="#">AmazonEMRServicePolicy_v2</a>
Kinesis Managed Service for Apache Flink を許可してストリーミングデータソースにアクセスする	<a href="#">kinesis-*</a>	<a href="#">AWS ビッグデータブログ</a> に定義されているように、信頼ポリシーを適用したロールを作成します。	ユースケースに応じて選択できる 4 つのオプションの概要については、「 <a href="#">AWS ビッグデータブログ</a> 」を参照してください。
AWS Data Pipeline に、AWS へのアクセスを許可します	<a href="#">DataPipelineDefaultRole</a>	<a href="#">AWS Data Pipeline 開発者ガイド</a> に定義されているように、信頼ポリシーを適用したロールを作成します	AWS Data Pipeline のドキュメントには、このユースケースに必要なアクセス許可が記載されています。「 <a href="#">AWS Data Pipeline の IAM ロール</a> 」を参照してください。
Amazon EC2 インスタンスで実行されるアプリケーションに、AWS リソースへのアクセスを許可します	<a href="#">DataPipelineDefaultResourceRole</a>	<a href="#">AWS Data Pipeline 開発者ガイド</a> に定義されているように、信頼ポリシー	<a href="#">AmazonEC2RoleforDataPipelineRole</a>

ユースケース	ロール名 (* はワールドカードです)	選択するサービス ロールの種類	選択する AWS 管理 ポリシー
		を適用したロール を作成します	

## 開発者パワーユーザージョブ機能

AWS 管理ポリシー名: [PowerUserAccess](#)

ユースケース: このユーザーはアプリケーション開発タスクを実行し、AWS 対応アプリケーションの開発をサポートするリソースとサービスを作成および設定できます。

ポリシーの更新 AWS: このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの説明: このポリシーの最初のステートメントでは、[NotAction](#) 要素を使用して、すべての AWS のサービスのすべてのアクションと、すべてのリソース (AWS Identity and Access Management、AWS Organizations、AWS Account Management を除く) のすべてのアクションを許可します。2 つめのステートメントでは、サービスにリンクされたロールを作成する IAM アクセス許可を付与します。これは、別のサービスのリソース (Amazon S3 バケットなど) にアクセスしなければならないサービスに必要です。また、ユーザーの組織に関する情報 (管理アカウントの E メールや組織の制限など) を表示する Organizations アクセス許可を付与します。このポリシーは、IAM、Organizations を制限しますが、IAM Identity Center が有効化されている場合には、ユーザーが、IAM Identity Center のすべてのアクションを実行できるようになります。また、アカウントに対してどの AWS リージョンが有効が無効かを表示するためのアカウント管理のアクセス許可も付与されます。

## ネットワーク管理者のジョブ機能

AWS 管理ポリシー名: [NetworkAdministrator](#)

ユースケース: このユーザーは AWS; ネットワークリソースの設定とメンテナンスを担当します。

ポリシーの更新: AWS は、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの説明:このポリシーは、Auto Scaling、Amazon EC2、AWS Direct Connect、Route 53、Amazon CloudFront、Elastic Load Balancing、AWS Elastic Beanstalk、Amazon SNS、CloudWatch、CloudWatch Logs、Amazon S3、IAM、Amazon Virtual PrivateCloud でネットワークリソースを作成および維持するためのアクセス許可を付与します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[NetworkAdministrator](#)」を参照してください。

このジョブ機能には、ロールを AWS サービスへ渡す機能が必要です。このポリシーは、次の表で指定されたロールに対してのみ `iam:GetRole` と `iam:PassRole` を付与します。詳細については、このトピックで後述する「[ロールの作成とポリシーのアタッチ \(コンソール\)](#)」を参照してください。

#### ネットワーク管理者のジョブ機能のオプション IAM サービスロール

ユースケース	ロール名 (* はワールドカードです)	選択するサービスロールの種類	選択する AWS 管理ポリシー
Amazon VPC に、ユーザーに代わって CloudWatch Logs でログを作成および管理し、VPC を出入りする IP トラフィックをモニタリングするのを許可します	<a href="#">flow-logs-*</a>	<a href="#">Amazon VPC ユーザーガイド</a> に定義されているように、信頼ポリシーを適用したロールを作成します。	このユースケースには、既存の AWS 管理ポリシーがありませんが、ドキュメントには必要な権限が記載されています。 <a href="#">Amazon VPC User Guide</a> を参照してください。

## 読み取り専用アクセス

AWS 管理ポリシー名: [ReadOnlyAccess](#)

ユースケース: このユーザーには、AWS アカウント のすべてのリソースへの読み取り専用アクセス権が必要です。

ポリシーの更新 AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの説明: このポリシーは、リソースとその属性を一覧表示、取得、説明、その他の表示するためのアクセス許可を付与します。これは、作成や削除のような突然変異機能が含まれていません。このポリシーには、AWS や AWS Identity and Access Management などのセキュリティ関連の AWS Billing and Cost Management サービスへの読み取り専用アクセスが含まれています。このポリシーがサポートしているデータベースサービスの詳細なリストに対するポリシーを表示します。

## セキュリティ監査ジョブ機能

AWS 管理ポリシー名: [SecurityAudit](#)

ユースケース: このユーザーはセキュリティ要件の遵守についてアカウントをモニタリングします。このユーザーは、潜在的なセキュリティ侵害や悪意のある行為を調査するためのログおよびイベントにアクセスできます。

ポリシーの更新: AWS は、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、AWS の多数のサービスの設定データを表示し、それらのログを確認するアクセス許可を付与します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[SecurityAudit](#)」を参照してください。

## Support ユーザジョブ機能

AWS 管理ポリシー名: [SupportUser](#)

ユースケース: このユーザーは AWS サポートに連絡し、サポートケースを作成して、既存のケースの状態を確認します。

ポリシーの更新: AWS は、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、AWS Support ケースを作成および更新するアクセス許可を付与します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[SupportUser](#)」を参照してください。

## システム管理者のジョブ機能

AWS 管理ポリシー名: [SystemAdministrator](#)

ユースケース: このユーザーは開発運用リソースのセットアップおよびメンテナンスを行います。

ポリシーの更新: AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの説明: このポリシーでは、AWS CloudTrail、Amazon CloudWatch、AWS CodeCommit、AWS CodeDeploy、AWS Config、AWS Directory Service、Amazon EC2、AWS Identity and Access Management、AWS Key Management Service、AWS Lambda、Amazon RDS、Route 53、Amazon S3、Amazon SES、Amazon SQS、AWS Trusted Advisor、Amazon VPC などのさまざまな AWS サービスにわたってリソースを作成および維持するためのアクセス許可を付与します。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[SystemAdministrator](#)」を参照してください。

このジョブ機能には、ロールを AWS サービスへ渡す機能が必要です。このポリシーは、次の表で指定されたロールに対してのみ `iam:GetRole` と `iam:PassRole` を付与します。詳細については、このトピックで後述する「[ロールの作成とポリシーのアタッチ \(コンソール\)](#)」を参照してください。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

#### システム管理者のジョブ機能のオプション IAM サービスロール

ユースケース	ロール名 (* はワイルドカードです)	選択するサービスロールの種類	選択する AWS 管理ポリシー
Amazon ECS クラスターの EC2 インスタンスで実行されるアプリケーションに Amazon ECS へのアクセスを許可します	<a href="#">ecr-sysadmin-*</a>	EC2 コンテナサービスの Amazon EC2 のロール	<a href="#">AmazonEC2ContainerServiceforEC2Role</a>
ユーザーがデータベースをモニタリングすることを許可します	<a href="#">rds-monitoring-role</a>	拡張モニタリング用 Amazon RDS ロール	<a href="#">AmazonRDSEnhancedMonitoringRole</a>
EC2 インスタンスで実行されるアプリケーションの、AWS リソースへのアクセスを許可します。	<a href="#">ec2-sysadmin-*</a>	Amazon EC2	「 <a href="#">Amazon EC2 ユーザーガイド</a> 」に示されている、S3 バケットへのアクセスを許可するロールのサンプル



ユースケース	ロール名 (* はワールドカードです)	選択するサービス ロールの種類	選択する AWS 管理 ポリシー
			ポリシー (必要に応じてカスタマイズ します)。
Lambda が DynamoDB Streams の読み取り、CloudWatch Logs への書き込みを許可する	<a href="#">lambda-sysadmin-*</a>	AWS Lambda	<a href="#">AWSLambdaDynamoDBExecutionRole</a>

## 閲覧専用ユーザージョブ関数

AWS 管理ポリシー名: [ViewOnlyAccess](#)

ユースケース: このユーザーは、サービスにわたるアカウントの AWS リソースや基本のメタデータのリストを表示できます。このユーザーは、クォータを超えるリソースコンテンツやメタデータを読み取ることや、リソース情報をリスト表示することはできません。

ポリシーの更新: AWSは、このポリシーを維持および更新します。このポリシーの変更履歴については、IAM コンソールでポリシーを表示し、ポリシーバージョンタブを選択します。ジョブ関数のポリシーの更新の詳細については、「[ジョブ機能の AWS 管理ポリシー](#)」を参照してください。

ポリシーの詳細: このポリシーでは、AWS サービスのリソースに対する List\*、Describe\*、Get\*、View\*、および Lookup\* アクセス許可を付与します。このポリシーに含まれる各サービスのアクションを確認するには、「[ViewOnlyAccess](#)」を参照してください。マネージドポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「[ViewOnlyAccess](#)」を参照してください。

## ジョブ機能の AWS 管理ポリシー

これらのポリシーはすべて AWS によって維持され、AWS サービスによって追加された新しいサービスや新機能のサポートを含めるよう最新の状態に保たれます。これらのポリシーは、お客様が変更することはできません。ポリシーのコピーを作成してそのコピーを変更できますが、AWS での新しいサービスや API オペレーションの導入時に、そのコピーは自動的に更新されません。

ジョブ機能ポリシーの場合、IAM コンソールでバージョン履歴と各更新の日時を表示できます。これを行うには、このページのリンクを使用して、ポリシーの詳細を表示します。次に、[ポリシーの

バージョン] タブをクリックして、バージョンを表示します。このページには、ポリシーの最後の 25 バージョンが表示されます。ポリシーのすべてのバージョンを表示するには、[get-policy-version](#) AWS CLI コマンドまたは [GetPolicyVersion](#) API オペレーションを呼び出します。

#### Note

カスタマー管理ポリシーには最大 5 つのバージョンを含めることができますが、AWS は AWS 管理ポリシーの完全なバージョン履歴を保持します。

## ロールの作成とポリシーのアタッチ (コンソール)

上記のいくつかのポリシーでは、ロールを利用して、AWS サービスがお客様に代わってオペレーションを実行するアクセス許可をそれらのサービスに付与しています。ジョブ機能ポリシーは、使用しなければならない正確なロール名、または、使用可能な名前前半を少なくとも含んでいるプレフィックスを指定します。これらのロールのいずれかを作成するには、次の手順のステップを実行します。

AWS のサービスのロールを作成するには (IAM コンソール)

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
3. 信頼できるエンティティタイプで、AWS のサービスを選択します。
4. [サービスまたはユースケース] でサービスを選択し、次にユースケースを選択します。ユースケースは、サービスに必要な信頼ポリシーを含める定義になります。
5. [Next] を選択します。
6. [アクセス許可ポリシー] では、オプションは選択したユースケースによって異なります。
  - サービスがロールのアクセス許可を定義している場合、アクセス許可ポリシーを選択することはできません。
  - 制限されたアクセス許可ポリシーのセットから選択します。
  - すべてのアクセス許可ポリシーから選択します。
  - アクセス許可ポリシーを選択せずに、ロールの作成後にポリシーを作成し、そのポリシーをロールにアタッチします。
7. (オプション) [アクセス許可の境界](#)を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。



- a. [アクセス許可の境界の設定] セクションを開き、[アクセス許可の境界を使用してロールのアクセス許可の上限を設定する] を選択します。

IAM には、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストがあります。

- b. アクセス許可の境界として使用するポリシーを選択します。
8. [Next] を選択します。
  9. [ロール名] では、オプションはサービスによって異なります。
    - サービスでロール名が定義されている場合、ロール名を編集することはできません。
    - サービスでロール名のプレフィックスが定義されている場合、オプションのサフィックスを入力できます。
    - サービスでロール名が定義されていない場合、ロールに名前を付けることができます。

**⚠ Important**

ロールに名前を付けるときは、次のことに注意してください。

- ロール名は AWS アカウント内で一意である必要があります。ただし、大文字と小文字は区別されません。

例えば、**PRODROLE** と **prodrole** の両方の名前で作成することはできません。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。ただし、サインインプロセスなど、コンソールにロール名がユーザーに表示される場合、ロール名は大文字と小文字が区別されません。

- 他のエンティティがロールを参照する可能性があるため、ロールを作成した後にロール名を編集することはできません。

10. (オプション) [説明] にロールの説明を入力します。
11. (オプション) ロールのユースケースとアクセス許可を編集するには、[ステップ 1: 信頼されたエンティティを選択] または [ステップ 2: アクセス権限を追加] のセクションで [編集] を選択します。
12. (オプション) ロールの識別、整理、検索を簡単にするには、キーと値のペアとしてタグを追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「[IAM リソースにタグを付ける](#)」を参照してください。

13. ロールを確認したら、[Create role] (ロールを作成) を選択します。

#### 例 1: データベース管理者としてユーザーを設定する (コンソール)

この例では、IAM ユーザーである Alice を データベース管理者 として設定するために必要な手順を示しています。そのセクションのテーブルの最初の列情報を使用し、ユーザーが Amazon RDS モニタリングを有効にできるようにします。Alice の IAM ユーザーに DatabaseAdministrator ポリシーをアタッチして、IAM ユーザーが Amazon データベースサービスを管理できるようにします。このポリシーによって、Alice は `rds-monitoring-role` という名前のロールを Amazon RDS サービスに渡すことができ、サービスが Alice の代わりに Amazon RDS データベースをモニタリングすることができます。

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. [ポリシー] を選択し、検索ボックスに **database** と入力してから [Enter] を押します。
3. [DatabaseAdministrator] ポリシーのラジオボタンを選択し、[アクション]、[アタッチ] の順に選択します。
4. エンティティのリストから [Alice] を選択してから、[ポリシーをアタッチ] を選択します。これで Alice は、AWS データベースを管理できます。ただし、Alice がこれらのデータベースをモニタリングできるようにするには、サービスロールを設定する必要があります。
5. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
6. [AWS サービス] ロールタイプを選択してから、[Amazon RDS] を選択します。
7. [拡張モニタリングの Amazon RDS ロール] ユースケースを選択します。
8. Amazon RDS により、ロールのアクセス許可が定義されます。[Next: Review (次へ: 確認)] を選択して続行します。
9. ロール名は、Alice が現在持っている DatabaseAdministrator ポリシーによって指定されたものである必要があります。たとえば **rds-monitoring-role** です。[Role name] (ロール名) にそれを入力します。
10. (オプション) [Role description] (ロールの説明) に、新しいロールの説明を入力します。
11. 詳細を確認したら、[ロールの作成] を選択します。
12. これで Alice は、Amazon RDS コンソールの [Monitoring] (モニタリング) セクションで [RDS Enhanced Monitoring] (RDS 拡張モニタリング) を有効にできるようになりました。たとえば、DB インスタンスを作成する場合、リードレプリカを作成する場合、または、DB インスタンスを修正する場合に有効にします。Alice は、[Enable Enhanced Monitoring] (拡張モニタリン

グを有効にする) を [Yes] (はい) に設定するとき、作成したロール名 (rds-monitoring-role) を [Monitoring Role] (モニタリングロール) ボックスに入力する必要があります。

## 例 2: ネットワーク管理者としてユーザーを設定する (コンソール)


この例では、IAM ユーザーである Jorge を [ネットワーク管理者](#) として設定するために必要な手順を示しています。このセクションの表の情報を使用して、Jorge が VPC との間で送受信される IP トラフィックをモニタリングできるようにします。また、Jorge がその情報を CloudWatch Logs のログに取り込むことができるようにします。 [NetworkAdministrator](#) ポリシーを Jorge の IAM ユーザーにアタッチして、IAM ユーザーが AWS ネットワークリソースを設定できるようにします。このポリシーでは、フローログを作成する際に、Jorge が flow-logs\* で始まる名前のロールを Amazon EC2 に渡すことも可能です。このシナリオでは、例 1 と違って事前定義されたサービスロールの種類がないため、いくつかのステップが異なります。

1. AWS Management Console にサインインして、IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
2. ナビゲーションペインで、[ポリシー] を選択し、検索ボックスに **network** と入力してから [Enter] を押します。
3. NetworkAdministrator ポリシーの横にあるチェックボックスをオンにし、[アクション]、[アタッチ] の順に選択します。
4. ユーザーのリストで、Jorge の横にあるチェックボックスを選択してから、[Attach policy] (ポリシーのアタッチ) を選択します。これで Jorge は、AWS ネットワークリソースを管理できます。ただし、VPC の IP トラフィックのモニタリングを有効にするには、サービスロールを設定する必要があります。
5. 作成する必要があるサービスロールに事前定義された管理ポリシーがないため、先にそれを作成する必要があります。ナビゲーションペインで、[ポリシー]、[ポリシーの作成] の順に選択します。
6. [ポリシーエディタ] セクションで、[JSON] オプションを選択し、以下の JSON ポリシードキュメントからテキストをコピーします。このテキストを [JSON] ボックスに貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

7. [ポリシーの検証](#)中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Next] (次へ) を選択します。

 Note

いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「[ポリシーの再構成](#)」を参照してください。

8. [確認および作成] ページで、ポリシー名として「**vpc-flow-logs-policy-for-service-role**」と入力します。[このポリシーで定義されているアクセス許可]を確認し、ポリシーによって付与されたアクセス許可を確認し、[ポリシーを作成] を選択して作業を保存します。  
  
新しいポリシーが管理ポリシーの一覧に表示され、アタッチの準備ができます。
9. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
10. [AWS サービス] ロールタイプを選択し、続いて [Amazon EC2] を選択します。
11. [Amazon EC2] ユースケースを選択します
12. [アクセス許可ポリシーをアタッチする] ページで、先ほど作成したポリシー [vpc-flow-logs-policy-for-service-role] を選択してから、[Next: Review (次へ: 確認)] を選択します。
13. ロール名は Jorge が現在持っている、NetworkAdministrator ポリシーによって許可されている必要があります。flow-logs- で始まる名前であれば、使用可能です。この例では、[Role name] (ロール名) に **flow-logs-for-jorge** と入力します。
14. (オプション) [Role description] (ロールの説明) に、新しいロールの説明を入力します。
15. 詳細を確認したら、[ロールの作成] を選択します。

- これで、このシナリオに必要な信頼ポリシーを設定することができます。[Roles] (ロール) ページで、flow-logs-for-jorge ロール (チェックボックスではなく名前) を選択します。新しいロールの詳細ページで、[信頼関係] タブを選択してから、[信頼関係の編集] を選択します。
- 「Service」 行の ec2.amazonaws.com のエントリを置き換えて以下のように変更します。

```
"Service": "vpc-flow-logs.amazonaws.com"
```

- Jorge はこれで、Amazon EC2 コンソールで VPC またはサブネットのフローログを作成できます。フローログを作成するときに、flow-logs-for-jorge ロールを指定します。このロールには、ログを作成し、そのログにデータを書き込む権限があります。

## AWS グローバル条件コンテキストキー

[プリンシパル](#)が AWS に[リクエスト](#)を行うと、AWS はリクエスト情報を[リクエストコンテキスト](#)に収集します。JSON ポリシーの Condition 要素を使用して、リクエストコンテキストのキーを、ポリシーで指定したキー値と比較できます。リクエスト情報は、リクエストを行うプリンシパル、リクエストの対象となるリソース、リクエスト自体に関するメタデータなど、さまざまなソースから提供されます。

グローバル条件キーは、すべての AWS サービスで使用できます。これらの条件キーはすべてのポリシーで使用できますが、キーはすべてのリクエストコンテキストで使用できるわけではありません。例えば、aws:SourceAccount 条件キーは、リソースへの呼び出しが[AWS サービスプリンシパル](#)によって直接行われた場合にのみ使用できます。グローバルキーがリクエストコンテキストに含まれる状況の詳細については、各キーの可用性情報を参照してください。

個別のサービスの中には、他のサービスのリクエストコンテキストで使用できる独自の条件キーを作成するものもあります。クロスサービス条件キーは、ec2: や lambda: など、サービスの名前と一致するプレフィックスを含むグローバル条件キーの一種ですが、他のサービスで使用できます。

サービス固有の条件キーは、個々の AWS サービスで使用するよう定義されています。例えば、Amazon S3 では、s3:VersionId 条件キーを使用してポリシーを作成し、Amazon S3 オブジェクトの特定のバージョンへのアクセスを制限できます。この条件キーはサービス固有です。つまり、Amazon S3 サービスへのリクエストでのみ機能します。サービス固有の条件キーについては、「[AWS サービスのアクション、リソース、および条件キー](#)」を参照し、キーを表示するサービスを選択してください。

**Note**

いくつかの状況でのみ利用可能な条件キーを使用する場合は、条件演算子の [IfExists](#) バージョンを使用できます。リクエストコンテキストに条件キーがない場合、ポリシーは評価に失敗する可能性があります。たとえば、...IfExists 演算子を使用する以下の条件ブロックでは、リクエストの送信元が特定の IP 範囲または VPC である場合、この条件に一致となります。いずれかまたは両方のキーがリクエストコンテキストに含まれていない場合でも、条件は引き続き true を返します。これらの値は、指定したキーがリクエストコンテキスト内に含まれる場合にのみ確認されます。他の演算子のキーが存在しない場合のポリシーの評価方法の詳細については、「[条件演算子](#)」を参照してください。

```
"Condition": {
  "IpAddressIfExists": {"aws:SourceIp" : ["xxx"] },
  "StringEqualsIfExists" : {"aws:SourceVpc" : ["yyy"]}
}
```

**Important**

条件を複数のキー値を持つ要求コンテキストと比較するには、ForAllValues または ForAnyValue 集合演算子を使用する必要があります。セット演算子は、複数值の条件キーでのみ使用してください。単一値の条件キーで集合演算子を使用しないでください。詳細については、「[複数值のコンテキストキー](#)」を参照してください。

プリンシパルのプロパティ	ロールセッションのプロパティ	ネットワークのプロパティ	リソースのプロパティ	リクエストのプロパティ
<a href="#">aws:PrincipalArn</a>	<a href="#">aws:FederatedProvider</a>	<a href="#">aws:SourceIp</a>	<a href="#">aws:ResourceAccount</a>	<a href="#">aws:CalledVia</a>
<a href="#">aws:PrincipalAccount</a>	<a href="#">aws:TokenIssueTime</a>	<a href="#">aws:SourceVpc</a>	<a href="#">aws:ResourceOrgPaths</a>	<a href="#">aws:CalledViaFirst</a>
<a href="#">aws:PrincipalOrgPaths</a>	<a href="#">aws:MultiFactorAuthAge</a>	<a href="#">aws:VpcSourceIp</a>	<a href="#">aws:ResourceOrgID</a>	<a href="#">aws:CalledViaLast</a>
<a href="#">aws:PrincipalOrgID</a>				<a href="#">aws:ViaAWSService</a>

プリンシパルのプロパティ	ロールセッションのプロパティ	ネットワークのプロパティ	リソースのプロパティ	リクエストのプロパティ
<a href="#">aws:PrincipalTag/tag-key</a>	<a href="#">aws:MultiFactorAuthPresent</a>		<a href="#">aws:ResourceTag/tag-key</a>	<a href="#">aws:CurrentTime</a>
<a href="#">aws:PrincipalIsAWSService</a>	<a href="#">aws:Ec2InstanceSourceVpc</a>			<a href="#">aws:EpochTime</a>
<a href="#">aws:PrincipalServiceName</a>	<a href="#">aws:Ec2InstanceSourcePrivateIpv4</a>			<a href="#">aws:referrer</a>
<a href="#">aws:PrincipalServiceNamesList</a>	<a href="#">aws:SourceIdentity</a>			<a href="#">aws:RequestedRegion</a>
<a href="#">aws:PrincipalType</a>	<a href="#">ec2:RoleDelivery</a>			<a href="#">aws:RequestedTag/tag-key</a>
<a href="#">aws:user-id</a>	<a href="#">ec2:SourceInstanceArn</a>			<a href="#">aws:TagKeys</a>
<a href="#">aws:username</a>	<a href="#">glue:RoleAssumedBy</a>			<a href="#">aws:SecureTransport</a>
	<a href="#">glue:CredentialIssuingService</a>			<a href="#">aws:SourceArn</a>
	<a href="#">lambda:SourceFunctionArn</a>			<a href="#">aws:SourceAccount</a>
	<a href="#">ssm:SourceInstanceArn</a>			<a href="#">aws:SourceOrgPaths</a>
	<a href="#">identitystore:UserId</a>			<a href="#">aws:SourceOrgID</a>
				<a href="#">aws:UserAgent</a>



## プリンシパルのプロパティ

次の条件キーを使用して、リクエストを行うプリンシパルの詳細を、ポリシーで指定したプリンシパルのプロパティと比較します。リクエストを実行できるプリンシパルのリストについては、「[プリンシパルの指定](#)」を参照してください。

### 目次

- [aws:PrincipalArn](#)
- [aws:PrincipalAccount](#)
- [aws:PrincipalOrgPaths](#)
- [aws:PrincipalOrgID](#)
- [aws:PrincipalTag/tag-key](#)
- [aws:PrincipalsAWSService](#)
- [aws:PrincipalServiceName](#)
- [aws:PrincipalServiceNamesList](#)
- [aws:PrincipalType](#)
- [aws:userid](#)
- [aws:username](#)

### aws:PrincipalArn

このキーを使用して、リクエストを行ったプリンシパルの [Amazon リソースネーム \(ARN\)](#) をポリシーで指定した ARN と比較します。IAM ロールの場合、リクエストコンテキストは、ロールを引き受けたユーザーの ARN ではなく、ロールの ARN を返します。

- 可用性 – このキーは、すべての署名付きリクエストのリクエストコンテキストに含まれます。匿名リクエストには、このキーは含まれません。この条件キーで指定できるプリンシパルのタイプは次のとおりです。
  - IAM ロール
  - IAM ユーザー
  - AWS STS フェデレーションユーザーセッション
  - AWS アカウントのルートユーザー
- データ型 - ARN、文字列



AWS では、ARN を比較する場合、[文字列演算子](#)の代わりに [ARN 演算子](#)を使用することをお勧めします。

- 値タイプ — 単一値
- 値の例 次のリストは、aws:PrincipalArn 条件キーで指定できるさまざまなタイプのプリンシパルに対して返されるリクエストコンテキスト値を示しています。
- IAM ロール — リクエストコンテキストには、条件キー aws:PrincipalArn に次の値が含まれています。引き受けたロールのセッション ARN をこの条件キーの値として指定しないでください。引き受けたロールセッションのプリンシパルの詳細については、「[ロールセッションプリンシパル](#)」を参照してください。

```
arn:aws:iam::123456789012:role/role-name
```

- IAM ユーザー — リクエストコンテキストには、条件キー aws:PrincipalArn に次の値が含まれています。

```
arn:aws:iam::123456789012:user/user-name
```

- AWS STS フェデレーションユーザーセッション — リクエストコンテキストには、条件キー aws:PrincipalArn に次の値が含まれています。

```
arn:aws:sts::123456789012:federated-user/user-name
```

- AWS アカウント ルートユーザー — リクエストコンテキストには、条件キー aws:PrincipalArn に次の値が含まれています。ルートユーザー ARN を aws:PrincipalArn 条件キーの値として指定すると、アクセス許可が AWS アカウント のルートユーザーのみに制限されます。これは、リソースベースポリシーのプリンシパル要素でルートユーザー ARN を指定して、AWS アカウント に権限を委任することとは異なります。リソースベースのポリシーのプリンシパル要素でルートユーザー ARN を指定する方法については、「[AWS アカウント プリンシパル](#)」を参照してください。

```
arn:aws:iam::123456789012:root
```

AWS Organizations サービスコントロールポリシー (SCP) の条件キー aws:PrincipalArn の値としてルートユーザー ARN を指定することができます。SCP は、組織のアクセス許可の管理に使用される組織ポリシーの一種で、組織内のメンバーアカウントにのみ影響します。SCP は、メンバーアカウントのルートユーザーを含む、メンバーアカウントの IAM ユーザーとロールのアクセス許可

を制限します。SCP がアクセス許可に与える影響については、「Organizations ユーザーガイド」の「[アクセス許可における SCP 効果](#)」を参照してください。

## aws:PrincipalAccount

このキーを使用して、リクエスト元のプリンシパルが属するアカウントと、ポリシーで指定したアカウント識別子を比較します。匿名リクエストの場合、リクエストコンテキストは `anonymous` を返します。

- 可用性 – このキーは、匿名リクエストなどのすべてのリクエストのリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ – 単一値

次の例では、アカウント番号が 123456789012 のプリンシパル以外に対するアクセスは拒否されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:accountID:resource"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

## aws:PrincipalOrgPaths

このキーを使用して、リクエストを行っているプリンシパルの AWS Organizations パスをポリシー内のパスと比較します。そのプリンシパルは、IAM ユーザー、IAM ロール、フェデレーションユーザー、または AWS アカウントのルートユーザーです。ポリシーでは、この条件キーによって、リクエストが AWS Organizations で指定された組織ルートまたは組織単位 (OU) 内のアカウントメンバーであることが保証されます。AWS Organizations パスは、Organizations エンティティの構造をテキストで表記したものです。パスの使用と理解の詳細については、「[AWS Organizations エンティティパスを理解する](#)」を参照してください。

- 可用性 – このキーは、プリンシパルが組織のメンバーである場合にのみリクエストコンテキストに含まれます。匿名リクエストには、このキーは含まれません。
- データ型 – [文字列](#) (リスト)
- 値タイプ – 複数値

### Note

組織 ID はグローバルに一意ですが、OU ID とルート ID は組織内でのみ一意です。これは、2 つの組織が同じ組織 ID を共有しないことを意味します。ただし、別の組織が自分と同じ ID を持つ OU またはルートを持っている可能性があります。OU またはルートを指定するときは、必ず組織 ID を含めることをお勧めします。

たとえば、次の条件は、true OU に直接アタッチされているが、子の OU にはアタッチされていないアカウントのプリンシパルに対して ou-ab12-22222222 を返します。

```
"Condition" : { "ForAnyValue:StringEquals" : {  
  "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]  
}}
```

次の条件は、OU またはその子の OU に直接アタッチされているアカウントのプリンシパルに対して true を返します。ワイルドカードを含める場合は、StringLike 条件演算子を使用する必要があります。

```
"Condition" : { "ForAnyValue:StringLike" : {  
  "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/  
*"]  
}}
```

次の条件は、OU またはその子の OU に直接アタッチされているアカウントのプリンシパルに対して true を返します。以前の条件は、OU または任意の子を対象としています。以下の条件は、子のみ (およびそれらの子のすべての子) を対象としています。

```
"Condition" : { "ForAnyValue:StringLike" : {  
  "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/  
ou-*"]  
}}
```

次の条件は、親 OU に関係なく、o-a1b2c3d4e5 組織内のすべてのプリンシパルへのアクセスを許可します。

```
"Condition" : { "ForAnyValue:StringLike" : {  
  "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/*"]  
}}
```

aws:PrincipalOrgPaths は複数の値を持つ条件キーです。複数值のキーは、リクエストコンテキストに複数の値を持つことができます。ForAnyValue 条件演算子で複数の値を使用する場合、プリンシパルのパスはポリシーに一覧表示されているパスの 1 つと一致する必要があります。複数值を持つ条件キーの詳細については、「[複数值のコンテキストキー](#)」を参照してください。

```
"Condition": {  
  "ForAnyValue:StringLike": {  
    "aws:PrincipalOrgPaths": [  
      "o-a1b2c3d4e5/r-ab12/ou-ab12-33333333/*",  
      "o-a1b2c3d4e5/r-ab12/ou-ab12-22222222/*"  
    ]  
  }  
}
```

## aws:PrincipalOrgID

このキーを使用して、リクエスト元のプリンシパルが属する AWS Organizations の組織の識別子と、ポリシーで指定された識別子を比較します。

- 可用性 – このキーは、プリンシパルが組織のメンバーである場合にのみリクエストコンテキストに含まれます。匿名リクエストには、このキーは含まれません。
- データ型 - [文字列](#)
- 値タイプ — 単一値

このグローバルキーは、組織内のすべての AWS アカウントのすべてのアカウント ID を一覧表示する代わりに使用できます。この条件キーを使用して、[リソースベースのポリシー](#)での Principal 要素の指定を簡素化します。[組織 ID](#) は、条件要素で指定できます。アカウントを追加したり削除したりすると、aws:PrincipalOrgID を含むポリシーには正しいアカウントが自動的に組み込まれ、手動で更新する必要はありません。

たとえば、以下の o-xxxxxxxxxxxxx バケットポリシーでは、policy-ninja-dev 組織の任意のアカウントのメンバーに Amazon S3 バケットへのオブジェクトの追加を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AllowPutObject",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::policy-ninja-dev/*",
    "Condition": {"StringEquals":
      {"aws:PrincipalOrgID": "o-xxxxxxxxxxxxx"}
    }
  }
}
```

#### Note

また、このグローバル条件は、AWS 組織の管理アカウントにも適用されます。このポリシーは、指定した組織以外のすべてのプリンシパルが、Amazon S3 バケットにアクセスできないようにします。これには、AWS CloudTrail がログデータを Amazon S3 バケットに送信する場合など、お客様の内部リソースと相互にやり取りを行う AWS サービスも含まれます。AWS サービスでアクセス許可を安全に付与する方法については、「[aws:PrincipalsAWSService](#)」を参照してください。

AWS Organizations の詳細については、AWS Organizationsユーザーガイドの「[AWS Organizations とは](#)」を参照してください。

## aws:PrincipalTag/tag-key

このキーを使用して、リクエストを行うプリンシパルにアタッチされたタグと、ポリシーで指定したタグを比較します。プリンシパルに複数のタグがアタッチされている場合、リクエストコンテキストには、アタッチされたタグキーごとに 1 つの `aws:PrincipalTag` キーが含まれます。

- 可用性 – このキーは、プリンシパルが、タグがアタッチされた IAM ユーザーである場合にリクエストコンテキストに含まれます。これは、タグまたは [セッションタグ](#) がアタッチされた IAM ロールを使用するプリンシパルのために含まれます。匿名リクエストには、このキーは含まれません。
- データ型 - [文字列](#)
- 値タイプ — 単一値

カスタム属性は、キーバリューのペアの形式でユーザーまたはロールに追加できます。IAM タグの使用の詳細については、「[IAM リソースのタグ付け](#)」を参照してください。aws:PrincipalTag を使用して AWS プリンシパルの [アクセスをコントロール](#) できます。

この例では、`department=hr` タグを持つユーザーが IAM ユーザー、グループ、またはロールを管理できるようにする ID ベースポリシーを作成する方法を示しています。このポリシーを使用するには、サンプルポリシーの `#####` を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": "hr"
        }
      }
    }
  ]
}
```

## aws:PrincipalsAWSService

このキーを使用して、リソースへの呼び出しが AWS [サービスプリンシパル](#)によって直接行われているかどうかを確認します。たとえば、AWS CloudTrail はサービスプリンシパル `cloudtrail.amazonaws.com` を使用して、Amazon S3 バケットにログを書き込みます。サービスがサービスプリンシパルを使用してリソースに対して直接アクションを実行する場合、リクエストコンテキストキーは `true` に設定されます。サービスが IAM プリンシパルの認証情報を使用し、プリンシパルに代わってリクエストを行う場合、コンテキストキーは `false` に設定されます。また、サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合も、`false` に設定されます。

- 可用性 — このキーは、AWS 認証情報を使用するすべての署名済み API リクエストのリクエストコンテキストに存在します。匿名リクエストには、このキーは含まれません。
- データ型 – [ブール値](#)
- 値タイプ — 単一値

この条件キーを使用して、AWS サービスへのアクセスを安全に許可しながら、信頼できる ID と予想されるネットワークの場所へのアクセスを制限できます。

次の Amazon S3 バケットポリシーの例では、リクエストが `vpc-111bbb22` から発信されているか、CloudTrail などのサービスプリンシパルから発信されていない限り、バケットへのアクセスが制限されています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Expected-network+service-principal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/AWSLogs/AccountNumber/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22"
        },
        "BoolIfExists": {
          "aws:PrincipalIsAWSService": "false"
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

次のビデオでは、ポリシーで `aws:PrincipalIsAWSService` 条件キーを使用する方法について詳しく説明します。

[許可されたユーザー、予想されるネットワークの場所、および AWS のサービスへのアクセスを一括に安全に許可します。](#)

### `aws:PrincipalServiceName`

このキーを使用して、[サービスプリンシパル](#)の名前を、リソースにリクエストするサービスプリンシパルとともに設定します。このキーを使用して、この呼び出しが特定のサービスプリンシパルによって行われたかどうかを確認できます。サービスプリンシパルがリソースに直接リクエストする場合、`aws:PrincipalServiceName` キーには、サービスプリンシパルの名前が含まれます。たとえば、AWS CloudTrail のサービスプリンシパル名は `cloudtrail.amazonaws.com` です。

- 可用性 — このキーは、呼び出しが AWS サービスプリンシパル。このキーは、次のような他の状況では存在しません。
  - サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合。
  - サービスが IAM プリンシパルの認証情報を使用し、プリンシパルに代わってリクエストを行う場合。
  - 呼び出しが IAM プリンシパルによって直接行われた場合。
  - 匿名の依頼者によって電話された場合。
- データ型 - [文字列](#)
- 値タイプ — 単一値

この条件キーを使用して、AWS サービスへのアクセスを安全に許可しながら、信頼できる ID と予想されるネットワークの場所へのアクセスを制限できます。

次の Amazon S3 バケットポリシーの例では、リクエストが `vpc-111bbb22` から発信されているか、CloudTrail などのサービスプリンシパルから発信されていない限り、バケットへのアクセスが制限されています。

```
{
```



```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "expected-network+service-principal",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/AWSLogs/AccountNumber/*",
    "Condition": {
      "StringNotEqualsIfExists": {
        "aws:SourceVpc": "vpc-111bbb22",
        "aws:PrincipalServiceName": "cloudtrail.amazonaws.com"
      }
    }
  }
]
```

#### aws:PrincipalServiceNamesList

このキーは、サービスに属するすべての[サービスプリンシパル](#)名のリストを提供します。これは高度な条件キーです。これを使用して、サービスが特定のリージョンからのみリソースにアクセスすることを制限できます。一部のサービスでは、特定のリージョン内のサービスの特定のインスタンスを示すために、リージョナルサービスプリンシパルを作成できます。リソースへのアクセスをサービスの特定のインスタンスに制限できます。サービスプリンシパルがリソースに直接リクエストを行うと、aws:PrincipalServiceNamesList には、サービスのリージョナルインスタンスに関連付けられているすべてのサービスプリンシパル名の順序付けられていないリストが含まれます。

- 可用性 — このキーは、呼び出しがAWSサービスプリンシパル。このキーは、次のような他の状況では存在しません。
  - サービスが[サービスロール](#)または[サービスリンクロール](#)を使用してプリンシパルに代わって呼び出しを行う場合。
  - サービスが IAM プリンシパルの認証情報を使用し、プリンシパルに代わってリクエストを行う場合。
  - 呼び出しが IAM プリンシパルによって直接行われた場合。
  - 匿名の依頼者によって電話された場合。
- データ型 – [文字列](#) (リスト)
- 値タイプ — 複数値

`aws:PrincipalServiceNamesList` は複数の値を持つ条件キーです。複数値のキーは、リクエストコンテキストに複数の値を持つことができます。このキーを使用する場合は、[文字列条件演算子](#)とともに `ForAnyValue` または `ForAllValues` の集合演算子を使用する必要があります。複数値を持つ条件キーの詳細については、「[複数値のコンテキストキー](#)」を参照してください。

#### `aws:PrincipalType`

このキーを使用して、リクエストを行うプリンシパルと、ポリシーで指定したプリンシパルのタイプを比較します。詳細については、「[プリンシパルの指定](#)」を参照してください。principal キーの値の具体的な例については、「[プリンシパルキーの値](#)」を参照してください。

- 可用性 – このキーは、匿名リクエストなどのすべてのリクエストのリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

#### `aws:userid`

このキーを使用して、リクエストのプリンシパル ID とポリシーで指定した識別子を比較します。IAM ユーザーの場合、リクエストコンテキスト値はユーザー ID です。IAM ロールの場合、この値の形式はさまざまです。さまざまなプリンシパルで情報がどのように表示されるかについては、「[プリンシパルの指定](#)」を参照してください。principal キーの値の具体的な例については、「[プリンシパルキーの値](#)」を参照してください。

- 可用性 – このキーは、匿名リクエストなどのすべてのリクエストのリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

#### `aws:username`

このキーを使用して、リクエストのユーザー名をポリシーで指定したユーザー名と比較します。さまざまなプリンシパルで情報がどのように表示されるかについては、「[プリンシパルの指定](#)」を参照してください。principal キーの値の具体的な例については、「[プリンシパルキーの値](#)」を参照してください。

- 可用性 – このキーは常に IAM ユーザーのリクエストコンテキストに含まれます。匿名のリクエストと AWS アカウントのルートユーザー または IAM ロールを使用して行われたリクエストには、

このキーは含まれません。IAM Identity Center の認証情報を使用して行われたリクエストでは、このキーはコンテキストに含まれません。

- データ型 - [文字列](#)
- 値タイプ — 単一値

## ロールセッションのプロパティ

次の条件キーを使用して、セッション生成時のロールセッションのプロパティを比較します。これらの条件キーは、ロールセッションまたはフェデレーションユーザー認証情報を持つプリンシパルによってリクエストが行われた場合にのみ使用できます。これらの条件キーの値は、ロールのセッショントークンに埋め込まれます。

[ロール](#)はプリンシパルの一種です。[プリンシパルのプロパティ](#) セクションの条件キーを使用して、ロールのリクエスト実行中にロールのプロパティを評価することもできます。

### 目次

- [aws:FederatedProvider](#)
- [aws:TokenIssueTime](#)
- [aws:MultiFactorAuthAge](#)
- [aws:MultiFactorAuthPresent](#)
- [aws:Ec2InstanceSourceVpc](#)
- [aws:Ec2InstanceSourcePrivateIpv4](#)
- [aws:SourceIdentity](#)
- [ec2:RoleDelivery](#)
- [ec2:SourceInstanceArn](#)
- [glue:RoleAssumedBy](#)
- [glue:CredentialIssuingService](#)
- [lambda:SourceFunctionArn](#)
- [ssm:SourceInstanceArn](#)
- [identitystore:UserId](#)

## aws:FederatedProvider

このキーを使用して、プリンシパルが発行した ID プロバイダー (IdP) とポリシーで指定した識別子とを比較します。つまり IAM ロールは、AssumeRoleWithWebIdentity AWS STS オペレーションを使用して引き受けられたということです。結果のロールセッションの一時認証情報を使用してリクエストを作成する場合、リクエストコンテキストは、元のフェデレーテッド ID を認証した IdP を識別します。

- 可用性 — このキーは、プリンシパルがロールセッションプリンシパルであり、AssumeRoleWithWebIdentity を使用してロールが引き受けられたときにそのセッションが発行された場合に存在します。
- データ型 - [文字列](#)
- 値タイプ — 単一値

例えば、ユーザーが Amazon Cognito を使用して認証された場合、リクエストコンテキストには cognito-identity.amazonaws.com という値が含まれます。同様に、ユーザーが Login with Amazon を使用して認証された場合、リクエストコンテキストには www.amazon.com という値が含まれます。

単一値の条件キーは、[変数](#)として使用できます。次のリソースベースのポリシーの例では、リソースの ARN において aws:FederatedProvider キーをポリシー変数として使用しています。このポリシーでは、IdP を使用して認証されたプリンシパルが、発行元の ID プロバイダーに特有のパスを使用して、Amazon S3 からオブジェクトを取得できます。

## aws:TokenIssueTime

このキーを使用して、一時的なセキュリティ認証情報が発行された日時と、ポリシーで指定した日時を比較します。

- 可用性 – このキーは、プリンシパルが一時的な認証情報を使用してリクエストを行う場合にのみ、リクエストコンテキストに含まれます。このキーは、アクセスキーを使用して行われた AWS CLI、AWS API、または AWS SDK リクエストには存在しません。
- データ型 – [日付](#)
- 値タイプ — 単一値

一時的認証情報の使用がサポートされているサービスについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

## aws:MultiFactorAuthAge

このキーを使用して、リクエスト元のプリンシパルが MFA を使用して承認されてからの秒数と、ポリシーで指定した数値を比較します。MFA の詳細については、「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

### Important

この条件キーは、フェデレーテッド ID や、AWS CLI、AWS API、または AWS SDK リクエストに署名するためにアクセスキーを使用して行われたリクエストには存在しません。一時的なセキュリティ認証情報を使用して API 操作に MFA 保護を追加する方法の詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

IAM フェデレーテッド ID の検証に MFA が使用されているかどうかを確認するには、認証方法を ID プロバイダーから AWS にセッションタグとして渡すことができます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。IAM アイデンティティセンター ID に MFA を適用するには、[アクセスコントロールの属性を有効にし](#)、認証方法を使用して SAML アサーションクレームを ID プロバイダーから IAM アイデンティティセンターに渡すことができます。

- 可用性 – このキーは、プリンシパルが [一時的なセキュリティ認証情報](#) を使用してリクエストを行う場合にのみ、リクエストコンテキストに含まれます。MFA 条件を指定したポリシーは、次にアタッチすることができます。
  - IAM ユーザーまたはグループ
  - Amazon S3 バケット、Amazon SQS キュー、または Amazon SNS トピックなどのリソース
  - ユーザーが引き受けることのできる IAM ロールの信頼ポリシー
- データ型 – [数値](#)
- 値タイプ – 単一値

## aws:MultiFactorAuthPresent

このキーを使用して、リクエストを行った [一時的なセキュリティ認証情報](#) を検証するために多要素認証 (MFA) を使用したか確認します。

**⚠ Important**

この条件キーは、フェデレーテッド ID や、AWS CLI、AWS API、または AWS SDK リクエストに署名するためにアクセスキーを使用して行われたリクエストには存在しません。一時的なセキュリティ認証情報を使用して API 操作に MFA 保護を追加する方法の詳細については、「[MFA 保護 API アクセスの設定](#)」を参照してください。

IAM フェデレーテッド ID の検証に MFA が使用されているかどうかを確認するには、認証方法を ID プロバイダーから AWS にセッションタグとして渡すことができます。詳細については、「[AWS STS でのセッションタグの受け渡し](#)」を参照してください。IAM アイデンティティセンター ID に MFA を適用するには、[アクセスコントロールの属性を有効にし、認証方法を使用して SAML アサーションクレームを ID プロバイダーから IAM アイデンティティセンターに渡すことができます。](#)

- 可用性 – このキーは、プリンシパルが一時的な認証情報を使用してリクエストを行う場合にのみ、リクエストコンテキストに含まれます。MFA 条件を指定したポリシーは、次にアタッチすることができます。
  - IAM ユーザーまたはグループ
  - Amazon S3 バケット、Amazon SQS キュー、または Amazon SNS トピックなどのリソース
  - ユーザーが引き受けることのできる IAM ロールの信頼ポリシー
- データ型 – [ブール値](#)
- 値タイプ – 単一値

一時的な認証情報は、[AssumeRole](#) または [GetSessionToken](#) の一時トークンを持つ IAM ロールと IAM ユーザー、および AWS Management Console のユーザーを認証するために使用されます。

IAM ユーザーアクセスキーは、長期的な認証情報ですが、場合によっては AWS は、IAM ユーザーの代わりに一時的な認証情報を作成し、操作を実行します。このような場合、`aws:MultiFactorAuthPresent` キーはリクエストに存在し、`false` の値に設定されます。これが起こる一般的なケースは、次の 2 つです。

- AWS Management Console の IAM ユーザーが、知らないうちに一時的認証情報を使用します。ユーザーは、長期的な認証情報であるユーザー名とパスワードを使用してコンソールにサインインします。ただし、バックグラウンドでは、コンソールがユーザーに代わって一時的な認証情報を作成します。

- IAM ユーザーがサービスを呼び出すと、AWS のサービスはユーザーの認証情報を再利用して、別のサービスに対して別のリクエストを行います。たとえば、Athena を呼び出して Amazon S3 バケットにアクセスする場合、または AWS CloudFormation Amazon EC2 インスタンスを作成するには、次の手順に従います。後続のリクエストで、AWS は一時的認証情報を使用します。

一時的認証情報の使用がサポートされているサービスについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

`aws:MultiFactorAuthPresent` キーは、API または CLI コマンドがアクセスキーペアなど長期的な認証情報で呼び出された場合には表示されません。したがって、このキーを確認する場合は [...IfExists](#) バージョンの条件演算子の使用をお勧めします。

以下の Condition 要素は、MFA を使用してリクエストが認証されるかどうかを確認する信頼性の高い方法ではない点に注意してください。

```
##### WARNING: NOT RECOMMENDED #####
"Effect" : "Deny",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny 効果、Bool 要素、false 値のこの組み合わせは、MFA を使用して認証できるが認証されなかったリクエストを拒否します。このステートメントは、MFA の使用をサポートする一時的認証情報にのみ適用されます。このステートメントは、長期的認証情報を使用して行われるリクエスト、または MFA を使用して認証されるリクエストへのアクセスを拒否しません。ロジックが複雑で MFA 認証が実際に使用されたかどうかをテストしないため、この例は慎重に使用してください。

また、Deny 効果、Null 要素、true の組み合わせは使用しないでください。この組み合わせも同様に、ロジックはさらに複雑になるためです。

推奨される組み合わせ

[BoolIfExists](#) 演算子を使用して、リクエストが MFA を使用して認証されたかどうかを確認することをお勧めします。

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny、BoolIfExists、false のこの組み合わせは、MFA を使用して認証されないリクエストを拒否します。具体的には、MFA を使用しないで一時的認証情報を使用して行われたリクエストを拒否します。また、AWS CLI などの長期的な認証情報またはアクセスキーを使用して行われ



る AWS API オペレーションを使用して行われるリクエストも拒否されます。\*IfExists 演算子は、aws:MultiFactorAuthPresent キーが存在するかどうか、MFA が使用されているかどうかを確認します。これは、MFA を使用して認証されないリクエストを拒否する場合に使用します。これはより安全ですが、AWS CLI または AWS API にアクセスするためにアクセスキーを使用するコードやスクリプトを破損する可能性があります。

### 代替の組み合わせ

また、[BoolIfExists](#) 演算子を使用して、長期的認証情報を使用して行われる MFA 認証リクエストおよび AWS CLI または AWS API リクエストを許可することもできます。

```
"Effect" : "Allow",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "true" } }
```

キーが存在して MFA が使用されている場合またはキーが存在しない場合、この条件に一致となります。Allow、BoolIfExists、true のこの組み合わせは、MFA を使用して認証されたリクエスト、または MFA を使用して認証できないリクエストを許可します。つまり、リクエストが長期的なアクセスキーを使用する場合、AWS CLI、AWS API、および AWS SDK オペレーションが許可されます。この組み合わせでは、MFA を含むことはできるが含んでいない、一時的な認証情報からのリクエストが許可されません。

IAM コンソールのビジュアルエディタを使用してポリシーを作成し、[MFA required (MFA 必須)] を選択すると、この組み合わせが適用されます。この設定では、コンソールアクセスに MFA が必要ですが、MFA なしでプログラムによるアクセスを許可します。

または、Bool 演算子を使用して、MFA を使用して認証された場合にのみ、プログラムによるリクエストとコンソールによるリクエストを許可できます。

```
"Effect" : "Allow",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "true" } }
```

Allow、Bool、true のこの組み合わせは、MFA 認証リクエストのみを許可します。このステートメントは、MFA の使用をサポートする一時的認証情報にのみ適用されます。このステートメントは、長期的アクセスキーを使用して行われたリクエスト、または MFA を使用しないで一時的認証情報を使用して行われたリクエストへのアクセスを許可しません。

MFA キーが存在するかを確認する場合、以下のようなポリシーは使用しないでください。

```
##### WARNING: USE WITH CAUTION #####
```



```
"Effect" : "Allow",  
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Allow 効果、Null 要素、false 値のこの組み合わせは、リクエストが実際に認証されたかどうかにかかわらず、MFA を使用して認証できるリクエストのみを許可します。これにより、一時的認証情報を使用して行われるすべてのリクエストが許可され、長期的認証情報を使用して行われたリクエストは拒否されます。MFA 認証が実際に使用されたかどうかをテストしないため、この例は慎重に使用してください。

aws: Ec2InstanceSourceVpc

このキーは、Amazon EC2 IAM ロール認証情報が配信された VPC を識別します。[aws:SourceVPC](#) グローバルキーを含むポリシーでこのキーを使用すると、認証情報が配信された VPC (aws:SourceVPC) と一致する VPC から呼び出しが行われたかどうかを確認できます (aws:Ec2InstanceSourceVpc)。

- 可用性 — リクエスターが Amazon EC2 ロールクレデンシャルでリクエストに署名しているときはいつでも、このキーがリクエストコンテキストに含まれます。IAM ポリシー、サービスコントロールポリシー、VPC エンドポイントポリシー、リソースポリシーで使用できます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

このキーは VPC 識別子の値と共に使用できますが、aws:SourceVpc コンテキストキーと組み合わせた変数として使用すると最も便利です。この aws:SourceVpc コンテキストキーは、リクエスターが VPC エンドポイントを使用してリクエストを行う場合にのみリクエストコンテキストに含まれます。aws:Ec2InstanceSourceVpc と aws:SourceVpc を使用すると、通常は一緒に変化する値を比較できるため、aws:Ec2InstanceSourceVpc をより幅広く使用できます。

#### Note

この条件キーは、EC2-Classic では使用できません。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "RequireSameVPC",
```

```
"Effect": "Deny",
"Action": "*",
"Resource": "*",
"Condition": {
  "StringNotEquals": {
    "aws:SourceVpc": "${aws:Ec2InstanceSourceVpc}"
  },
  "Null": {
    "ec2:SourceInstanceARN": "false"
  },
  "BoolIfExists": {
    "aws:ViaAWSService": "false"
  }
}
}
```

上記の例では、aws:SourceVpc 値が aws:Ec2InstanceSourceVpc 値と等しくない場合、アクセスは拒否されます。ポリシーステートメントは、ec2:SourceInstanceARN 条件キーの有無をテストして Amazon EC2 インスタンスのロールとして使用されるロールのみに限定されます。

このポリシーでは、Amazon EC2 インスタンスロールに代わってリクエストが行われた場合に、aws:ViaAWSService が AWS リクエストを許可するようにしています。たとえば、Amazon EC2 インスタンスから暗号化された Amazon S3 バケットにリクエストを行うと、Amazon S3 がお客様に代わって AWS KMS を呼び出します。リクエストが AWS KMS に行われたときに、一部のキーが表示されません。

#### aws:Ec2InstanceSourcePrivateIPv4

このキーは、Amazon EC2 IAM ロール認証情報が配信されたプライマリ elastic network interface プライベート IPv4 アドレスを識別します。VPC ID とソースプライベート IP をグローバルに一意に組み合わせるには、この条件キーをコンパニオンキー aws:Ec2InstanceSourceVpc と一緒に使用する必要があります。このキー aws:Ec2InstanceSourceVpc を使用して、認証情報が配信されたのと同じプライベート IP アドレスからリクエストが行われたことを確認します。

- 可用性 — このキーは、リクエストが Amazon EC2 ロールクレデンシャルでリクエストに署名しているときはいつでも、リクエストコンテキストに含まれます。IAM ポリシー、サービスコントロールポリシー、VPC エンドポイントポリシー、リソースポリシーで使用できます。
- データ型 – [IP アドレス](#)
- 値タイプ — 単一値

**⚠ Important**

このキーは Allow ステートメント内で単独で使用しないでください。プライベートIPアドレスは、定義上、グローバルに一意ではありません。Amazon EC2 インスタンスの認証情報を使用できる VPC を指定するには、aws:Ec2InstanceSourcePrivateIPv4キーを使用するたびに aws:Ec2InstanceSourceVpc キーを使用する必要があります。

**ℹ Note**

この条件キーは、EC2-Classicでは使用できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:Ec2InstanceSourceVpc": "${aws:SourceVpc}"
        },
        "Null": {
          "ec2:SourceInstanceARN": "false"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:Ec2InstanceSourcePrivateIPv4": "${aws:VpcSourceIp}"
        },
        "Null": {
```

```
        "ec2:SourceInstanceARN": "false"
      },
      "BoolIfExists": {
        "aws:ViaAWSService": "false"
      }
    }
  ]
}
```

## aws:SourceIdentity

プリンシパルによって設定されたソース ID と、ポリシーで指定したソース ID を比較するには、このキーを使用します。

- 可用性 — このキーは、AWS STS `assume-role` CLI コマンドまたは AWS STS `AssumeRole` API オペレーションを使用してロールが引き継がれるときに、ソース ID が設定された後、リクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

ポリシーでこのキーを使用して、ロールを引き受けるときにソースIDを設定したプリンシパルによる AWS でのアクションを許可できます。ロールの指定されたソースIDのアクティビティが [AWS CloudTrail](#) に表示されます。これにより、管理者は AWS のロールで誰または何がアクションを実行したかを簡単に判断できます。

[sts:RoleSessionName](#) とは異なり、ソースIDを設定した後は、値を変更できません。これは、ロールが実行するすべてのアクションのリクエストコンテキストに存在します。この値は、セッション認証情報を使用して別のロールを引き受けるときに、後続のロールセッションに保持されます。別のロールからあるロールを引き受けると、[ロールの連鎖](#)と呼ばれます。

[sts:SourceIdentity](#) キーは、プリンシパルが、AWS STS `assume-role` CLI コマンドまたは AWS STS `AssumeRole` API オペレーションを使用してロールを引き受けるときに、最初にソース ID を設定するときに、リクエストに存在します。aws:SourceIdentity キーは、ソースIDが設定されているロールセッションで実行されるすべてのアクションの要求に存在します。

アカウント `CriticalRole` の `111122223333` の次のロール信頼ポリシーには、Saanvi または Diego に設定されているソース ID を持たないプリンシパルがロールを引き受けられることを防ぐ `aws:SourceIdentity` のための条件が含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleIfSourceIdentity",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::123456789012:role/CriticalRole"},
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi","Diego"]
        }
      }
    }
  ]
}
```

ソース ID 情報の使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

## ec2:RoleDelivery

このキーを使用して、署名付きリクエストのインスタンスメタデータサービスのバージョンを、Amazon EC2 の IAM ロール認証情報と比較します。インスタンスメタデータサービスは、所定のリクエストについて、IMDSv2 に固有の PUT または GET ヘッダーがそのリクエストに存在するかどうかによって、IMDSv1 と IMDSv2 リクエストを区別します。

- 可用性 — このキーは、ロールセッションが Amazon EC2 インスタンスで作成されるときは必ずリクエストコンテキストに含まれます。
- データ型 – [数値](#)
- 値タイプ — 単一値
- 値の例 – 1.0、2.0

ローカルコードまたはユーザーに IMDSv2 を使用させるように、各インスタンスのインスタンスメタデータサービス (IMDS) を設定することができます。IMDSv2を使用しなければならないように指定すると、IMDSv1はもう機能しなくなります。

- インスタンスメタデータサービスバージョン 1 (IMDSv1) – リクエスト/レスポンスメソッド
- インスタンスメタデータサービスバージョン 2 (IMDSv2) – セッション指向メソッド

IMDSv2 を使用するようにインスタンスを設定する方法については、「[インスタンスメタデータオプションの設定](#)」を参照してください。

次の例では、リクエストコンテキストの `ec2:RoleDelivery` の値が 1.0 (IMDSv1) の場合、アクセスが拒否されます。このポリシーステートメントは、リクエストが Amazon EC2 ロールの認証情報によって署名されていない場合は無効となるため、一般的に適用できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireAllEc2RolesToUseV2",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "NumericLessThan": {
          "ec2:RoleDelivery": "2.0"
        }
      }
    }
  ]
}
```

詳細については、「[インスタンスメタデータを使用する際のサンプルポリシー](#)」を参照してください。

#### ec2:SourceInstanceArn

このキーを使用して、ロールのセッション生成元インスタンスの ARN を比較します。

- 可用性 — このキーは、ロールセッションが Amazon EC2 インスタンスで作成される場合は必ずリクエストコンテキストに含まれます。
- データ型 – [ARN](#)
- 値タイプ — 単一値
- 値の例 – `arn:aws:ec2:us-west-2:111111111111:instance/instance-id`

ポリシーの例については、「[特定のインスタンスが他の AWS サービスでリソースを表示できるようにする](#)」を参照してください。

#### glue:RoleAssumedBy

AWS Glue サービスは、AWS Glue がお客様に変わってサービスロールを使用してリクエストを実行する AWS API リクエストごとに条件キーを設定します (ジョブまたは開発者のエンドポイントではなく、AWS Glue サービスによる直接のリクエスト)。このキーを使用して、AWS リソースへの呼び出しが AWS Glue サービスからのものであるかどうかを確認します。

- 可用性 – このキーは、AWS Glue がお客様に代わってサービスロールを使用してリクエストを行うときに、リクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ – 単一値
- 値の例 – このキーは常に `glue.amazonaws.com` に設定されます。

次の例では、AWS Glue サービスが Amazon S3 バケットからオブジェクトを取得できるようにする条件を追加します。

```
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::confidential-bucket/*",
  "Condition": {
    "StringEquals": {
      "glue:RoleAssumedBy": "glue.amazonaws.com"
    }
  }
}
```

#### glue:CredentialIssuingService

AWS Glue サービスは、ジョブまたは開発者エンドポイントから取得したサービスロールを使用して、AWS API リクエストごとにこのキーを設定します。このキーを使用して、AWS リソースへの呼び出しが AWS Glue ジョブまたは開発者エンドポイントからのものであるかどうかを確認します。

- 可用性 – このキーは、AWS Glue がジョブまたは開発者エンドポイントからリクエストを実行するときに、リクエストコンテキストに含まれます。
- データ型 - [文字列](#)

- 値タイプ — 単一値
- 値の例 – このキーは常に `glue.amazonaws.com` に設定されます。

次の例では、AWS Glue ジョブで使用される IAM ロールにアタッチされる条件を追加します。これで、特定のアクションが、ロールセッションが AWS Glue ジョブランタイム環境で使用されているかどうかに基づいて、許可/拒否されます。

```
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::confidential-bucket/*",
  "Condition": {
    "StringEquals": {
      "glue:CredentialIssuingService": "glue.amazonaws.com"
    }
  }
}
```

#### lambda:SourceFunctionArn

このキーを使用して、IAM ロール認証情報が配信された Lambda 関数 ARN を識別します。Lambda サービスは、関数の実行環境から送信される AWS API リクエストごとにこのキーを設定します。このキーを使用して、AWS リソースへの呼び出しが特定の Lambda 関数のコードからのものであるかどうかを確認します。Lambda は、CloudWatch へのログの書き込みや X-Ray へのトレース送信など、実行環境外から送信される一部のリクエストにもこのキーを設定します。

- 可用性 – このキーは、Lambda 関数コードが呼び出されるときは必ずリクエストコンテキストに含まれます。
- データ型 – [ARN](#)
- 値タイプ — 単一値
- 値の例 – `arn:aws:lambda:us-east-1:123456789012:function:TestFunction`

次の例では、1 つの特定の Lambda 関数に、指定されたバケットへの `s3:PutObject` アクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```
{
  "Sid": "ExampleSourceFunctionArn",
  "Effect": "Allow",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "ArnEquals": {
      "lambda:SourceFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:source_lambda"
    }
  }
}
```

詳細については、「AWS Lambda デベロッパーガイド」の「[Lambda 実行環境での認証情報の使用](#)」を参照してください。

ssm:SourceInstanceArn

このキーを使用して、IAM ロール認証情報が配信された AWS Systems Manager マネージドインスタンス ARN を識別します。この条件キーは、リクエストが Amazon EC2 インスタンスプロファイルに関連付けられている IAM ロールを使用してマネージドインスタンスから実行された場合は存在しません。

- 可用性 – このキーは、ロール認証情報が AWS Systems Manager マネージドインスタンスに配信される場合は必ずリクエストコンテキストに含まれます。
- データ型 – [ARN](#)
- 値タイプ – 単一値
- 値の例 – arn:aws:ec2:us-west-2:111111111111:instance/instance-id

identitystore:UserId

このキーを使用して、署名付きリクエストの IAM アイデンティティセンターワークフォース ID を、ポリシーで指定された ID と比較します。

- 可用性 – このキーは、リクエストの呼び出し元が IAM アイデンティティセンターのユーザーである場合に含まれます。
- データ型 – [文字列](#)

- 値タイプ — 単一値
- 値の例 – 94482488-3041-7026-18f3-be45837cd0e4

AWS CLI、AWS API、AWS SDK を使用して [GetUserId](#) API にリクエストを実行することで、IAM アイデンティティセンターのユーザーの `UserId` を見つけることができます。

## ネットワークのプロパティ

次の条件キーを使用して、リクエストの発信元ネットワークまたは経由するネットワークの詳細を、ポリシーで指定したプリンシパルのプロパティと比較します。

### 目次

- [aws:SourceIp](#)
- [aws:SourceVpc](#)
- [aws:SourceVpce](#)
- [aws:VpcSourceIp](#)

### aws:SourceIp

このキーを使用して、リクエストの IP アドレスをポリシーで指定した IP アドレスと比較します。aws:SourceIp 条件キーは、パブリック IP アドレス範囲にのみ使用できます。

- 可用性 – このキーは、リクエストが VPC エンドポイントを使用してリクエストを行う場合を除き、リクエストコンテキストに含まれます。
- データ型 – [IP アドレス](#)
- 値タイプ — 単一値

aws:SourceIp 条件キーをポリシーで使用して、プリンシパルが指定された IP 範囲内からのみリクエストを行うことを許可できます。

#### Note

aws:SourceIp は IPv4 と IPv6 の両方の IP アドレスおよびレンジをサポートしています。IPv6 をサポートする AWS のサービスのリストについては、「Amazon VPC ユーザーガイド」の「[IPv6 をサポートする AWS のサービス](#)」を参照してください。

例えば、次の ID ベースのポリシーを IAM ロールにアタッチできます。このポリシーは、指定された IPv4 アドレス範囲から呼び出しを実行する場合に、ユーザーがオブジェクトを DOC-EXAMPLE-BUCKET3 Amazon S3 バケットに格納することを許可します。このポリシーでは、[転送アクセスセッション](#) がユーザーに代わってこの操作を実行する AWS サービスも許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PrincipalPutObjectIfIpAddress",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "203.0.113.0/24"
        }
      }
    }
  ]
}
```

IPv4 と IPv6 の両方のアドレス指定をサポートするネットワークからのアクセスを制限する必要がある場合は、IAM ポリシー条件に IPv4 と IPv6 のアドレス、または IP アドレスの範囲を含めることができます。次の ID ベースのポリシーは、ユーザーが指定された IPv4 または IPv6 のいずれかのアドレス範囲から呼び出しを実行した場合に、オブジェクトを DOC-EXAMPLE-BUCKET3 Amazon S3 バケットに格納することを許可します。IAM ポリシーに IPv6 アドレス範囲を含める前に、使用している AWS のサービスが IPv6 をサポートしていることを確認してください。IPv6 をサポートする AWS のサービスのリストについては、「Amazon VPC ユーザーガイド」の「[IPv6 をサポートする AWS のサービス](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PrincipalPutObjectIfIpAddress",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
      "Condition": {
        "IpAddress": {
```

```
        "aws:SourceIp": [
            "203.0.113.0/24",
            "2001:DB8:1234:5678::/64"
        ]
    }
}
]
```

リクエスト実行元が Amazon VPC エンドポイントを使用するホストである場合、`aws:SourceIp` キーは使用できません。代わりに、[aws:VpcSourceIp](#)などの VPC 固有のキーを使用する必要があります。VPC エンドポイント使用の詳細については、「AWS PrivateLink ガイド」の「[VPC エンドポイントおよび VPC エンドポイントサービスの ID およびアクセス管理](#)」を参照してください。

#### `aws:SourceVpc`

このキーを使用して、リクエストが VPC エンドポイントのアタッチされている VPC を通過するかどうかを確認します。ポリシーでは、このキーを使用して、特定の VPC にのみアクセスを許可できます。詳細については、[Amazon Simple Storage Service ユーザーガイド](#)の「特定の VPC へのアクセスの制限」を参照してください。

- 可用性 – このキーは、リクエストが VPC エンドポイントを使用してリクエストを行う場合にのみリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

#### `aws:SourceVpce`

このキーを使用して、リクエストの VPC エンドポイント識別子をポリシーで指定したエンドポイント ID と比較します。ポリシーでは、このキーを使用して、特定の VPC エンドポイントへのアクセスを制限できます。詳細については、[Amazon Simple Storage Service ユーザーガイド](#)の「特定の VPC エンドポイントへのアクセスの制限」を参照してください。

- 可用性 – このキーは、リクエストが VPC エンドポイントを使用してリクエストを行う場合にのみリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

## aws:VpcSourceIp

このキーを使用して、リクエストの作成元の IP アドレスと、ポリシーで指定した IP アドレスを比較します。ポリシーでは、リクエストが指定された IP アドレスから送信され、VPC エンドポイントを経由する場合にのみキーが一致します。

- 可用性 – このキーは、リクエストが VPC エンドポイントを使用して行われた場合にのみリクエストコンテキストに含まれます。
- データ型 – [IP アドレス](#)
- 値タイプ – 単一値

詳細については、「Amazon VPC ユーザーガイド」の「[VPC エンドポイントによるサービスのアクセス制御](#)」を参照してください。

### Note

aws:VpcSourceIp は IPv4 と IPv6 の両方の IP アドレスおよびレンジをサポートしています。IPv6 をサポートする AWS のサービスのリストについては、「Amazon VPC ユーザーガイド」の「[IPv6 をサポートする AWS のサービス](#)」を参照してください。

## リソースのプロパティ

次の条件キーを使用して、リクエストの送信先リソースの詳細を、ポリシーで指定したリソースのプロパティと比較します。

### 目次


- [aws:ResourceAccount](#)
- [aws:ResourceOrgPaths](#)
- [aws:ResourceOrgID](#)
- [aws:ResourceTag/tag-key](#)

## aws:ResourceAccount

このキーを使用して、ポリシーのリソースアカウントと要求されたリソース所有者の [AWS アカウント ID](#) を比較します。その後、リソースを所有するアカウントに応じて、そのリソースへのアクセスを許可または拒否することができます。

- 可用性 - このキーは常時ほとんどのサービスアクションのリクエストコンテキストに含まれます。以下のアクションではこのキーをサポートしていません。
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store - すべてのアクション
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`
    - `ec2>CreateTransitGatewayPeeringAttachment`
    - `ec2>CreateVolume`
    - `ec2>CreateVpcEndpoint`
    - `ec2>CreateVpcPeeringConnection`
    - `ec2>DeleteTransitGatewayPeeringAttachment`
    - `ec2>DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents` – 2023 年 3 月 2 日以前にそのイベントバスがクロスアカウントの EventBridge ターゲットとして設定されていた場合、EventBridge PutEvents は別のアカウントのイベントバスを呼び出します。詳細については、「Amazon EventBridge ユーザーガイド」の「[他の AWS アカウントからのイベントを許可するアクセス許可の付与](#)」を参照してください。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`

- `macie2:AcceptInvitation`
- Amazon OpenSearch Service
  - `es:AcceptInboundConnection`
  - `es:CreateOutboundConnection`
- Amazon Route 53
  - `route53:AssociateVpcWithHostedZone`
  - `route53:CreateVPCAssociationAuthorization`
  - `route53>DeleteVPCAssociationAuthorization`
  - `route53:DisassociateVPCFromHostedZone`
  - `route53:ListHostedZonesByVPC`
- AWS Security Hub
  - `securityhub:AcceptAdministratorInvitation`
- データ型 - [文字列](#)
- 値タイプ — 単一値

 Note

上記のサポートされていないアクションに関するその他の考慮事項については、[data-perimeter-policy-examples](#) リポジトリを参照してください。

このキーはリクエストで評価されたリソースを持つアカウントの AWS アカウント ID に相当します。

アカウントのリソースのほとんどで、[ARN](#) にそのリソースの所有者アカウント ID が含まれています。Amazon S3 バケットなどの特定のリソースでは、リソース ARN にアカウント ID は含まれていません。次の 2 つの例は、ARN にアカウント ID を持つリソースと、アカウント ID を持たない Amazon S3 ARN の違いを示しています。

- `arn:aws:iam::123456789012:role/AWSExampleRole` – アカウント 123456789012 内で作成および所有されている IAM ロール。
- `arn:aws:s3:::DOC-EXAMPLE-BUCKET2` — ARN に表示されていない 111122223333 アカウント内で作成および所有されている Amazon S3 バケット。

AWS コンソールや API、または CLI を使用して、すべてのリソースおよび対応する ARN を検索します。

リソース所有者のアカウント ID に応じて、リソースへのアクセス権を拒否するポリシーを記述します。例えば、リソースが指定されたアカウントに属さない場合に、以下の ID ベースポリシーでは指定されたリソースへのアクセスを拒否します。

このポリシーを使用するには、イタリック体のプレースホルダーテキストをお客様のアカウント情報と置き換えます。

### Important

このポリシーでは、アクションを許可しません。代わりに、ステートメントに記載され、リストされたアカウントに属さないすべてのリソースへのアクセスを明示的に拒否する Deny 効果を使用します。特定のリソースへのアクセスを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInteractionWithResourcesNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:account:*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "account"
          ]
        }
      }
    }
  ]
}
```

このポリシーでは、特定の AWS アカウント でリソースを所有する場合を除き、特定の AWS サービスのすべてのリソースへのアクセスを拒否します。



**Note**

一部の AWS のサービス では他の AWS アカウント でホストされている AWS 所有リソースへのアクセスを必要とします。ID ベースのポリシーで `aws:ResourceAccount` を使用する場合は、これらのリソースにアクセスする ID の機能に影響を与える可能性があります。

AWS Data Exchange などの特定の AWS サービスでは、通常のオペレーションで AWS アカウントの外部のリソースへのアクセスに依存しています。ポリシーに `aws:ResourceAccount` 要素を使う場合、それらのサービスの免除を指定するために追加のステートメントを含めます。ポリシー [AWS: AWS Data Exchange を除くアカウント外の Amazon S3 リソースへのアクセスを拒否する](#) の例では、サービスが所有するリソースの例外を定義しながら、リソースアカウントに応じてアクセスを拒否する方法を示しています。

独自のカスタムポリシーを作成する場合に、このポリシーの例をテンプレートとして使用します。詳細については、サービスの [ドキュメント](#) を参照してください。

**aws:ResourceOrgPaths**

このキーを使用してアクセス済みリソースの AWS Organizations パスをポリシー内のパスと比較します。ポリシーでは、この条件キーによって、このリソースが AWS Organizations の指定した組織ルートまたは組織単位 (OU) 内のアカウントメンバーに属していることを保証します。AWS Organizations パスは、Organizations エンティティの構造をテキストで表記したものです。パスの使用と理解の詳細については、「[AWS Organizations エンティティパスを理解する](#)」を参照してください。

- 可用性 – このキーは、リソースを所有するアカウントが組織のメンバーである場合にのみ、リクエストコンテキストに含まれます。このグローバル条件キーでは以下のアクションがサポートされません。
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store - すべてのアクション
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`

- `ec2:AcceptVpcPeeringConnection`
- `ec2:CopyImage`
- `ec2:CopySnapshot`
- `ec2:CreateTransitGatewayPeeringAttachment`
- `ec2:CreateVolume`
- `ec2:CreateVpcEndpoint`
- `ec2:CreateVpcPeeringConnection`
- `ec2>DeleteTransitGatewayPeeringAttachment`
- `ec2>DeleteVpcPeeringConnection`
- `ec2:RejectTransitGatewayPeeringAttachment`
- `ec2:RejectVpcEndpointConnections`
- `ec2:RejectVpcPeeringConnection`
- Amazon EventBridge
  - `events:PutEvents` – 2023 年 3 月 2 日以前にそのイベントバスがクロスアカウントの EventBridge ターゲットとして設定されていた場合、EventBridge PutEvents は別のアカウントのイベントバスを呼び出します。詳細については、「Amazon EventBridge ユーザーガイド」の「[他の AWS アカウントからのイベントを許可するアクセス許可の付与](#)」を参照してください。
- Amazon GuardDuty
  - `guardduty:AcceptAdministratorInvitation`
- Amazon Macie
  - `macie2:AcceptInvitation`
- Amazon OpenSearch Service
  - `es:AcceptInboundConnection`
  - `es:CreateOutboundConnection`
- Amazon Route 53
  - `route53:AssociateVpcWithHostedZone`
  - `route53>CreateVPCAssociationAuthorization`
  - `route53>DeleteVPCAssociationAuthorization`
  - `route53:DisassociateVPCFromHostedZone`
  - `route53>ListHostedZonesByVPC`

- AWS Security Hub
  - securityhub:AcceptAdministratorInvitation
- データ型 – [文字列](#) (リスト)
- 値タイプ — 複数値

**Note**

上記のサポートされていないアクションに関するその他の考慮事項については、[data-perimeter-policy-examples](#) リポジトリを参照してください。

`aws:ResourceOrgPaths` は複数の値を持つ条件キーです。複数値のキーは、リクエストコンテキストに複数の値を持つことができます。このキーを使用する場合は、[文字列条件演算子](#)とともに `ForAnyValue` または `ForAllValues` の集合演算子を使用する必要があります。複数値を持つ条件キーの詳細については、「[複数値のコンテキストキー](#)」を参照してください。

例えば次の条件では、その組織 `o-a1b2c3d4e5` に属するリソースに `True` を返します。ワイルドカードを含める場合は、条件演算子 [StringLike](#) を使用する必要があります。

```
"Condition": {
  "ForAnyValue:StringLike": {
    "aws:ResourceOrgPaths":["o-a1b2c3d4e5/*"]
  }
}
```

次の条件では、OU ID `ou-ab12-11111111` を持つリソースには `True` が返されます。OU `ou-ab12-11111111`、または子 OU のいずれかにアタッチするアカウントが所有するリソースと一致することになります。

```
"Condition": { "ForAnyValue:StringLike" : {
  "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/*"]
}}
```

次の条件が OU ID `ou-ab12-22222222` に直接アタッチしたアカウントが所有するリソースに `True` が返されますが、子 OU には返されません。次の例では [StringEquals](#) 条件演算子を使用して、ワイルドカード一致ではなく OU ID の完全一致要件を指定します。

```
"Condition": { "ForAnyValue:StringEquals" : {
```

```
"aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

### Note

一部の AWS のサービスでは他の AWS アカウントでホストされている AWS 所有リソースへのアクセスを必要とします。ID ベースのポリシーで `aws:ResourceOrgPaths` を使用する場合は、これらのリソースにアクセスする ID の機能に影響を与える可能性があります。

AWS Data Exchange などの特定の AWS サービスでは、通常のオペレーションで AWS アカウントの外部のリソースへのアクセスに依存しています。ポリシーに `aws:ResourceOrgPaths` キーを使う場合、それらのサービスの免除を指定するために追加のステートメントを含めます。ポリシー [AWS: AWS Data Exchange を除くアカウント外の Amazon S3 リソースへのアクセスを拒否する](#) の例では、サービスが所有するリソースの例外を定義しながら、リソースアカウントに応じてアクセスを拒否する方法を示しています。類似ポリシーを作成して、サービス所有のリソースを考慮しながら、`aws:ResourceOrgPaths` キーを使用して組織単位 (OU) 内のリソースへのアクセスを制限できます。

独自のカスタムポリシーを作成する場合に、このポリシーの例をテンプレートとして使用します。詳細については、サービスの [ドキュメント](#) を参照してください。

### aws:ResourceOrgID

このキーを使用して、ポリシーで指定した識別子とリクエストしたリソースが属する AWS の Organizations 内の組織の識別子を比較します。

- 可用性 – このキーは、リソースを所有するアカウントが組織のメンバーである場合にのみ、リクエストコンテキストに含まれます。このグローバル条件キーでは以下のアクションがサポートされません。
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store - すべてのアクション
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`

- ec2:AcceptVpcEndpointConnections
- ec2:AcceptVpcPeeringConnection
- ec2:CopyImage
- ec2:CopySnapshot
- ec2>CreateTransitGatewayPeeringAttachment
- ec2:CreateVolume
- ec2>CreateVpcEndpoint
- ec2>CreateVpcPeeringConnection
- ec2>DeleteTransitGatewayPeeringAttachment
- ec2>DeleteVpcPeeringConnection
- ec2:RejectTransitGatewayPeeringAttachment
- ec2:RejectVpcEndpointConnections
- ec2:RejectVpcPeeringConnection
- Amazon EventBridge
  - events:PutEvents – 2023 年 3 月 2 日以前にそのイベントバスがクロスアカウントの EventBridge ターゲットとして設定されていた場合、EventBridge PutEvents は別のアカウントのイベントバスを呼び出します。詳細については、「Amazon EventBridge ユーザーガイド」の「[他の AWS アカウントからのイベントを許可するアクセス許可の付与](#)」を参照してください。
- Amazon GuardDuty
  - guardduty:AcceptAdministratorInvitation
- Amazon Macie
  - macie2:AcceptInvitation
- Amazon OpenSearch Service
  - es:AcceptInboundConnection
  - es>CreateOutboundConnection
- Amazon Route 53
  - route53:AssociateVpcWithHostedZone
  - route53>CreateVPCAssociationAuthorization
  - route53>DeleteVPCAssociationAuthorization
  - route53:DisassociateVPCFromHostedZone

- `route53:ListHostedZonesByVPC`
- AWS Security Hub
  - `securityhub:AcceptAdministratorInvitation`
- データ型 - [文字列](#)
- 値タイプ — 単一値

#### Note

上記のサポートされていないアクションに関するその他の考慮事項については、[data-perimeter-policy-examples](#) リポジトリを参照してください。

このグローバルキーは、与えられたリクエストのリソース組織 ID を返します。これにより、[ID ベースポリシー](#)の Resource 要素で指定した組織のすべてのリソースに適用されるルールが作成できます。[組織 ID](#) は、条件要素で指定できます。アカウントを追加および削除する場合、`aws:ResourceOrgID` のキーを含むポリシーには正しいアカウントが自動的に組み込まれ、手動で更新する必要はありません。

例えば、以下のポリシーでは、Amazon S3 リソースがリクエストを行うプリンシパルと同じ組織に属している場合を除き、プリンシパルは `policy-genius-dev` リソースにオブジェクトを追加できません。

#### Important

このポリシーでは、アクションを許可しません。代わりに、ステートメントに記載され、リストされたアカウントに属さないすべてのリソースへのアクセスを明示的に拒否する Deny 効果を使用します。特定のリソースへのアクセスを許可する他のポリシーと組み合わせてこのポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "DenyPutObjectToS3ResourcesOutsideMyOrganization",
    "Effect": "Deny",
    "Action": "s3:PutObject",
```

```
"Resource": "arn:partition:s3:::policy-genius-dev/*",
"Condition": {
  "StringNotEquals": {
    "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
  }
}
}
```

### Note

一部の AWS のサービスでは他の AWS アカウント でホストされている AWS 所有リソースへのアクセスを必要とします。ID ベースのポリシーで `aws:ResourceOrgID` を使用する場合は、これらのリソースにアクセスする ID の機能に影響を与える可能性があります。

AWS Data Exchange などの特定の AWS サービスでは、通常のオペレーションで AWS アカウントの外部のリソースへのアクセスに依存しています。ポリシーに `aws:ResourceOrgID` キーを使う場合、それらのサービスの免除を指定するために追加のステートメントを含めます。ポリシー [AWS: AWS Data Exchange を除くアカウント外の Amazon S3 リソースへのアクセスを拒否する](#) の例では、サービスが所有するリソースの例外を定義しながら、リソースアカウントに応じてアクセスを拒否する方法を示しています。類似ポリシーを作成して、サービス所有のリソースを考慮しながら、`aws:ResourceOrgID` キーを使用して組織内のリソースへのアクセスを制限できます。

独自のカスタムポリシーを作成する場合に、このポリシーの例をテンプレートとして使用します。詳細については、サービスの [ドキュメント](#) を参照してください。

次のビデオでは、ポリシーで `aws:ResourceOrgID` 条件キーを使用する方法について詳しく説明します。

[Ensure identities and networks can only be used to access trusted resources](#) (ID とネットワークを確実に信頼できるリソースへのアクセスにのみ使用できるようにする)

`aws:ResourceTag/tag-key`

このキーを使用して、ポリシーで指定したタグキーバリューのペアと、リソースにアタッチされているキーバリューのペアを比較します。たとえば、リソースに値 "Marketing" の付いたタグキー "Dept" がアタッチされている場合にのみ、そのリソースへのアクセスを許可するように要求することができます。詳細については、「[AWS のリソースへのアクセスの制御](#)」を参照してください。

- 可用性 – このキーは、リクエストコンテキスト (リクエストされたリソースにすでにアタッチしたタグがある場合) またはアタッチしたタグを持つリソースを作成するリクエストに含まれます。このキーは、[タグに基づいて認証をサポートする](#) リソースに対してのみ返されます。タグキーバリューのペアごとに 1 つのコンテキストキーがあります。
- データ型 - [文字列](#)
- 値タイプ — 単一値

このコンテキストキーは "aws:ResourceTag/*tag-key*":"*tag-value*" という形式です。ここで *tag-key* および *tag-value* はタグキーバリューのペアです。タグのキーと値は大文字と小文字が区別されません。つまり、ポリシーの条件要素で "aws:ResourceTag/TagKey1": "Value1" で指定した場合、その条件は TagKey1 または tagkey1 という名前のリソースタグキーに一致しますが、その両方には一致しません。

aws:ResourceTag キーを使用して IAM リソースへのアクセスを制御する例については、「[AWS のリソースへのアクセスの制御](#)」を参照してください。

aws:ResourceTag キーを使用して他の AWS リソースへのアクセスを制御する例については、「[タグを使用した AWS リソースへのアクセスの制御](#)」をご参照ください。

属性ベースのアクセス制御 (ABAC) の aws:ResourceTag 条件キーの使用に関するチュートリアルについては、「[IAM チュートリアル: タグに基づいて AWS リソースにアクセスするためのアクセス許可を定義する](#)」をご参照ください。

## リクエストのプロパティ

次の条件キーを使用して、リクエスト自体とリクエストの内容の詳細を、ポリシーで指定したリクエストのプロパティと比較します。

### 目次

- [aws:CalledVia](#)
- [aws:CalledViaFirst](#)
- [aws:CalledViaLast](#)
- [aws:ViaAWSService](#)
- [aws:CurrentTime](#)
- [aws:EpochTime](#)
- [aws:referer](#)



- [aws:RequestedRegion](#)
- [aws:RequestTag/tag-key](#)
- [aws:TagKeys](#)
- [aws:SecureTransport](#)
- [aws:SourceArn](#)
- [aws:SourceAccount](#)
- [aws:SourceOrgPaths](#)
- [aws:SourceOrgID](#)
- [aws:UserAgent](#)

### aws:CalledVia

このキーを使用して、ポリシー内のサービスと、IAM プリンシパル (ユーザーまたはロール) に代わってリクエストを実行したサービスを比較します。プリンシパルが AWS サービスに対してリクエストを実行すると、そのサービスはプリンシパルの認証情報を使用して、後続のリクエストを他のサービスに対して実行することがあります。aws:CalledVia キーには、プリンシパルに代わってリクエストを実行したチェーン内の各サービスの順序付きリストが含まれます。

例えば、AWS CloudFormation を使用して Amazon DynamoDB テーブルからの読み取りと書き込みを行うことができます。次に、DynamoDB は AWS Key Management Service ( AWS KMS ) によって提供される暗号化を使用します。

- 可用性 – このキーは、aws:CalledVia をサポートするサービスが IAM プリンシパルの認証情報を使用して別のサービスにリクエストを実行する場合に、リクエスト内に存在します。サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合、このキーは存在しません。また、プリンシパルが直接呼び出しを行う場合にも存在しません。
- データ型 – [文字列](#) (リスト)
- 値タイプ – 複数値

ポリシーで aws:CalledVia 条件キーを使用するには、AWS サービスリクエストを許可または拒否するサービスプリンシパルを提供する必要があります。AWS は、aws:CalledVia で次のサービスの使用をサポートしています。

## サービスプリンシパル

aoss.amazonaws.com

athena.amazonaws.com

backup.amazonaws.com

cloud9.amazonaws.com

cloudformation.amazonaws.com

databrew.amazonaws.com

dataexchange.amazonaws.com

dynamodb.amazonaws.com

imagebuilder.amazonaws.com

kms.amazonaws.com

mgn.amazonaws.com

nimble.amazonaws.com

omics.amazonaws.com

ram.amazonaws.com

robomaker.amazonaws.com

servicecatalog-appregistry.amazonaws.com

sqlworkbench.amazonaws.com

ssm-guiconnect.amazonaws.com

いずれかのサービスがプリンシパルの認証情報を使用してリクエストを実行したときにアクセスを許可または拒否するには、[aws:ViaAWSService](#) 条件キーを使用します。この条件キーは、AWS サービスをサポートします。

`aws:CalledVia` キーは [複数値を持つキー](#) です。ただし、条件でこのキーを使用して順序を強制することはできません。上記の例を使用すると、ユーザー 1 は、AWS CloudFormation に対してリクエストを実行し、それにより AWS KMS が呼び出され、DynamoDB が呼び出されます。これらは 3 つの個別のリクエストです。AWS KMS への最後の呼び出しは、AWS CloudFormation を介し、次に DynamoDB を介して、ユーザー 1 によって実行されます。

この場合、リクエストコンテキストの `aws:CalledVia` キーには、`cloudformation.amazonaws.com`、`dynamodb.amazonaws.com` がこの順序で含まれます。呼び出しがリクエストのチェーン内のどこかで、DynamoDB を介して行われたことだけが重要である場合は、ポリシーでこの条件キーを使用できます。

たとえば、次のポリシーでは、AWS KMS という名前の `my-example-key` キーを管理できますが、DynamoDB がリクエスト元のサービスの 1 つである場合に限りです。 [ForAnyValue:StringEquals](#) 条件演算子により、DynamoDB が呼び出し元サービスの 1 つであることが保証されます。プリンシパルが AWS KMS を直接呼び出す場合、条件は `false` を返し、リクエストはこのポリシーで許可されません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KmsActionsIfCalledViaDynamodb",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": ["dynamodb.amazonaws.com"]
        }
      }
    }
  ]
}
```

チェーン内の最初または最後の呼び出しを行うサービスを強制する場合は、[aws:CalledViaLast](#) キー [aws:CalledViaFirst](#) とキーを使用できます。たとえば、次のポリシーでは、`my-example-key` という名前のキーを AWS KMS で管理できます。これらの AWS KMS オペレーションは、複数のリクエストがチェーンに含まれている場合にのみ許可されます。最初のリクエストは AWS CloudFormation を介して、最後のリクエストは DynamoDB を介して実行される必要があります。他のサービスがチェーンの途中でリクエストを実行しても、オペレーションは許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KmsActionsIfCalledViaChain",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
      "Condition": {
        "StringEquals": {
          "aws:CalledViaFirst": "cloudformation.amazonaws.com",
          "aws:CalledViaLast": "dynamodb.amazonaws.com"
        }
      }
    }
  ]
}
```

サービスで IAM プリンシパルの認証情報を使用して別のサービスを呼び出す場合、[aws:CalledViaFirst](#) キーと [aws:CalledViaLast](#) キーはリクエスト内に存在します。これらは、リクエストのチェーンで呼び出しを行った最初と最後のサービスを示します。たとえば、AWS CloudFormation が X Service という名前の別のサービスを呼び出し、それにより DynamoDB が呼び出され、AWS KMS が呼び出されるとします。AWS KMS への最後の呼び出しは、User 1 を介して AWS CloudFormation、DynamoDB の順に X Service によって実行されます。これは、最初に AWS CloudFormation を介して呼び出され、最後に DynamoDB を介して呼び出されました。

## aws:CalledViaFirst

このキーを使用して、ポリシー内のサービスと、IAM プリンシパル (ユーザーまたはロール) に代わってリクエストを実行した最初のサービスを比較します。詳細については、「[aws:CalledVia](#)」を参照してください。

- 可用性 – このキーは、サービスが IAM プリンシパルの認証情報を使用して、別のサービスに対して少なくとも 1 つの他のリクエストを実行する場合に、リクエスト内に存在します。サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合、このキーは存在しません。また、プリンシパルが直接呼び出しを行う場合にも存在しません。
- データ型 - [文字列](#)
- 値タイプ — 単一値

## aws:CalledViaLast

このキーを使用して、ポリシー内のサービスと、IAM プリンシパル (ユーザーまたはロール) に代わってリクエストを実行した最後のサービスを比較します。詳細については、「[aws:CalledVia](#)」を参照してください。

- 可用性 – このキーは、サービスが IAM プリンシパルの認証情報を使用して、別のサービスに対して少なくとも 1 つの他のリクエストを実行する場合に、リクエスト内に存在します。サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合、このキーは存在しません。また、プリンシパルが直接呼び出しを行う場合にも存在しません。
- データ型 - [文字列](#)
- 値タイプ — 単一値

## aws:ViaAWSService

このキーを使用して、AWS サービスがユーザーに代わって別のサービスにリクエストを実行するかどうか確認します。

サービスが IAM プリンシパルの認証情報を使用し、プリンシパルに代わってリクエストを実行すると、リクエストコンテキストキーは `true` を返します。サービスが [サービスロール](#) または [サービスリンクロール](#) を使用してプリンシパルに代わって呼び出しを行う場合、コンテキストキーは `false` を

返します。リクエストコンテキストキーは、プリンシパルが直接呼び出しを行ったときも `false` を返します。

- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 – [ブール値](#)
- 値タイプ – 単一値

この条件キーを使用して、リクエストがサービスによって行われたかどうかに基づいてアクセスを許可または拒否できます。

#### aws:CurrentTime

このキーを使用して、リクエストの日時と、ポリシーで指定した日時を比較します。この条件キーを使用するポリシーの例を表示するには、「[AWS: 日付と時刻に基づいてアクセスを許可します](#)」を参照してください。

- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 – [日付](#)
- 値タイプ – 単一値

#### aws:EpochTime

このキーを使用して、リクエストの日時 ( epoch または Unix 時間 ) をポリシーで指定した値と比較します。このキーは、1970 年 1 月 1 日からの秒数も受け付けます。

- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 – [日付](#)、[数値](#)
- 値タイプ – 単一値

#### aws:referrer

このキーを使用して、クライアントブラウザでリクエストを参照したユーザーとポリシーで指定したリファラーを比較します。aws:referrer リクエストコンテキストの値は、HTTP ヘッダーで呼び出し元によって提供されます。Referer ヘッダーは、ウェブページ上のリンクを選択すると、ウェブブラウザリクエストに含まれます。Referer ヘッダーには、リンクが選択されたウェブページの URL が含まれます。

- 可用性 – このキーは、ブラウザでウェブページ URL からリンクすることによって AWS リソースへのリクエストが呼び出された場合にのみ、リクエストコンテキストに含まれます。このキーは、AWS リソースへのアクセスにブラウザリンクを使用しないため、プログラムによるリクエストには含まれません。
- データ型 - [文字列](#)
- 値タイプ — 単一値

例えば、URL または直接 API 呼び出しを使用して、Amazon S3 オブジェクトに直接アクセスできます。詳細については、「[ウェブブラウザを使用して Amazon S3 API オペレーションを直接呼び出す](#)」をご参照ください。ウェブページに存在する URL から Amazon S3 オブジェクトにアクセスすると、ソースウェブページの URL が `aws:referer` で使用されます。ブラウザに URL を入力して Amazon S3 オブジェクトにアクセスすると、`aws:referer` は存在しません。API を直接呼び出すと、`aws:referer` も存在していません。ポリシーで `aws:referer` 条件キーを使用して、会社のドメイン内のウェブページ上のリンクなど、特定のリファラーからのリクエストを許可できます。

#### Warning

このキーは慎重に使用する必要があります。一般に知られている参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の `aws:referer` 値を提供することができます。そのため `aws:referer` は、不正な当事者から AWS にリクエストが直接行われることを防止するために使用しないでください。このキーは、Amazon S3 に保存されているデジタルコンテンツなど、不正なサードパーティーサイトで参照されることから保護するためにのみ、お客様に提供されています。

## aws:RequestedRegion

このキーを使用して、リクエストで呼び出された AWS リージョンとポリシーで指定したリージョンを比較します。このグローバル条件キーを使用して、リクエストできるリージョンを制御できます。各サービスの AWS リージョンを表示するには、「[Amazon Web Services 全般のリファレンス](#)」の「[サービスエンドポイントとクォータ](#)」を参照してください。

- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

IAM などのグローバルサービスには、単一のエンドポイントがあります。このエンドポイントは、物理的に米国東部 (バージニア北部) リージョンにあるため、IAM 呼び出しは常に us-east-1 リージョンで行われます。たとえば、リクエストされたリージョンが us-west-2 でない場合にすべてのサービスへのアクセスを拒否するポリシーを作成すると、IAM 呼び出しは必ず失敗します。これを回避する方法の例を表示するには、「[Deny での NotAction の使用](#)」を参照してください。

#### Note

aws:RequestedRegion 条件キーを使用すると、呼び出されるサービスのエンドポイントを制御できますが、オペレーションの影響を制御することはできません。一部のサービスではリージョン間の影響があります。

例えば、Amazon S3 には複数のリージョンにまたがる API オペレーションがあります。

- 1 つのリージョン ( aws:RequestedRegion の条件キーの影響を受ける ) で s3:PutBucketReplication を呼び出すことはできますが、他のリージョンはレプリケーションの構成設定に基づいて影響を受けます。
- s3:CreateBucket を呼び出し、別のリージョンにバケットを作成できます。また、s3:LocationConstraint 条件キーを使用して該当するリージョンを制御できます。

このコンテキストキーを使用して、指定された一連のリージョン内の AWS サービスへのアクセスを制限できます。たとえば、以下のポリシーでは、AWS Management Console でのすべての Amazon EC2 インスタンスの表示をユーザーに許可します。ただし、変更できるインスタンスは、アイルランド (eu-west-1)、ロンドン (eu-west-2)、パリ (eu-west-3) のみです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InstanceConsoleReadOnly",
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "ec2:Export*",
        "ec2:Get*",
        "ec2:Search*"
      ],
      "Resource": "*"
    }
  ],
}
```



```
{
  "Sid": "InstanceWriteRegionRestricted",
  "Effect": "Allow",
  "Action": [
    "ec2:Associate*",
    "ec2:Import*",
    "ec2:Modify*",
    "ec2:Monitor*",
    "ec2:Reset*",
    "ec2:Run*",
    "ec2:Start*",
    "ec2:Stop*",
    "ec2:Terminate*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:RequestedRegion": [
        "eu-west-1",
        "eu-west-2",
        "eu-west-3"
      ]
    }
  }
}
```

### aws:RequestTag/tag-key

このキーを使用して、リクエストで渡されたタグキーバリューのペアと、ポリシーで指定したタグペアを比較します。たとえば、リクエストに「"Dept"」タグキーが含まれ、「"Accounting"」という値が含まれているかどうかを確認できます。詳細については、「[AWS リクエスト時のアクセスの制御](#)」を参照してください。

- 可用性 – このキーは、リクエストでタグのキー/値のペアが渡されたときにリクエストコンテキストに含まれます。複数のタグがリクエストで渡されると、タグキーバリューのペアごとに1つのコンテキストキーがあります。
- データ型 - [文字列](#)
- 値タイプ – 単一値

このコンテキストキーは "aws:RequestTag/*tag-key*":"*tag-value*" という形式です。ここで *tag-key* および *tag-value* はタグキーバリューのペアです。タグのキーと値は大文字と小文字が区別されません。つまり、ポリシーの条件要素で "aws:RequestTag/TagKey1": "Value1" で指定した場合、その条件は TagKey1 または tagkey1 という名前のリクエストタグキーに一致しますが、その両方には一致しません。

この例では、キーが単一値であっても、キーが異なる場合でもリクエストで複数のキー/値のペアが使用できることを示しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:::instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ],
        "aws:RequestTag/team": [
          "engineering"
        ]
      }
    }
  }
}
```

## aws:TagKeys

このキーを使用して、リクエスト内のタグキーとポリシーで指定したキーを比較します。ポリシーでタグを使用してアクセスを制御する場合は、aws:TagKeys 条件キーを使用して、許可されるタグキーを定義することをお勧めします。ポリシー例と詳細については、「[the section called “タグキーに基づいたアクセスの制御”](#)」を参照してください

- 可用性 – このキーは、オペレーションがリクエストのタグを渡すサポートしている場合にのみ、リクエストコンテキストに含まれます。
- データ型 – [文字列](#) (リスト)
- 値タイプ – 複数值

このコンテキストキーは "aws:TagKeys": "*tag-key*" という形式であり、ここで *tag-key* は値 (["Dept", "Cost-Center"] など) のないタグキーのリストです。

1つのリクエストに複数のタグとキーバリューのペアを含めることができるため、リクエストのコンテンツは [複数の値を持つリクエスト](#) である場合があります。この場合、ForAllValues または ForAnyValue 集合演算子を使用する必要があります。詳細については、「[複数值のコンテキストキー](#)」を参照してください。

一部のサービスでは、リソースの作成、変更、削除などのリソースオペレーションを使用したタグ付けをサポートしています。1回の呼び出しでタグ付けとオペレーションを許可するには、タグ付けアクションとリソース変更アクションの両方を含むポリシーを作成する必要があります。次に、aws:TagKeys 条件キーを使用して、リクエストで特定のタグキーを使用して適用できます。たとえば、だれかが Amazon EC2 スナップショットを作成するときにタグを制限するには、ポリシーに `ec2:CreateSnapshot` の作成アクションおよび `ec2:CreateTags` のタグ付けアクションを含める必要があります。aws:TagKeys を使用するこのシナリオのポリシーを表示するには、「Amazon EC2 ユーザーガイド」の [タグ付きのスナップショットの作成](#) に関するページを参照してください。

#### aws:SecureTransport

このキーを使用してリクエストが SSL を使用して送信されたかどうかを確認します。リクエストコンテキストは true または false を返します。ポリシーでは、リクエストが SSL を使用して送信された場合にのみ、特定のアクションを許可できます。

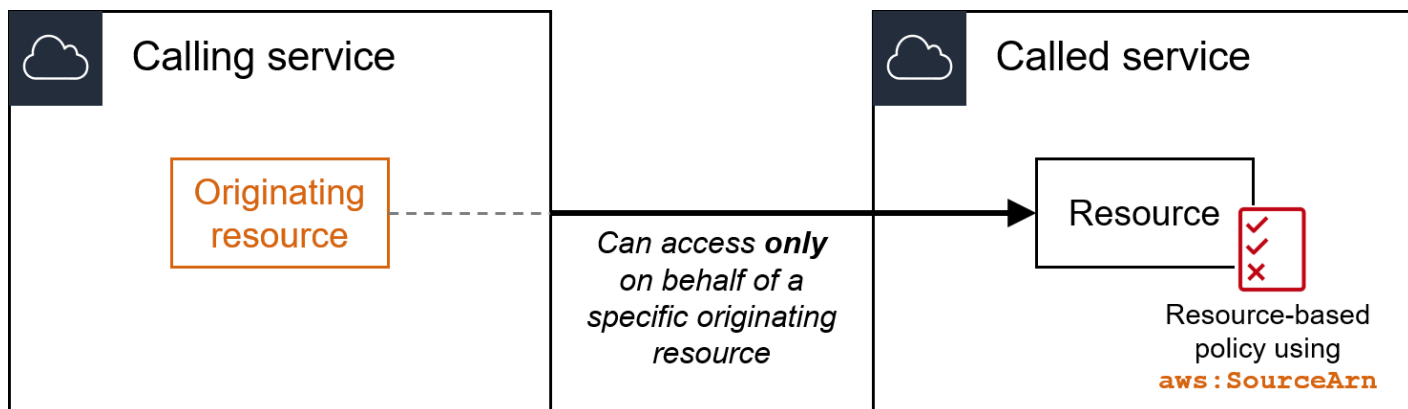
- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 – [ブール値](#)
- 値タイプ – 単一値

#### aws:SourceArn

このキーを使用して、サービス間リクエストを行っているリソースの [Amazon リソースネーム \(ARN\)](#) を、ポリシーで指定した ARN と比較します。ただし、このリクエストが AWS サービスプリンシパルによって実行された場合に限定されます。ソースの ARN にアカウント ID が含まれている場合は、aws:SourceArn で aws:SourceAccount を使用する必要はありません。

このキーは、リクエストを行うプリンシパルの ARN では機能しません。代わりに [aws:PrincipalArn](#) を使用してください。

- 可用性 — このキーは、設定によってサービス間リクエストがトリガーされたリソースに代わって、[AWS サービスプリンシパル](#)が直接リソースを呼び出している場合にのみリクエストコンテキストに含まれます。呼び出し元のサービスは、元のリソースの ARN を呼び出し先のサービスに渡します。



以下のサービスインテグレーションは、このグローバル条件キーをサポートしていません。

呼び出し元サービス (サービスプリンシパル)	呼び出し先サービス (リソーススペースのポリシー)	説明
logdelivery.elb.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>

### Note

AWS Security Token Service (AWS STS) と AWS Key Management Service (AWS KMS) とのすべてのサービス統合がサポートされているわけではありません。詳細については、呼び出し元サービスのドキュメントを参照してください。KMS キーの付与を介して AWS のサービスにより使用される KMS キーポリシーで `aws:SourceArn` を使用すると、予期しない動作が発生する可能性があります。

- データ型 - ARN、文字列

AWS では、ARN を比較する場合、[文字列演算子](#)の代わりに [ARN 演算子](#)を使用することをお勧めします。

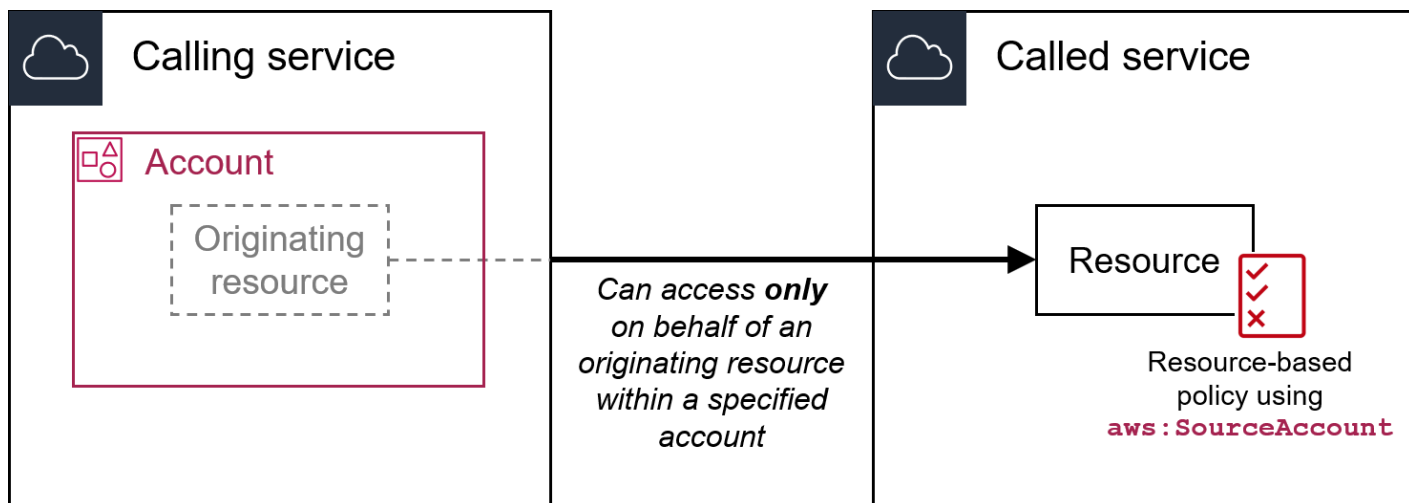
- 値タイプ — 単一値

この条件キーを使用して、サービス間のトランザクション中に AWS サービスが[混乱した代理](#)として使用されるのを防ぐことができます。このキーは、Principal が AWS のサービスプリンシパルであるリソースベースのポリシーでのみ使用してください。この条件キーの値を、リクエスト内のリソースの ARN に設定します。例えば、Amazon S3 バケットの更新によって Amazon SNS トピックの公開がトリガーされると、Amazon S3 サービスは sns:Publish API 操作を呼び出します。sns:Publish 操作を許可するトピックポリシーで、条件キーの値を Amazon S3 バケットの ARN に設定します。この条件キーが推奨される方法とタイミングについては、使用している AWS サービスのドキュメントを参照してください。

#### aws:SourceAccount

このキーを使用して、サービス間リクエストを行っているリソースのアカウント ID を、ポリシーで指定したアカウント ID と比較します。ただし、このリクエストが AWS サービスプリンシパルによって実行された場合に限定されます。

- 可用性 — このキーは、設定によってサービス間リクエストがトリガーされたリソースに代わって、[AWS サービスプリンシパル](#)が直接リソースを呼び出している場合にのみリクエストコンテキストに含まれます。呼び出し元のサービスは、オリジナルのリソースのアカウント ID を呼び出し先のサービスに渡します。



以下のサービスインテグレーションは、このグローバル条件キーをサポートしていません。

呼び出し元サービス (サービスプリンシパル)	呼び出し先サービス (リソーススペースのポリシー)	説明
logdelivery.elb.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>

### Note

AWS Security Token Service (AWS STS) と AWS Key Management Service (AWS KMS) とのすべてのサービス統合がサポートされているわけではありません。詳細については、呼び出し元サービスのドキュメントを参照してください。KMS キーの付与を介して AWS のサービスにより使用される KMS キーポリシーで `aws:SourceAccount` を使用すると、予期しない動作が発生する可能性があります。

- データ型 - [文字列](#)
- 値タイプ — 単一値

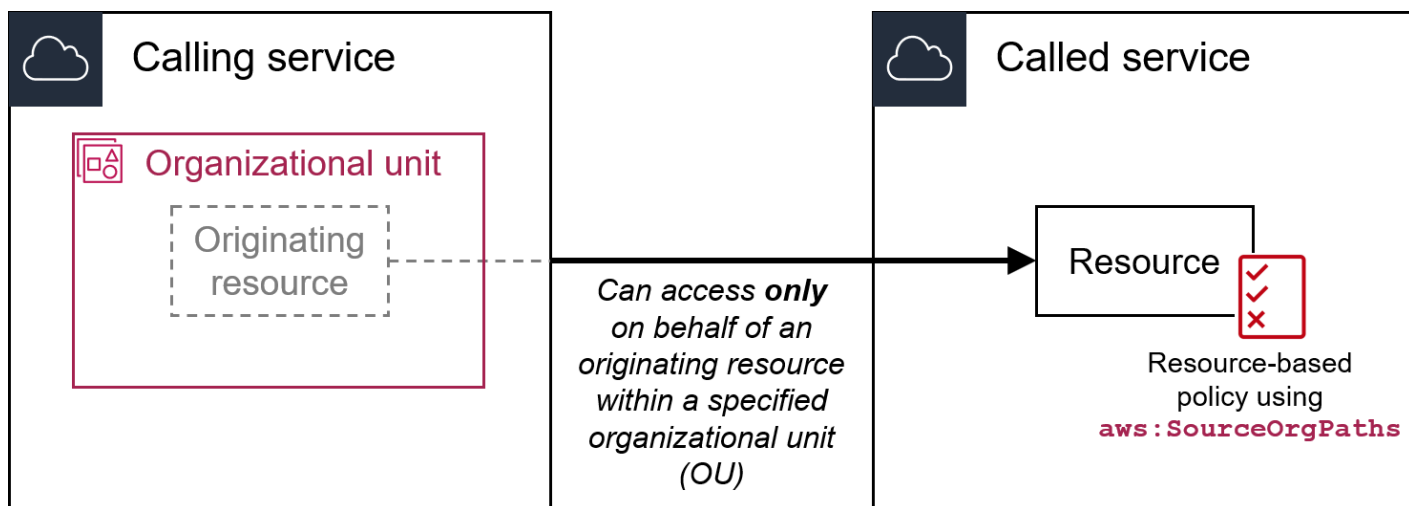
この条件キーを使用して、サービス間のトランザクション中に AWS サービスが [混乱した代理](#) として使用されるのを防ぐことができます。このキーは、Principal が AWS のサービスプリンシパルであるリソーススペースのポリシーでのみ使用してください。この条件キーの値を、リクエスト内のリソースのアカウント ID に設定します。例えば、Amazon S3 バケットの更新によって Amazon SNS トピックの公開がトリガーされると、Amazon S3 サービスは `sns:Publish` API 操作を呼び出します。`sns:Publish` 操作を許可するトピックポリシーで、条件キーの値を Amazon S3 バケットのアカウント ID に設定します。この条件キーが推奨される方法とタイミングについては、使用している AWS サービスのドキュメントを参照してください。

### `aws:SourceOrgPaths`

このキーを使用して、サービス間リクエストを行っているリソースの AWS Organizations パスを、ポリシーで指定した組織パスと比較します。ただし、このリクエストが AWS サービスプリンシパル

によって実行された場合に限定されます。Organizations パスは、Organizations エンティティの構造をテキストで表記したものです。パスの使用と理解の詳細については、「[AWS Organizations エンティティパスを理解する](#)」を参照してください。

- 可用性 — このキーがリクエストコンテキストに含まれるのは、組織のメンバーであるアカウントが所有するリソースに代わって、[AWSサービスプリンシパル](#)が直接リソースを呼び出している場合のみです。呼び出し元のサービスは、元のリソースの組織パスを呼び出し先のサービスに渡します。



以下のサービスインテグレーションは、このグローバル条件キーをサポートしていません。

呼び出し元サービス (サービスプリンシパル)	呼び出し先サービス (リソーススペースのポリシー)	説明
logdelivery.elb.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
すべてサービスプリンシパル	Amazon Lex ボット	<a href="#">AWS のサービスによる Amazon Lex ボットの使用を許可する</a>

**Note**

AWS Security Token Service (AWS STS) と AWS Key Management Service (AWS KMS) とのすべてのサービス統合がサポートされているわけではありません。詳細については、呼び出し元サービスのドキュメントを参照してください。KMS キーの付与を介して AWS のサービスにより使用される KMS キーポリシーで `aws:SourceOrgPaths` を使用すると、予期しない動作が発生する可能性があります。

- データ型 – [文字列](#) (リスト)
- 値タイプ — 複数値

この条件キーを使用して、サービス間のトランザクション中に AWS サービスが [混乱した代理](#) として使用されるのを防ぐことができます。このキーは、Principal が AWS のサービスプリンシパルであるリソースベースのポリシーでのみ使用してください。この条件キーの値を、リクエスト内のリソースの組織パスに設定します。例えば、Amazon S3 バケットの更新によって Amazon SNS トピックの公開がトリガーされると、Amazon S3 サービスは `sns:Publish` API 操作を呼び出します。`sns:Publish` 操作を許可するトピックポリシーで、条件キーの値を Amazon S3 バケットの組織パスに設定します。この条件キーが推奨される方法とタイミングについては、使用している AWS サービスのドキュメントを参照してください。

`aws:SourceOrgPaths` は複数の値を持つ条件キーです。複数値のキーは、リクエストコンテキストに複数の値を持つことができます。このキーを使用する場合は、[文字列条件演算子](#) とともに `ForAnyValue` または `ForAllValues` の集合演算子を使用する必要があります。複数値を持つ条件キーの詳細については、「[複数値のコンテキストキー](#)」を参照してください。

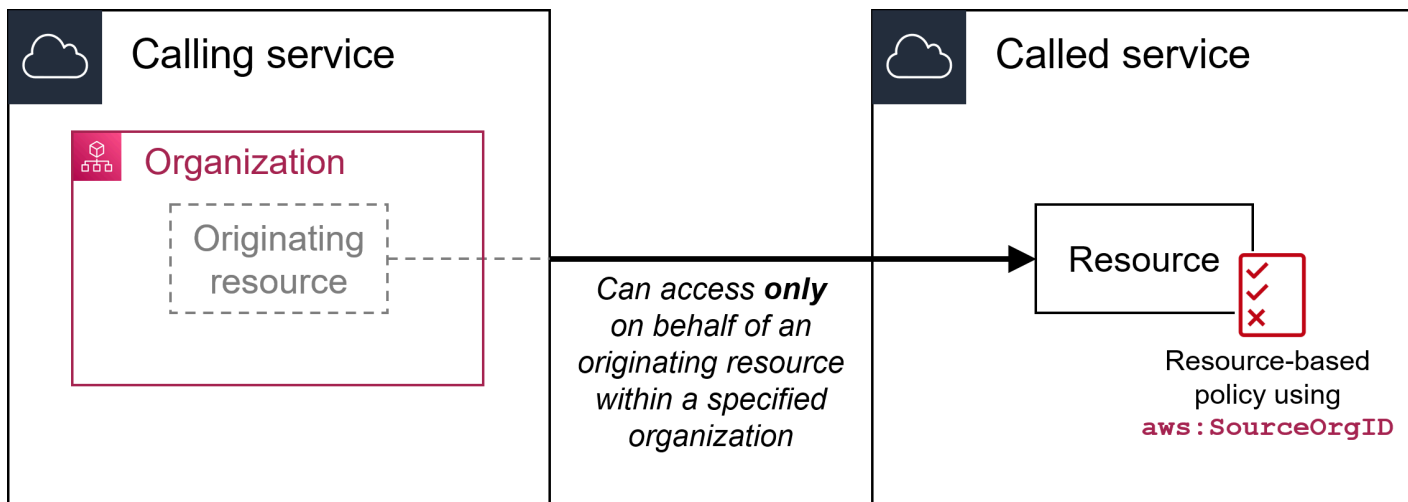
**aws:SourceOrgID**

このキーを使用して、サービス間リクエストを行っているリソースの [組織 ID](#) を、ポリシーで指定した組織 ID と比較します。ただし、このリクエストが AWS サービスプリンシパルによって実行された場合に限定されます。AWS Organizations の組織にアカウントを追加する、または組織からアカウントを削除する場合、`aws:SourceOrgID` のキーを含むポリシーには正しいアカウントが自動的に組み込まれるため、手動で更新する必要はありません。

- 可用性 — このキーがリクエストコンテキストに含まれるのは、組織のメンバーであるアカウントが所有するリソースに代わって、[AWS サービスプリンシパル](#) が直接リソースを呼び出している場



合のみです。呼び出し元のサービスは、元のリソースの組織 ID を呼び出し先のサービスに渡します。



以下のサービスインテグレーションは、このグローバル条件キーをサポートしていません。

呼び出し元サービス (サービスプリンシパル)	呼び出し先サービス (リソーススペースのポリシー)	説明
logdelivery.elb.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 バケット	<a href="#">Amazon S3 バケットで Elastic Load Balancing のアクセスログ記録を有効にする</a>
すべてサービスプリンシパル	Amazon Lex ボット	<a href="#">AWS のサービスによる Amazon Lex ボットの使用を許可する</a>

#### Note

AWS Security Token Service (AWS STS) と AWS Key Management Service (AWS KMS) とのすべてのサービス統合がサポートされているわけではありません。詳細については、呼び出し元サービスのドキュメントを参照してください。KMS キーの付与を介して AWS

のサービスにより使用される KMS キーポリシーで `aws:SourceOrgID` を使用すると、予期しない動作が発生する可能性があります。

- データ型 - [文字列](#)
- 値タイプ — 単一値

この条件キーを使用して、サービス間のトランザクション中に AWS サービスが [混乱した代理](#)として使用されるのを防ぐことができます。このキーは、Principal が AWS のサービスプリンシパルであるリソースベースのポリシーでのみ使用してください。この条件キーの値を、リクエスト内のリソースの組織 ID に設定します。例えば、Amazon S3 バケットの更新によって Amazon SNS トピックの公開がトリガーされると、Amazon S3 サービスは `sns:Publish` API 操作を呼び出します。`sns:Publish` 操作を許可するトピックポリシーで、条件キーの値を Amazon S3 バケットの組織 ID に設定します。この条件キーが推奨される方法とタイミングについては、使用している AWS サービスのドキュメントを参照してください。

`aws:UserAgent`

このキーを使用して、リクエストのクライアントアプリケーションとポリシーで指定したアプリケーションを比較します。

- 可用性 – このキーは常にリクエストコンテキストに含まれます。
- データ型 - [文字列](#)
- 値タイプ — 単一値

#### Warning

このキーは慎重に使用する必要があります。しかし、`aws:UserAgent` 値は HTTP ヘッダー内の発信者によって渡されるため、不正な当事者が改造またはカスタムブラウザを使用して任意の `aws:UserAgent` 値を渡すことができます。そのため `aws:UserAgent` は、不正な当事者から AWS にリクエストが直接行われることを防止するために使用しないでください。このステートメントを使用して、ポリシーをテストした後にのみ、特定のクライアントアプリケーションのみを許可できます。

## その他のクロスサービス条件キー

AWS STS は、[SAML ベースのフェデレーション条件キー](#)と [OIDC フェデレーション](#)のクロスサービス条件キーをサポートします。これらのキーは、SAML を使用してフェデレーションされたユーザーが他のサービスで AWS オペレーションを実行するときに使用できます。

## IAM および AWS STS の条件コンテキストキー

JSON ポリシーの Condition 要素を使用して、すべての AWS リクエストのリクエストコンテキストに含まれるキーの値をテストできます。これらのキーは、リクエスト自体、またはリクエストが参照するリソースに関する情報を示します。ユーザーが要求したアクションを許可する前に、キーが指定された値を持っているかどうかを確認できます。これにより、JSON ポリシーステートメントが着信リクエストと照合するとき、しないときを細かく制御できます。JSON ポリシーで Condition 要素を使用する方法の詳細については、「[IAM JSON ポリシー要素Condition](#)」を参照してください。

このトピックでは、IAM サービス (iam: プレフィックス付き) および AWS Security Token Service (AWS STS) サービス (sts: プレフィックス付き) で定義および提供されているキーについて説明します。他のいくつかの AWS サービスは、そのサービスによって定義されたアクションおよびリソースに関連するサービス固有のキーも提供します。詳細については、「[AWS のサービスのアクション、リソース、および条件キー](#)」を参照してください。条件キーをサポートするサービスのドキュメントには、追加情報がある場合があります。例えば、Amazon S3 リソースのポリシーで使用できるキーについては、[Amazon Simple Storage Service ユーザーガイド](#)の「Amazon S3 ポリシーキー」を参照してください。

### トピック

- [IAM で使用可能なキー](#)
- [OIDC AWS フェデレーションで使用できるキー](#)
- [SAML ベースの AWS STS フェデレーションに利用可能なキー](#)
- [クロスサービス SAML ベースの AWS STS フェデレーションコンテキストキー](#)
- [AWS STS に利用可能なキー](#)

### IAM で使用可能なキー

IAM リソースへのアクセスを制御するポリシーで以下の条件キーを使用できます。

iam:AssociatedResourceArn

[ARN 演算子](#)で動作します。

宛先サービスでこのロールが関連付けられるリソースの ARN を指定します。通常、リソースは、プリンシパルがロールを渡すサービスに属します。場合によっては、リソースが 3 番目のサービスに属することがあります。たとえば、Amazon EC2 インスタンスで使用するロールを Amazon EC2 Auto Scaling に渡すことができます。この場合、条件は Amazon EC2 インスタンスの ARN と一致します。

この条件キーは、ポリシーの [PassRole](#) アクションにのみ適用されます。他のアクションを制限するために使用することはできません。

エンティティがロールを渡すことを許可するには、ポリシーでこの条件キーを使用しますが、そのロールが指定されたリソースに関連付けられている場合に限りです。ワイルドカード (\*) を使用すると、リージョンまたはリソース ID を制限することなく、特定の種類のリソースに対して操作を実行できます。たとえば、IAM ユーザーまたはロールが、リージョン us-east-1 または us-west-1 のインスタンスで使用する任意のロールを Amazon EC2 サービスに渡すことを許可できます。IAM ユーザーまたはロールは、ロールを他のサービスに渡すことはできません。さらに、Amazon EC2 が他のリージョンのインスタンスでロールを使用することも許可しません。

```
{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "*",
  "Condition": {
    "StringEquals": {"iam:PassedToService": "ec2.amazonaws.com"},
    "ArnLike": {
      "iam:AssociatedResourceARN": [
        "arn:aws:ec2:us-east-1:111122223333:instance/*",
        "arn:aws:ec2:us-west-1:111122223333:instance/*"
      ]
    }
  }
}
```

#### Note

[iam:PassedToService](#) をサポートする AWS サービスは、この条件キーもサポートしません。

## iam:AWSServiceName

[文字列演算子](#)で動作します。

このロールがアタッチされる AWS のサービスを指定します。

この例では、サービス名が `access-analyzer.amazonaws.com` である場合、エンティティがサービスにリンクされたロールを作成することを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": "access-analyzer.amazonaws.com"
      }
    }
  }]
}
```

## iam:FIDO-certification

[文字列演算子](#)で動作します。

FIDO セキュリティキーの登録時に MFA デバイスの FIDO 証明書レベルをチェックします。デバイス証明書は [FIDO Alliance Metadata Service \(MDS\)](#) から取得できます。FIDO セキュリティキーの証明書ステータスまたはレベルが変更されても、デバイスを登録し直して更新された証明書情報を取得しない限り、FIDO セキュリティキーが更新されることはありません。

L1、L1plus、L2、L2plus、L3、L3plus に使用できる値

この例では、セキュリティキーを登録し、デバイスの FIDO レベル1 plus 証明書を取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
```

```
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Create"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Activate",
        "iam:FIDO-certification": "L1plus"
      }
    }
  }
]
```

iam:FIDO-FIPS-140-2-certification

[文字列演算子](#)で動作します。

FIDO セキュリティキーの登録時に MFA デバイスの FIPS-140-2 検証証明書レベルをチェックします。デバイス証明書は [FIDO Alliance Metadata Service \(MDS\)](#) から取得できます。FIDO セキュリティキーの証明書ステータスまたはレベルが変更されても、デバイスを登録し直して更新された証明書情報を取得しない限り、FIDO セキュリティキーが更新されることはありません。

L1、L2、L3、L4 に使用できる値

この例では、セキュリティキーを登録し、デバイスの FIPS-140-2 レベル2 証明書を取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
```

```

        "iam:RegisterSecurityKey" : "Create"
    }
}
},
{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "iam:RegisterSecurityKey" : "Activate",
            "iam:FIDO-FIPS-140-2-certification": "L2"
        }
    }
}
]
}
}

```

### iam:FIDO-FIPS-140-3-certification

[文字列演算子](#)で動作します。

FIDO セキュリティキーの登録時に MFA デバイスの FIPS-140-3 検証証明書レベルをチェックします。デバイス証明書は [FIDO Alliance Metadata Service \(MDS\)](#) から取得できます。FIDO セキュリティキーの証明書ステータスまたはレベルが変更されても、デバイスを登録し直して更新された証明書情報を取得しない限り、FIDO セキュリティキーが更新されることはありません。

L1、L2、L3、L4 に使用できる値

この例では、セキュリティキーを登録し、デバイスの FIPS-140-3 レベル3 証明書を取得します。

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:EnableMFADevice",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:RegisterSecurityKey" : "Create"
            }
        }
    }
}

```

```
    },
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:RegisterSecurityKey" : "Activate",
          "iam:FIDO-FIPS-140-3-certification": "L3"
        }
      }
    }
  ]
}
```

### iam:RegisterSecurityKey

[文字列演算子](#)で動作します。

MFA デバイス有効化の現在の状態をチェックします。

Create または Activate に使用できる値です。

この例では、セキュリティキーを登録し、デバイスの FIPS-140-3 レベル1 証明書を取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:EnableMFADevice",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:RegisterSecurityKey" : "Create"
      }
    }
  ]
},
{
  "Effect": "Allow",
  "Action": "iam:EnableMFADevice",
  "Resource": "*",
  "Condition": {
```



```
        "StringEquals": {
            "iam:RegisterSecurityKey" : "Activate",
            "iam:FIDO-FIPS-140-3-certification": "L1"
        }
    }
}
]
```

## iam:OrganizationsPolicyId

[文字列演算子](#)で動作します。

指定された AWS Organizations IDを持つポリシーがリクエストで使用されたポリシーと一致するかどうかを確認します。この条件キーを使用する IAM ポリシーの例を表示するには、「[IAM: Organizations ポリシーのサービスの最終アクセス情報を表示](#)」を参照してください。

## iam:PassedToService

[文字列演算子](#)で動作します。

ロールを渡すことができるサービスのサービスプリンシパルを指定します。この条件キーは、ポリシーの [PassRole](#) アクションにのみ適用されます。他のアクションを制限するために使用することはできません。

ポリシーでこの条件キーを使用する場合は、サービスプリンシパルを使用してサービスを指定します。サービスプリンシパルは、ポリシーの Principal 要素で指定できるサービスの名前です。通常の形式は、SERVICE\_NAME\_URL.amazonaws.com です。

特定のサービスにのみロールを渡すことができるようにユーザーを制限するには、iam:PassedToService を使用します。たとえば、ユーザーは、代理で Amazon S3 バケットにログデータを書き込むために CloudWatch を信頼する [サービスロール](#) を作成する場合があります。次に、ユーザーは新しいサービスロールにアクセス許可ポリシーと信頼ポリシーをアタッチする必要があります。この場合、信頼ポリシーで、cloudwatch.amazonaws.com 要素の Principal を指定する必要があります。ユーザーが CloudWatch にロールを渡すことを許可するポリシーを表示するには、「[IAM: IAM ロールを特定の AWS のサービスに渡す](#)」を参照してください。

この条件キーを使用することで、ユーザーは、指定したサービスにのみ、サービスロールを作成することができます。たとえば、前述のポリシーを持つユーザーが Amazon EC2 のサービスロー

ルを作成しようとする、オペレーションは失敗します。このエラーは、ユーザーにロールを Amazon EC2 に渡すアクセス許可がないために発生します。

ロールをサービスに渡し、次に別のサービスにロールを渡すことがあります。iam:PassedToService には、ロールを引き受ける最終サービスのみが含まれ、ロールを渡す中間サービスは含まれません。

**Note**

一部のサービスでは、この条件キーをサポートしていません。

iam:PermissionsBoundary

[ARN 演算子](#)で動作します。

指定したポリシーが IAM プリンシパルリソースのアクセス許可の境界としてアタッチされていることを確認します。詳細については、「[IAM エンティティのアクセス許可境界](#)」を参照してください。

iam:PolicyARN

[ARN 演算子](#)で動作します。

管理ポリシーを含むリクエストの管理ポリシーの Amazon リソースネーム (ARN) を確認します。詳細については、「[ポリシーへのアクセスの制御](#)」を参照してください。

iam:ResourceTag/**key-name**

[文字列演算子](#)で動作します。

リソース (ユーザーまたはロール) を識別するためにアタッチされたタグが、指定されたキーの名前および値と一致するかどうかをチェックします。

**Note**

IAM と AWS STS は iam:ResourceTag IAM 条件キーと aws:ResourceTag グローバル条件キーの両方をサポートします。

カスタム属性は、キーバリューのペアの形式で IAM リソースに追加できます。IAM リソースのタグ付けの詳細については、「[the section called “IAM リソースのタグ付け”](#)」を参照してください

い。ResourceTag リソース (IAM リソースを含む) への[アクセスを制御](#)するには AWS を使用します。ただし、IAM ではグループのタグをサポートしていないため、タグを使用してグループへのアクセスを制御することはできません。

この例は、**status=terminated** タグを持つユーザーの削除を許可する ID ベースポリシーを作成する方法を示しています。このポリシーを使用するには、サンプルポリシーの#####を独自の情報に置き換えます。次に、「[ポリシーの作成](#)または[ポリシーの編集](#)」の手順に従います。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:DeleteUser",
    "Resource": "*",
    "Condition": {"StringEquals": {"iam:ResourceTag/status": "terminated"}}
  }]
}
```

## OIDC AWS フェデレーションで使用できるキー

OIDC フェデレーションを使用して、OpenID Connect 互換の ID プロバイダー (IdP) を介して認証されたユーザーに対して、AWS アカウントの IAM OpenID Connect (OIDC) ID プロバイダーへの一時的なセキュリティ認証情報を付与できます。このようなプロバイダーの例として、GitHub、Amazon Cognito、Login with Amazon、Google などがあります。自分の IdP から取得した ID トークンとアクセストークン、および Amazon Elastic Kubernetes Service ワークロードに付与された[サービスアカウントトークン](#)を使用できます。

これらの AWS OIDC 条件コンテキストキーを使用して、フェデレーティッドユーザーの特定のプロバイダー、アプリケーション、またはユーザーに関連付けられたリソースへのアクセスを制限するポリシーを作成できます。これらのキーは、通常、ロールの信頼ポリシーで使用されます。OIDC プロバイダーの名前 (token.actions.githubusercontent.com) の後にクレーム (:aud) を付けて条件キー (**token.actions.githubusercontent.com:aud**) を定義します。

一部の OIDC フェデレーション条件キーは、リソースアクセスを承認するためにロールセッションで使用できます。「セッションで使用可能」列の値が「はい」の場合、これらの条件キーをポリシーで使用して、他の AWS サービスでユーザーがアクセスできる内容を定義できます。クレームがセッションで使用できない場合、OIDC 条件コンテキストキーは、最初の [AssumeRoleWithWebIdentity](#) 認証のロール信頼ポリシーでのみ使用できます。

IdP を選択すると、IdP からのクレームが AWS の IAM 条件コンテキストキーにどのようにマッピングされるかを確認できます。

## Default

「デフォルト」には、標準の OIDC クレームと、そのクレームが AWS の AWS STS 条件コンテキストキーにどのようにマッピングされるかがリストされています。これらのキーを使用して、ロールへのアクセスを制御できます。そのためには、「AWS STS 条件キー」を「IdP JWT クレーム」列の値と比較します。IdP がタブオプションに表示されていない場合は、このマッピングを使用します。

OIDC JWT ID トークンでデフォルト実装を使用する IdPs の例としては、GitHub Actions ワークフローと Google があります。

AWS STS 条件キー	IdP JWT クレーム	セッションで使用可能
amr	amr	はい
aud	azp  azp の値が設定されていない場合、aud 条件キーは aud クレームにマッピングされます。	はい
email	email	いいえ
oaud	aud	いいえ
sub	sub	はい

GitHub での OIDC 条件コンテキストキーの使用の詳細については、「[GitHub OIDC ID プロバイダーのロールの設定](#)」を参照してください。Google aud フィールドおよび azp フィールドの詳細については、「[Google Identity Platform OpenID 接続ガイド](#)」を参照してください。

## amr

[文字列演算子](#)で動作します。このキーには複数の値が含まれるため、[条件 set 演算子](#)を使用してポリシーでキーをテストします。

例: `token.actions.githubusercontent.com:amr`

認証方法リファレンスには、ユーザーに関するログイン情報が含まれます。キーには、次の値が含まれている可能性があります。

- ユーザーが認証されていない場合、キーには `unauthenticated` のみが含まれています。
- ユーザーが認証されている場合、キーには `authenticated` という値と、呼び出しで使用されるログインプロバイダーの名前 (`accounts.google.com`) が含まれています。

`aud`

[文字列演算子](#)で動作します。

例:

- `accounts.google.com:aud`
- `token.actions.githubusercontent.com:aud`

`aud` 条件キーを使用して、オーディエンスがポリシーで指定したものと一致することを確認します。同じ ID プロバイダーに対して、`aud` キーと `sub` キーを一緒に使用することができます。

この条件キーは、以下のトークンフィールドから設定されます。

- `azp` フィールドが設定されていない場合の、アプリケーションの `aud` for OAuth 2.0 Google クライアント ID。 `azp` フィールドが設定されると、`aud` フィールドは `accounts.google.com:oauth` 条件キーと一致します。
- `azp` フィールドが設定されている場合の `azp`。これは、ウェブアプリケーションおよび Android アプリに異なる OAuth 2.0 Google クライアント ID があるが同じ Google API プロジェクトを共有している、ハイブリッドアプリで発生する可能性があります。

`accounts.google.com:aud` 条件キーを使用してポリシーを記述する場合、アプリが `azp` フィールドを設定するハイブリッドアプリであるかどうかを知る必要があります。

`azp` フィールドが設定されていません

次のサンプルポリシーは、`azp` フィールドを設定しない非ハイブリッドアプリケーションに対して機能します。この場合、Google ID トークン `aud` フィールドの値は、`accounts.google.com:aud` および `accounts.google.com:oauth` 条件キーの両方の値に一致します。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {"Federated": "accounts.google.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "accounts.google.com:aud": "aud-value",
        "accounts.google.com:oauth": "aud-value",
        "accounts.google.com:sub": "sub-value"
      }
    }
  }
]
```

azp フィールドが設定されました

次のサンプルポリシーは、azp フィールドを設定するハイブリッドアプリケーションに対して機能します。この場合、Google ID トークン aud フィールドの値は accounts.google.com:oauth 条件キーの値にのみ一致します。azp フィールドの値は accounts.google.com:aud 条件キーの値と一致します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"Federated": "accounts.google.com"},
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "accounts.google.com:aud": "azp-value",
          "accounts.google.com:oauth": "aud-value",
          "accounts.google.com:sub": "sub-value"
        }
      }
    }
  ]
}
```

## email

[文字列演算子](#)で動作します。

例: `accounts.google.com:email`

この条件キーはユーザーの E メールアドレスを検証します。このクレームの値は、このアカウントに固有のものではなく、時間の経過と共に変化する可能性があります。そのため、この値をユーザーレコードを検証するための主要な識別子として使用しないでください。

## oaud

[文字列演算子](#)で動作します。

例: `accounts.google.com:oaud`

このキーは、この ID トークンが対象としている他のオーディエンス (aud) を指定します。アプリケーションの OAuth 2.0 クライアント ID のひとつである必要があります。

## sub

[文字列演算子](#)で動作します。

例:

- `accounts.google.com:sub`
- `token.actions.githubusercontent.com:sub`

これらのキーを使用して、サブジェクトがポリシーで指定したものと一致することを確認します。sub キーは、同じ ID プロバイダーの aud キーで使用できます。

以下のロール信頼ポリシーでは、sub 条件キーによって、ロールが demo という名前の GitHub ブランチに限定されています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-repo:ref:refs/heads/demo"
        }
      }
    }
  ]
}
```

```
    ]
  }
```

## Amazon Cognito

このタブでは、Amazon Cognito が OIDC クレームを AWS 内の AWS STS 条件コンテキストキーにどのようにマッピングするかについて説明します。これらのキーを使用して、ロールへのアクセスを制御できます。そのためには、「AWS STS 条件キー」を「IdP JWT クレーム」列の値と比較します。

Amazon Cognito が使用するロールの場合、キーはクレームの後に `cognito-identity.amazonaws.com` を使用して定義されます。

アイデンティティプールクレームのマッピングの詳細については、「Amazon Cognito 開発者ガイド」の「[デフォルトのプロバイダーマッピング](#)」を参照してください。ユーザープールクレームのマッピングの詳細については、「Amazon Cognito 開発者ガイド」の「[ID トークンの使用](#)」を参照してください。

AWS STS 条件キー	IdP JWT クレーム	セッションで使用可能
amr	amr	はい
aud	aud	はい
oaud	aud	いいえ
sub	sub	はい

### amr

[文字列演算子](#)で動作します。このキーには複数の値が含まれるため、[条件 set 演算子](#)を使用してポリシーでキーをテストします。

例 - `cognito-identity.amazonaws.com:amr`

認証方法リファレンスには、ユーザーに関するログイン情報が含まれます。キーには、次の値が含まれている可能性があります。

- ユーザーが認証されていない場合、キーには `unauthenticated` のみが含まれています。



- ユーザーが認証されている場合、キーには `authenticated` という値と、呼び出しで使用されるログインプロバイダーの名前 (`cognito-identity.amazonaws.com`) が含まれています。

例えば、Amazon Cognito ロールの信頼ポリシー内の次の条件では、ユーザーが認証されていないかどうかをテストします。

```
"Condition": {
  "StringEquals":
    { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
  "ForAnyValue:StringLike":
    { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
}
```

aud

[文字列演算子](#)で動作します。

例 - `cognito-identity.amazonaws.com:aud`

ユーザーを認証したユーザープールアプリケーション。Amazon Cognito は、アクセストークン `client_id` クレームで同じ値をレンダリングします。

oaud

[文字列演算子](#)で動作します。

例 - `cognito-identity.amazonaws.com:oaud`

ユーザーを認証したユーザープールアプリケーション。Amazon Cognito は、アクセストークン `client_id` クレームで同じ値をレンダリングします。

sub

[文字列演算子](#)で動作します。

例 - `cognito-identity.amazonaws.com:sub`

認証されたユーザーの固有識別子 (UID) または件名。ユーザー名はユーザープール内で一意ではない可能性があります。sub クレームは、特定のユーザーを識別する最良の方法です。sub キーは、同じ ID プロバイダーの aud キーで使用できます。

```
{
  "Version": "2012-10-17",
```

```

    "Statement": [
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
          "cognito-identity.amazonaws.com:sub": [
            "us-east-1:12345678-1234-1234-1234-123456790ab",
            "us-east-1:98765432-1234-1234-1243-123456790ab"
          ]
        }
      }
    ]
  }
}

```

## Login with Amazon

このタブでは、Login with Amazon が OIDC クレームを AWS 内の AWS STS 条件コンテキストキーにどのようにマッピングするかについて説明します。これらのキーを使用して、ロールへのアクセスを制御できます。そのためには、「AWS STS 条件キー」を「IdP JWT クレーム」列の値と比較します。

AWS STS 条件キー	IdP JWT クレーム	セッションで使用可能
app_id	アプリケーション ID	はい
sub	ユーザー ID	はい
user_id	ユーザー ID	はい

### app\_id

[文字列演算子](#)で動作します。

例 - `www.amazon.com:app_id`

このキーは、他の ID プロバイダーが使用する aud フィールドと一致するオーディエンスコンテキストを指定します。

### sub

[文字列演算子](#)で動作します。

例 - `www.amazon.com:sub`

このキーは、ユーザー ID がポリシーで指定したものと一致することを確認します。sub キーは、同じ ID プロバイダーの aud キーで使用できます。

user\_id

[文字列演算子](#)で動作します。

例 - `www.amazon.com:user_id`

このキーは、他の ID プロバイダーが使用する aud フィールドと一致するオーディエンスコンテキストを指定します。同じ ID プロバイダーに対して、user\_id キーと id キーを一緒に使用することができます。

## Facebook

このタブでは、Facebook が OIDC クレームを AWS 内の AWS STS 条件コンテキストキーにどのようにマッピングするかについて説明します。これらのキーを使用して、ロールへのアクセスを制御できます。そのためには、「AWS STS 条件キー」を「IdP JWT クレーム」列の値と比較します。

AWS STS 条件キー	IdP JWT クレーム	セッションで使用可能
app_id	アプリケーション ID	はい
id	id	はい

app\_id

[文字列演算子](#)で動作します。

例 - `graph.facebook.com:app_id`

このキーは、オーディエンスコンテキストが、他の ID プロバイダーが使用する aud フィールドと一致することを確認します。

id

[文字列演算子](#)で動作します。

例 - `graph.facebook.com:id`

このキーは、アプリケーション (またはサイト) ID がポリシーで指定したものと一致することを確認します。

## OIDC フェデレーションに関する追加情報

- [Amazon Cognito ユーザーガイド](#)
- [OIDC フェデレーション](#)

## SAML ベースの AWS STS フェデレーションに利用可能なキー

AWS Security Token Service (AWS STS) を使用して [SAML ベースのフェデレーション](#) を操作する場合、追加の条件キーをポリシーに含めることができます。

### SAML ロールの信頼ポリシー

ロールの信頼ポリシーには、以下のキーを含めることができます。これらのキーは、ロールの引き受け許可を発信者に与えるかどうかを規定するのに役立ちます。saml:doc を除いて、すべての値が SAML アサーションから派生します。条件付きのポリシーを作成または編集すると、リスト内のすべての項目が IAM コンソールのビジュアルエディタで使用できるようになります。[] とマークされている項目は、指定された型のリストである値を持つことができます。

saml:aud

[文字列演算子](#) で動作します。

SAML アサーションの提供先のエンドポイント URL。このキーの値は、アサーションの SAML Recipient フィールドから取得されます。Audience フィールドから取得された値ではありません。

saml:commonName[]

[文字列演算子](#) で動作します。

これは commonName 属性です。

saml:cn[]

[文字列演算子](#) で動作します。

これは eduOrg 属性です。

saml:doc

[文字列演算子](#)で動作します。

これは、ロールの引き受けに使用されたプリンシパルを表します。形式は、*account-ID/provider-friendly-name* です (123456789012/SAMLProviderName など)。account-ID 値は、[SAML プロバイダー](#)を所有するアカウントを参照します。

saml:edupersonaffiliation[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonassurance[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonentitlement[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonnickname[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonorgdn

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonorgunitdn[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonprimaryaffiliation

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonprimaryorgunitdn

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonprincipalname

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersonscopedaffiliation[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:edupersontargetedid[]

[文字列演算子](#)で動作します。

これは eduPerson 属性です。

saml:eduorghomepageuri[]

[文字列演算子](#)で動作します。

これは eduOrg 属性です。

saml:eduorgidentityauthnpolicyuri[]

[文字列演算子](#)で動作します。

これは eduOrg 属性です。

saml:eduorglegalname[]

[文字列演算子](#)で動作します。

これは eduOrg 属性です。

saml:eduorgsuperioruri[]

[文字列演算子](#)で動作します。

これは eduOrg 属性です。

saml:eduorgwhitepagesuri[]

[文字列演算子](#)で動作します。

これは eduOrg 属性です。

saml:givenName[]

[文字列演算子](#)で動作します。

これは givenName 属性です。

saml:iss

[文字列演算子](#)で動作します。

発行者。URN で表されます。

saml:mail[]

[文字列演算子](#)で動作します。

これは mail 属性です。

saml:name[]

[文字列演算子](#)で動作します。

これは name 属性です。

saml:namequalifier

[文字列演算子](#)で動作します。

SAML プロバイダーのフレンドリ名に基づくハッシュ値。値は、次の値を順番に連結し、'/' 文字で区切ります。

1. Issuer 応答値 (saml:iss)
2. AWS アカウント ID
3. IAM の SAML プロバイダーのフレンドリ名 (ARN の最後の部分)

SAML プロバイダーのアカウント ID とフレンドリ名の連結は、キー saml:doc として IAM ポリシーで使用できます。詳細については、「[SAML ベースのフェデレーションでユーザーを一意に識別する](#)」を参照してください。

## saml:organizationStatus[]

[文字列演算子](#)で動作します。

これは organizationStatus 属性です。

## saml:primaryGroupSID[]

[文字列演算子](#)で動作します。

これは primaryGroupSID 属性です。

## saml:sub

[文字列演算子](#)で動作します。

\* これはクレームの件名です。組織内の個々のユーザーを一意に識別する値が含まれます ( 例: `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3` )。

## saml:sub\_type

[文字列演算子](#)で動作します。

このキーの値は、persistent が transient、または SAML アサーションで使用されている Format と Subject 要素の完全な NameID URI で構成されます。persistent の値は、saml:sub 値がセッション間のユーザーでも同じことを意味します。値が transient の場合、ユーザーの saml:sub 値はセッションごとに異なります。NameID 要素の Format 属性の詳細については、「[認証レスポンス用の SAML アサーションを設定する](#)」を参照してください。

## saml:surname[]

[文字列演算子](#)で動作します。

これは surnameuid 属性です。

## saml:uid[]

[文字列演算子](#)で動作します。

これは uid 属性です。

## saml:x500UniqueIdentifier[]

[文字列演算子](#)で動作します。



これは `x500UniqueIdentifier` 属性です。

`eduPerson` および `eduOrg` 属性に関する一般的な情報については、[REFEDS Wiki ウェブサイト](#)を参照してください。`eduPerson` 属性のリストについては、「[eduPerson オブジェクトクラス仕様 \(201602\)](#)」を参照してください。

リストタイプの条件キーには、複数の値を含めることができます。リスト値のポリシー内で条件を作成するには、[set 演算子](#) (`ForAllValues`、`ForAnyValue`) を使用できます。たとえば、所属先が「`faculty`」または「`staff`」である（ただし、「`student`」ではない）すべてのユーザーを許可するには、次のような条件を使用します。

```
"Condition": {
  "ForAllValues:StringLike": {
    "saml:edupersonaffiliation":[ "faculty", "staff" ]
  }
}
```

## クロスサービス SAML ベースの AWS STS フェデレーションコンテキストキー

SAML ベースのフェデレーション条件キーの中には、後続のリクエストで使用することで、他のサービスや `AssumeRole` 呼び出しで AWS の操作を許可できるものがあります。フェデレーテッドプリンシパルが別のロールを引き受けるときのロール信頼ポリシーや、フェデレーションプリンシパルによるリソースアクセスを許可する他の AWS サービスのリソースポリシーで使用できる条件キーは以下のとおりです。これらのキーの使用に関する詳細は、「[SAML 2.0 ベースのフェデレーションについて](#)」を参照してください。

条件キーを選択すると、説明が表示されます。

- [saml:namequalifier](#)
- [saml:sub](#)
- [saml:sub\\_type](#)

### Note

最初の外部 ID プロバイダー (IdP) 認証レスポンスの後は、他の SAML ベースのフェデレーション条件キーは使用できません。

## AWS STS に利用可能なキー

AWS Security Token Service (AWS STS) オペレーションを使用して引き受けたロールについては、IAM ロール信頼ポリシーで以下の条件キーを使用できます。

saml:sub

[文字列演算子](#)で動作します。

\* これはクレームの件名です。組織内の個々のユーザーを一意に識別する値が含まれます ( 例: `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3` )。

sts:AWSServiceName

[文字列演算子](#)で動作します。

ベアラートークンを使用できるサービスを指定するには、このキーを使用します。ポリシーでこの条件キーを使用する場合は、サービスプリンシパルを使用してサービスを指定します。サービスプリンシパルは、ポリシーの Principal 要素で指定できるサービスの名前です。例えば、`codeartifact.amazonaws.com` は AWS CodeArtifact サービスプリンシパルです。

一部の AWS のサービスでは、プログラムでリソースにアクセスする前に、AWS STS サービスベアラートークンを取得するアクセス許可が必要です。例えば、AWS CodeArtifact では、プリンシパルがベアラートークンを使用して一部のオペレーションを実行する必要があります。aws codeartifact get-authorization-token コマンドは、ベアラートークンを返します。その後、ベアラートークンを使用して AWS CodeArtifact 操作を実行できます。ベアラートークンの詳細については、「[ベアラートークンを使用する](#)」を参照してください。

可用性 – このキーは、ベアラートークンを取得するリクエストに含まれています。AWS STS を直接呼び出してベアラートークンを取得することはできません。他のサービスでいくつかのオペレーションを実行すると、自動的にベアラートークンがリクエストされます。

この条件キーを使用すると、プリンシパルが特定のサービスで使用するベアラートークンを取得することを許可できます。

sts:DurationSeconds

[桁演算子](#)で動作します。

AWS STS ベアラートークンを取得するときにプリンシパルが使用できる時間 (秒) を指定するには、このキーを使用します。

一部の AWS のサービスでは、プログラムでリソースにアクセスする前に、AWS STS サービスベアラートークンを取得するアクセス許可が必要です。例えば、AWS CodeArtifact では、プリンシパルがベアラートークンを使用して一部のオペレーションを実行する必要があります。aws codeartifact get-authorization-token コマンドは、ベアラートークンを返します。その後、ベアラートークンを使用して AWS CodeArtifact 操作を実行できます。ベアラートークンの詳細については、「[ベアラートークンを使用する](#)」を参照してください。

可用性 – このキーは、ベアラートークンを取得するリクエストに含まれています。AWS STS を直接呼び出してベアラートークンを取得することはできません。他のサービスでいくつかのオペレーションを実行すると、自動的にベアラートークンがリクエストされます。このキーは、AWS STS assume-role オペレーションには含まれていません。

sts:ExternalId

[文字列演算子](#)で動作します。

IAM ロールを引き受けるときにプリンシパルが特定の識別子を提供するように要求するには、このキーを使用します。

可用性 – このキーは、AWS CLI または AWS API を使用してロールを引き受ける際にプリンシパルが外部 ID を提供するときにリクエストに含まれています。

別のアカウントでロールを引き受ける際に必要になる場合がある一意の ID。ロールが属するアカウントの管理者から外部 ID が提供された場合は、その値を ExternalId パラメータに指定します。この値は、パスフレーズやアカウント番号などの任意の文字列とすることができます。外部 ID の最も重要な機能は、混乱した代理問題の防止と対処です。外部 ID と混乱した代理問題の詳細については、「[AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法](#)」を参照してください。

ExternalId の値は、2~1,224 文字で構成されている必要があります。この値は、空白のない英数字にする必要があります。次の記号を含めることもできます。プラス記号 (+)、等号 (=)、カンマ (,)、ピリオド (.)、アットマーク (@)、コロンの (:)、スラッシュ (/)、およびハイフン (-)。

sts:RequestContext/context-key

[文字列演算子](#)で動作します。

このキーを使用して、リクエストで渡された信頼されたトークン発行者の署名付きコンテキストアサーションに埋め込まれているセッションコンテキストのキーと値のペアを、ロールの信頼ポリシーで指定されたコンテキストキー値と比較します。

可用性 — このキーは、AWS STS AssumeRole API 操作を使用してロールを引き受けるときに、ProvidedContexts リクエストパラメータでコンテキストアサーションが提供されるときに、リクエストに含まれます。

このコンテキストキーは "sts:RequestContext/context-key":"context-value" の形式になります。ここで、context-key と context-value はコンテキストキーと値のペアです。複数のコンテキストキーがリクエストで渡された署名付きコンテキストアサーションに埋め込まれると、キーと値のペアごとに 1 つのコンテキストキーが存在します。結果のセッショントークンにプリンシパルがコンテキストキーを設定できるようにするには、ロールの信頼ポリシーで sts:SetContext アクションのアクセス権限を付与する必要があります。このキーで使用できるサポートされている IAM Identity Center コンテキストキーの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[IAM Identity Center の AWS STS 条件キー](#)」を参照してください。

ロール信頼ポリシーでこのキーを使用すると、ユーザーがロールを引き受けるときに、ユーザーまたはその属性に基づいてきめ細かいアクセス制御を実施できます。例えば、Amazon Redshift を IAM アイデンティティセンターアプリケーションとして設定して、従業員またはフェデレーテッドアイデンティティに代わって Amazon S3 リソースにアクセスできます。

次のロール信頼ポリシーにより、Amazon Redshift サービスはアカウント 111122223333 でロールを引き受けられます。また、identitystore:UserId コンテキストキー値が 1111-22-3333-44-5555 に設定されている限り、リクエストにコンテキストキーを設定するアクセス許可を Amazon Redshift サービスプリンシパルに付与します。ロールを引き継ぐと、コンテキストプロバイダがロール割り当てリクエストで設定したセッションコンテキストのキーと値のペアを含むアクティビティが AdditionalEventData 要素内の AWS CloudTrail ログに表示されます。これにより、管理者がロールを異なるプリンシパルで使用する場合に、ロールセッションを区別しやすくなります。キーと値のペアは、AWS CloudTrail または AWS STS ではなく、指定されたコンテキストプロバイダーによって設定されます。これにより、コンテキストプロバイダーは、CloudTrail ログとセッション情報に含まれるコンテキストを制御できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
```

```
        "sts:SetContext"
    ],
    "Condition": {
        "ForAllValues:ArnEquals": {
            "sts:RequestContextProviders": [
                "arn:aws:iam::aws:contextProvider/IdentityCenter"
            ]
        },
        "StringEquals": {
            "aws:SourceAccount": "111122223333",
            "sts:RequestContext/identitystore:UserId":
"1111-22-3333-44-5555"
        }
    }
}
]
```

### sts:RequestContextProviders

[ARN 演算子](#)で動作します。

このキーを使用して、リクエスト内のコンテキストプロバイダー ARN をロールの信頼ポリシーで指定されたコンテキストプロバイダー ARN と比較します。

可用性 — このキーは、AWS STS AssumeRole API 操作を使用してロールを引き受けるときに、ProvidedContexts リクエストパラメータでコンテキストアサーションが提供されるときに、リクエストに含まれます。

次の条件例では、リクエストで渡されたコンテキストプロバイダー ARN が、ロールの信頼ポリシー条件で指定された ARN と一致することを確認します。

```
"Condition": {
  "ForAllValues:ArnEquals": {
    "sts:RequestContextProviders": [
      "arn:aws:iam::aws:contextProvider/IdentityCenter"
    ]
  }
}
```

### sts:RoleSessionName

[文字列演算子](#)で動作します。

このキーを使用して、ロールを引き受けるときにプリンシパルが指定するセッション名と、ポリシーで指定されている値を比較します。

可用性 – このキーは、プリンシパルが AWS Management Console、assume-role CLI コマンド、または AWS STS AssumeRole API オペレーションを使用してロールを引き受けるときにリクエストに含まれています。

ロール信頼ポリシーでこのキーを使用すると、ユーザーがロールを引き受けるときに特定のセッション名を指定するように要求できます。たとえば、IAM ユーザーがセッション名として自分のユーザー名を指定するように要求できます。IAM ユーザーがロールを引き受けると、ユーザー名と一致するセッション名とともにアクティビティが [AWS CloudTrail ログ](#) に表示されます。これにより、管理者がロールを異なるプリンシパルで使用する場合に、ロールセッションを区別しやすくなります。

次のロール信頼ポリシーは、アカウント 111122223333 の IAM ユーザーがロールを引き受けるときに、セッション名としてユーザー名を指定することを要求します。この要件は、条件キーの `aws:username` [条件変数](#) を使用して適用されます。このポリシーにより、IAM ユーザーはポリシーがアタッチされているロールを引き受けることができます。このポリシーでは、`username` 変数が IAM ユーザーのみに存在するため、一時的な認証情報を使用するどのユーザーもロールを引き受けることはできません。

**⚠ Important**

単一値の条件キーは、[変数](#) として使用できます。複数値の条件キーは、変数としては使用できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoleTrustPolicyRequireUsernameForSessionName",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Condition": {
        "StringLike": {"sts:RoleSessionName": "${aws:username}"}
      }
    }
  ]
}
```

```
}
```

管理者がアクションの AWS CloudTrail ログを表示すると、セッション名とアカウントのユーザー名を比較できます。次の例では、matjac という名前のユーザーが MateoRole というロールを使用してオペレーションを実行しました。管理者は、matjac という名前のユーザーを持っている Mateo Jackson に連絡することができます。

```
"assumedRoleUser": {  
  "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:matjac",  
  "arn": "arn:aws:sts::111122223333:assumed-role/MateoRole/matjac"  
}
```

[ロールを使用したクロスアカウントアクセス](#)を許可した場合、あるアカウントのユーザーが別のアカウントのロールを引き受けることができます。CloudTrail に一覧表示されている引き受けたロールユーザーの ARN には、そのロールが存在するアカウントが含まれます。ロールを引き受けるユーザーのアカウントは含まれません。ユーザーはアカウント内でのみ一意です。したがって、自分が管理しているアカウントのユーザーが引き受けたロールの CloudTrail ログを確認する場合のみこの方法を使用することをお勧めします。ユーザーは、複数のアカウントで同じユーザー名を使用していることがあります。

ST: 送信元アイデンティティ

[文字列演算子](#)で動作します。

このキーを使用して、ロールを引き受けるときにプリンシパルが指定するセッション名と、ポリシーで指定されている値を比較します。

可用性 — このキーは、AWS STS assume-role CLI コマンドまたは AWS STS AssumeRole API オペレーションを使用してロールを引き受けるときに、プリンシパルがソース ID を提供するときに要求に存在します。

ロール信頼ポリシーでこのキーを使用して、ユーザーがロールを引き受けるときに特定のセッション名を設定するように要求できます。たとえば、従業員またはフェデレーション ID にソース ID の値を指定するように要求できます。ユーザー名や電子メールなど、ユーザーに関連付けられている属性の 1 つをソース ID として使用するように ID プロバイダ (IdP) を設定できます。次に、IdP は、AWS に送信するアサーションまたはクレームの属性としてソース ID を渡します。ソース ID 属性の値は、ロールを引き受けるユーザーまたはアプリケーションを識別します。

ユーザーがロールを引き受けると、設定されたソース ID 値とともにアクティビティが [AWS CloudTrail ログ](#) に表示されます。これにより、管理者は、AWS のロールで誰がどういった



アクションを実行したかを簡単に判断できます。IDがソースIDを設定できるようにするには、`sts:SetSourceIdentity` アクションのアクセス許可を付与する必要があります。

[sts:RoleSessionName](#) とは異なり、ソースIDを設定した後は、値を変更できません。これは、ソース ID によってロールで実行されたすべてのアクションのリクエストコンテキストに存在します。この値は、セッション認証情報を使用して別のロールを引き受けるときに、後続のロールセッションに保持されます。別のロールからあるロールを引き受けると、[ロールの連鎖](#)と呼ばれます。

[aws:SourceIdentity](#) グローバル条件キーを使用して、後続の要求のソースIDの値に基づいてAWS リソースへのアクセスをさらに制御できます。

次のロール信頼ポリシーにより、IAM ユーザー AdminUser はアカウント 111122223333 でロールを引き受けることができます。また、ソース ID セットが AdminUser である限り、DiegoRamirez にソースIDを設定する権限を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAdminUserAssumeRole",
      "Effect": "Allow",
      "Principal": {"AWS": " arn:aws:iam::111122223333:user/AdminUser"},
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {"sts:SourceIdentity": "DiegoRamirez"}
      }
    }
  ]
}
```

ソース ID 情報の使用の詳細については、「[引き受けたロールで実行されるアクションのモニタリングと制御](#)」を参照してください。

`sts:TransitiveTagKeys`

[文字列演算子](#)で動作します。



このキーを使用して、リクエスト内の推移的なセッションタグキーとポリシーで指定されたセッションタグキーを比較します。

可用性 – このキーは、一時的なセキュリティ認証情報を使用してリクエストを作成するときにリクエストに含まれます。これらの認証情報には、`assume-role` オペレーションまたは `GetFederationToken` オペレーションを使用して作成されたものが含まれます。

一時的なセキュリティ認証情報を使用してリクエストを行う場合、[リクエストコンテキスト](#)には `aws:PrincipalTag` コンテキストキーが含まれます。このキーには、[セッションタグ](#)、[推移的セッションタグ](#)、およびロールタグのリストが含まれます。推移的セッションタグは、セッション認証情報を使用して別のロールを引き受けるときに、後続のすべてのセッションに保持されるタグです。別のロールからあるロールを引き受けると、[ロールの連鎖](#)と呼ばれます。

この条件キーをポリシーで使用すると、ロールの引き受け時またはユーザーのフェデレーション時に、特定のセッションタグを推移的として設定するよう要求できます。

## AWS のサービスのアクション、リソース、および条件キー

それぞれの AWS のサービスでは、IAM ポリシーで使用できるように、アクション、リソース、および条件コンテキストキーを定義することができます。AWS サービス、ならびにそのアクション、リソース、および条件コンテキストキーのリストについては、「サービス認証リファレンス」の「[アクション、リソース、および条件キー](#)」を参照してください。

# IAM の詳細についてのリソース

IAM には豊富なリソースが用意されており、IAM による AWS アカウント とそのリソースの保護方法について詳細を確認するために役立ちます。

## トピック

- [ID](#)
- [認証情報 \(パスワード、アクセスキー、MFA デバイス\)](#)
- [アクセス許可とポリシー](#)
- [フェデレーションと委任](#)
- [IAM と他の AWS 製品](#)
- [セキュリティに関する一般的な慣行](#)
- [一般的なリソース](#)

## ID

ID の作成、管理、使用については、次のリソースを参照してください。

- [IAM Identity Center での ID の管理](#) – IAM Identity Center でのユーザーとグループの作成に関する手順情報。
- [IAM ID \(ユーザー、ユーザーグループ、ロール\)](#) – ユーザー、グループ、およびロールの詳細な説明。

## 認証情報 (パスワード、アクセスキー、MFA デバイス)

AWS アカウント と IAM ユーザーのパスワード、アクセスキー、および MFA デバイスの管理については、次のガイドを確認してください。

- [AWS でのユーザーパスワードの管理](#) – お客様のアカウントの IAM ユーザーのパスワードを管理する方法について説明します。
- [IAM ユーザーのアクセスキーの管理](#) – アクセスキーの仕組みと、アクセスキーを使用してプログラムによる AWS の呼び出しを行う方法について説明します。アクセスキーよりもセキュリティが高い代替手段が他にもあり、最初に検討することをお勧めします。詳細については、「AWS 全

一般のリファレンスガイド」の「[長期的なアクセスキーの考慮事項と代替方法](#)」を参照してください。

- [AWS での多要素認証 \(MFA\) の使用](#) – サインインを許可する前に、デバイスで生成されるパスワードとワンタイムコードの両方を求めるように、お客様のアカウントと IAM ユーザーを設定する方法について説明します。(これは 2 要素認証と呼ばれることがあります)。

Amazon Web Services へのアクセスに使用する認証情報の種類に関する一般的な情報については、「AWS 全般のリファレンスガイド」の「[AWS セキュリティ認証情報](#)」を参照してください。

## アクセス許可とポリシー

IAM ポリシーのしくみについて説明し、アクセス許可の最適な付与方法についてヒントを示します。

- [IAM でのポリシーとアクセス許可](#) – アクセス許可の定義に使用されるポリシー言語を紹介します。アクセス許可をユーザーまたはグループに、あるいは AWS 製品によってはリソース自体にアタッチする方法について説明します。
- [IAM JSON ポリシー要素のリファレンス](#) – ポリシー言語の各要素の説明と例を示します。
- [IAM ポリシーの検証](#) – JSON ポリシーを検証するためのリソースを検索します。
- [IAM アイデンティティベースのポリシーの例](#) – さまざまな AWS 製品の一般的なタスクに関するポリシーの例を示します。
- [AWS Policy Generator](#) – リストから製品とアクションを選択することで、カスタムポリシーを作成します。
- [IAM Policy Simulator](#) – ポリシーによって AWS への特定のリクエストが許可されるか拒否されるかをテストします。

## フェデレーションと委任

他の場所で認証された (サインインした) ユーザーに AWS アカウントのリソースへのアクセスを許可できます。アクセスを許可するのは、別の AWS アカウントの IAM ユーザー (委任と呼ばれる) であっても、自社のサインインプロセスで認証されたユーザーであってもかまいません。また、Login with Amazon、Facebook、Google などの OpenID Connect (OIDC) 互換 ID プロバイダーのようなインターネット ID プロバイダーからのユーザーであってもかまいません。これらの場合、ユーザーは AWS リソースにアクセスするための一時的なセキュリティ証明書を取得します。

- [IAM チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任](#) - 別の AWS アカウントの IAM ユーザーにクロスアカウントアクセスを許可する手順について説明します。
- [一時的な認証情報の一般的なシナリオ](#) - AWS の外部で認証されたユーザーを AWS にフェデレーションする方法について説明します。

## IAM と他の AWS 製品

ほとんどの AWS 製品は IAM と統合されているため、IAM 機能の支援によりそれらの製品でリソースへのアクセスを保護できます。以下のリソースでは、最もよく使用されているいくつかの AWS 製品に対する IAM とセキュリティについて説明します。IAM と連携するすべての製品のリスト、各製品に関する詳細情報へのリンクについては、「[IAM と連携する AWS のサービス](#)」を参照してください。

### Amazon EC2 で IAM を使用する

- [Amazon EC2 リソースへのアクセスのコントロール](#) - IAM 機能を使用して Amazon EC2 のインスタンス、ボリュームなどの管理をユーザーに許可する方法について説明します。
- [インスタンスプロファイルの使用](#) - Amazon EC2 インスタンスで実行されるアプリケーション、および他の AWS 製品へのアクセスが必要なアプリケーションのための認証情報を安全に提供するために IAM ロールを使用する方法について説明します。

### Amazon S3 での IAM の使用

- [Amazon S3 リソースに対するアクセス許可の管理](#) - Dバケットとオブジェクトに関する Amazon S3 セキュリティモデル (IAM ポリシーを含む) について説明します。
- [IAM ポリシーの記述: Amazon S3 バケット内のユーザー固有のフォルダに対するアクセス許可を付与する](#) - 内の自分のフォルダをユーザーが保護できるようにする方法について説明します。(Amazon S3 と IAM に関するその他の投稿については、ブログ投稿のタイトルで S3 タグを選択してください)。

### Amazon RDS で IAM を使用する

- [AWS Identity and Access Management\(IAM\) を使用して Amazon RDS リソースへのアクセスを管理する](#) - IAM を使用して、データベースインスタンス、データベーススナップショットなどへのアクセスを制御する方法について説明します。

- [RDS のリソースレベルのアクセス許可入門](#) – IAM を使用して特定の Amazon RDS インスタンスへのアクセスをコントロールする方法について説明します。

## Amazon DynamoDB で IAM を使用する

- [IAM を使用した DynamoDB リソースへのアクセスのコントロール](#) – IAM を使用してユーザーに DynamoDB のテーブルとインデックスの管理を許可する方法について説明します。
- 以下のビデオ (8:55) では、DynamoDB データベースの個々の項目または属性 (またはその両方) へのアクセスを制御する方法について説明します。

### [DynamoDB のきめ細かなアクセスコントロールの開始方法](#)

## セキュリティに関する一般的な慣行

AWS アカウント とリソースの保護に最適な方法についてエキスパートからのヒントやガイダンスが見つかります。

- [セキュリティ、ID、コンプライアンスに関するベストプラクティス](#) - AWS アカウント および製品全体のセキュリティを管理する方法について、セキュリティアーキテクチャの提案、IAM の使用、暗号化とデータセキュリティなども含めたリソースを検索します。
- [Identity and Access Management](#) — AWS Well-Architected フレームワークは、クラウド上でワークロードを設計および実行するための主要な概念、設計原則、およびアーキテクチャのベストプラクティスを理解するのに役立ちます。
- [IAM でのセキュリティのベストプラクティス](#) - IAM を使用して AWS アカウント とそのリソースを保護する方法についてレコメンデーションを示します。
- [AWS CloudTrail ユーザーガイド](#) – AWS CloudTrail を使用して、AWS への API コールの履歴を追跡し、その情報をログファイルに保存します。この情報は、どのユーザーおよびアカウントがお客様のアカウントのリソースにアクセスしたか、いつその呼び出しが行われたか、どのアクションがリクエストされたかなどを調べるために役立ちます。

## 一般的なリソース

IAM と AWS の詳細については、以下のリソースを参照してください。

- <https://aws.amazon.com/iam/> IAM の製品情報 – AWS Identity and Access Management 製品に関する一般的な情報。

- [AWS Identity and Access Management に関する AWS re:Post](#) – AWS re:Post にアクセスして、AWS コミュニティと IAM 関連の技術的な質問について話し合います。
- [クラスとワークショップ](#) – AWS のスキルを磨き、実践的な経験を得るために役立つセルフペースラボに加えて、ロールベースのコースと特別コースへのリンクです。
- [AWS デベロッパーセンター](#) – チュートリアルを検索、ツールのダウンロード、AWS デベロッパーイベントの確認を行います。
- [AWS デベロッパーツール](#) – AWS アプリケーションを開発および管理するためのデベロッパーツール、SDK、IDE ツールキット、およびコマンドラインツールへのリンクです。
- [ご利用開始のためのリソースセンター](#) – AWS アカウント をセットアップする方法、AWS コミュニティに参加する方法、最初のアプリケーションを起動する方法を説明します。
- [ハンズオンチュートリアル](#) – ステップ バイ ステップのチュートリアルに従って、最初のアプリケーションを AWS で起動します。
- [AWS ホワイトペーパー](#) – アーキテクチャ、セキュリティ、エコノミクスなどのトピックについて、AWS のソリューションアーキテクトや他の技術エキスパートが記述した AWS の技術ホワイトペーパーの包括的なリストへのリンクです。
- [AWS Support センター](#) – AWS Support のケースを作成して管理するためのハブです。フォーラム、技術上のよくある質問、サービスヘルスステータス、AWS Trusted Advisor など、他の役立つリソースへのリンクも含まれています。
- [AWS Support](#) – AWS Support に関する情報のメインウェブページです。クラウド内でのアプリケーションの構築および実行を支援するために 1 対 1 での迅速な対応を行うサポートチャンネルとして機能します。
- [お問い合わせ](#) – AWS の請求、アカウント、イベント、不正使用、その他の問題などに関するお問い合わせの受付窓口です。
- [AWS サイトの利用規約](#) – 当社の著作権、商標、お客様のアカウント、ライセンス、サイトへのアクセス、その他のトピックに関する詳細情報。

# HTTP クエリリクエストを使用した IAM API の呼び出し

## 内容

- [エンドポイント](#)
- [HTTPS の必要性](#)
- [IAM API リクエストのサインアップ](#)

クエリ API を使用して、プログラムで IAM および AWS STS サービスにアクセスできます。Query API リクエストは、HTTPS リクエストであり、実行すべきアクションを示す Action パラメータを含める必要があります。IAM および AWS STS は、すべてのアクションの GET リクエストおよび POST リクエストをサポートしています。つまり、API は、あるアクションに対しては GET を、他のアクションに対しては POST をといたった使い分けを必要としません。しかしながら、GET リクエストは URL のサイズに制限があります。この制限はブラウザによって異なり、通常は 2048 バイトです。したがって、大きなサイズを必要とする Query API リクエストにおいては、POST リクエストを使用する必要があります。

レスポンスは XML 文書です。応答の詳細については、[IAM API リファレンス](#) または [AWS Security Token Service API リファレンス](#) にある個別のアクションページを参照してください。

### Tip

IAM または AWS STS API オペレーションを直接呼び出す代わりに、AWS SDK のいずれかを使用することができます。AWS SDK は、さまざまなプログラム言語およびプラットフォームのライブラリやサンプルコード (Java、Ruby、.NET、iOS、Android など) から成ります。SDK は、IAM や AWS へのプログラムによるアクセス権限を作成する際に役立ちます。例えば、SDK は暗号化された署名リクエスト (下記参照)、エラー管理、リクエストの自動再試行といったタスクに対応します。AWS SDK のダウンロードやインストールなどの詳細については、「[アマゾン ウェブ サービスのツール](#)」ページを参照してください。

API のアクションやエラーの詳細については、「[IAM API リファレンス](#)」または「[AWS Security Token Service API リファレンス](#)」を参照してください。

## エンドポイント

IAM および AWS STS にはそれぞれ 1 つずつグローバルエンドポイントがあります。



- (IAM)<https://iam.amazonaws.com>
- (AWS STS)<https://sts.amazonaws.com>

#### Note

AWS STS は、グローバルエンドポイント以外に、リージョンのエンドポイントへのリクエストの送信もサポートします。リージョンの AWS STS を使用する前に、AWS アカウントのそのリージョンで STS をアクティブ化する必要があります。AWS STS のその他のリージョンのアクティブ化については、「[AWS リージョンでの AWS STS の管理](#)」を参照してください。

すべてのサービスの AWS エンドポイントとリージョンの詳細については、「AWS 全般のリファレンス」の「[サービスのエンドポイントとクォータ](#)」を参照してください。

## HTTPS の必要性

クエリ API は認証情報のような取扱いに注意が必要な情報を応答返信するため、すべての API リクエストに HTTPS を必ず使用しなければなりません。

## IAM API リクエストのサインアップ

リクエストには、アクセスキー ID およびシークレットアクセスキーによる署名が必要です。IAM での日常業務には、AWS アカウントのルートユーザー 認証情報を使用しないよう強くお勧めします。IAM ユーザーの認証情報を使用できます。また、AWS STS を使用して一時的セキュリティ認証情報を生成することもできます。

API リクエストをサインアップする際には、AWS 署名バージョン 4 の使用が推奨されます。[署名バージョン 4 の使用については、『全般のリファレンス』の「AWS 署名バージョン 4 の署名プロセス」](#)を参照してください。

署名バージョン 2 を使用する必要がある場合には、「[AWS 全般のリファレンス](#)」を参照してください。

詳細については、次を参照してください。

- [AWS セキュリティ認証情報](#)。AWS へのアクセスに使用された認証情報の種類に関する一般的情報を提供します。



- [IAM でのセキュリティのベストプラクティス](#). AWS リソースの安全を確保するため、IAM サービスを使用するにあたっての提案事項のリストを表示します。
- [IAM の一時的な認証情報](#). 一時的な認証情報の作成方法と使用方法を説明します。

# IAM のドキュメント履歴

次の表は、IAM のドキュメントの主な更新をまとめたものです。

変更	説明	日付
<a href="#">AccessAnalyzerServiceRolePolicy - アクセス許可を追加</a>	IAM Access Analyzer で、IAM ユーザーおよびロールポリシーに関する情報を取得するためのアクセス許可が、 <a href="#">AccessAnalyzerServiceRolePolicy</a> のサービスレベルのアクセス許可に追加されました。	2024 年 5 月 8 日
<a href="#">AccessAnalyzerServiceRolePolicy - アクセス許可を追加</a>	IAM Access Analyzer は、Amazon EC2 スナップショットのブロックパブリックアクセスに関する現況を取得するためのアクセス許可を、 <a href="#">AccessAnalyzerServiceRolePolicy</a> のサービスレベルのアクセス許可に追加しました。	2024 年 1 月 23 日
<a href="#">AccessAnalyzerServiceRolePolicy - アクセス許可を追加</a>	IAM Access Analyzer は、DynamoDB ストリームとテーブルを <a href="#">AccessAnalyzerServiceRolePolicy</a> のサービスレベルのアクセス許可に追加しました。	2024 年 1 月 11 日
<a href="#">AccessAnalyzerServiceRolePolicy - アクセス許可を追加</a>	IAM Access Analyzer は、Amazon S3 ディレクトリバケットを <a href="#">AccessAnalyzerServiceRolePolicy</a> のサービスレベルのアクセス許可に追加しました。	2023 年 12 月 1 日

### [IAMAccessAnalyzerReadOnlyAccess - アクセス許可を追加](#)

IAM Access Analyzer で、ポリシーの更新によって追加のアクセス権限が付与されるかどうかをユーザーが確認できるようにするためのアクセス許可が [IAMAccessAnalyzerReadOnlyAccess](#) に追加されました。

2023 年 11 月 26 日

このアクセス許可は、IAM Access Analyzer でポリシーチェックを実行するために必要です。

### [IAM Access Analyzer に未使用のアクセスアナライザーを追加](#)

IAM Access Analyzer では、使用されていないアクセス権限の検査を簡略化することで、最小特権の付与を実現します。IAM Access Analyzer は、継続的にアカウントを分析して使用されていないアクセス権限を特定し、その検出結果に基づいて一元化されたダッシュボードを作成します。

2023 年 11 月 26 日

### [IAM Access Analyzer にカスタムポリシーチェックを追加](#)

IAM Access Analyzer では、IAM ポリシーがデプロイ前にセキュリティ標準に準拠していることを検証するためのカスタムポリシーチェックが提供されるようになりました。

2023 年 11 月 26 日

### [AccessAnalyzerServiceRolePolicy - アクセス許可を追加](#)

IAM Access Analyzer で、以下のアクションをサポートするための IAM アクションが [AccessAnalyzerServiceRolePolicy](#) のサービスレベルアクセス許可に追加されました。

2023 年 11 月 26 日

- ポリシーのエンティティをリストする
- サービスの最終アクセス時間の詳細を生成する
- アクセスキー情報をリストする

### [60 を超える追加サービスおよびアクションのアクション最終アクセス情報およびポリシー生成のサポート](#)

IAM は、アクション最終アクセス情報をサポートし、60 を超える追加サービスの [アクションレベルの情報を含むポリシー](#) と、アクション最終アクセス情報が利用可能なアクションのリストを生成するようになりました。

2023 年 11 月 1 日

### [140 を超えるサービスのアクション最終アクセス情報のサポート](#)

IAM は、140 を超えるサービスのアクション最終アクセス情報と、アクション最終アクセス情報が利用可能なアクションのリストを提供するようになりました。

2023 年 9 月 14 日

[ルートユーザーと IAM ユーザー向けの、複数の多要素認証 \(MFA\) デバイスのサポート](#)

1つのユーザーに対し、最大 8 台 (FIDO セキュリティキー、仮想認証アプリケーションのソフトウェアによるタイムベースワンタイムパスワード (TOTP)、ハードウェア TOTP トークンなど) の MFA デバイスを追加できるようになりました。

2022 年 11 月 16 日

[新しいリソースタイプに対する IAM Access Analyzer のサポート](#)

IAM Access Analyzer は、以下のリソースタイプに対するサポートを追加しました。

2022 年 10 月 25 日

- Amazon EBS ボリュームスナップショット
- Amazon ECR リポジトリ
- Amazon EFS ファイルシステム
- Amazon RDS DB スナップショット
- Amazon RDS DB クラスタースナップショット
- Amazon SNS トピック

[U2F の非推奨および WebAuthn/FIDO の更新](#)

MFA オプションとしての U2F についての記述を削除し、WebAuthn、FIDO2、および FIDO セキュリティキーに関する情報を追加しました。

2022 年 5 月 31 日

[IAM のレジリエンスに関する更新](#)

イベントによって AWS リージョン間の通信が中断されたときに IAM 認証情報へのアクセスを維持するための情報を追加しました。

2022 年 5 月 16 日

## [リソース用の新しいグローバル条件キー](#)

リソースを含む AWS Organizations のアカウント、組織単位 (OU)、または組織に基づいて、リソースへのアクセスを制御できるようになりました。IAM ポリシーでは、`aws:ResourceAccount`、`aws:ResourceOrgID` および `aws:ResourceOrgPaths` のグローバル条件キーを使用することができます。

2022 年 4 月 27 日

## [AWS SDK を使用した IAM のコード例](#)

AWS Software Development Kit (SDK) で IAM を使用方法を示すコード例を追加しました。例は、個々のサービス関数を呼び出す方法を示すコード抜粋と、同じサービス内で複数の関数を呼び出して特定のタスクを達成する方法を示す例に分けられています。

2022 年 4 月 7 日

## [ポリシー評価ロジックフローチャートの更新](#)

ポリシー評価ロジックフローチャートおよび [アカウント内でのリクエストの許可または拒否の決定](#) セクションの関連テキストの更新です。

2021 年 11 月 17 日

## [セキュリティベストプラクティスの更新](#)

ルートユーザーの認証情報を使用する代わりに管理者ユーザーを作成することについての情報を追加しました。また、ユーザーグループを使用して IAM ユーザーにアクセス許可を割り当てるベストプラクティスを削除しました。さらに、インラインポリシーの代わりに管理ポリシーを使用するタイミングを明確にしました。

2021 年 10 月 5 日

## [リソースベースのポリシーについての、ポリシー評価ロジックのトピックの更新](#)

リソースベースのポリシーの影響についての情報と、同じアカウントに存在する異なるプリンシパルタイプについての情報を追加しました。

2021 年 10 月 5 日

## [単一値および複数値の条件キーの更新](#)

単一値条件キーと複数値条件キーの違いについて、詳しく説明します。[AWS グローバル条件コンテキストキー](#)に、それぞれの値タイプが追加されました。

2021 年 9 月 30 日

## [IAM Access Analyzer は、Amazon S3 マルチリージョンアクセスポイントをサポートします](#)

IAM Access Analyzer は、Amazon S3 [マルチリージョンアクセスポイント](#)を使用するバケットを含め、パブリックおよびクロスアカウントアクセスを許可する Amazon S3 バケットを識別します。。

2021 年 9 月 2 日

[AWS管理ポリシーの更新-既存のポリシーへの更新](#)

IAM Access Analyzer は、既存のAWS管理ポリシーを更新しました。

2021 年 9 月 2 日

[アクションレベルのポリシー生成でサポートされるその他のサービス](#)

IAM Access Analyzer は、AWS の追加サービスのアクションレベルのアクセスアクティビティ情報を含む追加の IAM ポリシーを生成できません。

2021 年 8 月 24 日

[クロスアカウントの証跡のIAM ポリシーの生成](#)

IAM Access Analyzer を使用して、別のアカウントでの AWS CloudTrail 証跡、例えば、集中型 AWS Organizations 証跡を使用したアクセスアクティビティに基づいてきめ細かなポリシーを生成できるようになりました。

2021 年 8 月 18 日



## IAM Access Analyzer のポリシーチェック

IAM Access Analyzer は、IAM ポリシーに含まれる条件を検証する新しいポリシーチェックを追加することで、ポリシーの検証を拡張しました。これらのチェックは、ポリシーステートメント内の条件ブロックを分析し、セキュリティの警告、エラー、および提案を実行可能な推奨事項とともに報告します。

2021 年 6 月 29 日

IAM アクセサナライザでは、次のポリシーチェックが追加されました。

- [エラー — サービスプリンシパルの形式が無効です](#)
- [エラー — 条件にタグキーがありません](#)
- [セキュリティ警告 — サービスに対してサポートされていないタグ条件キーで NotAction を拒否する](#)
- [セキュリティ警告 — サービスに対してサポートされていないタグ条件キーで拒否](#)
- [セキュリティ警告 — ペアリングされた条件キーがありません](#)
- [提案 — サービス用にサポートされていないタグ条件キーで NotAction を許可する](#)

- [提案 — サービスに対してサポートされていないタグ条件キーで許可する](#)

[アクション最終アクセスによるより多くのサービスのサポート](#)

IAM プリンシパルが Amazon EC2、IAM、Lambda、および Amazon S3 管理アクションに対してアクションを前回使用した時点について、IAM コンソールでアクションの最後にアクセスした情報を表示できるようになりました。また、AWS CLI または AWS API を使用して、データレポートを取得することもできます。この情報を使用して不要なアクセス許可を識別し、最小権限の原則により良く準拠するよう IAM ポリシーを改善することができます。

2021 年 4 月 19 日

[引き受けたロールで実行されるアクションのモニタリングとコントロールアクション](#)

管理者は、ID が AWS CloudTrail にログインしているソース ID を渡すように要求するように IAM ロールを設定できます。ソース ID 情報を確認すると、管理者は、引き受けたロールセッションでアクションを実行したユーザーやアクションを判断できます。

2021 年 4 月 13 日

[アクセスアクティビティに基づいて IAM ポリシーを生成する](#)

IAM Access Analyzer を使用して、AWS CloudTrail で見つかったアクセスアクティビティに基づいてきめ細かいポリシーを生成できるようになりました。

2021 年 4 月 7 日

### [IAM Access Analyzer のポリシーチェック](#)

IAM Access Analyzer は、ポリシー作成中に 100 を超えるポリシーチェックと、実用的な推奨事項を提供するようになりました。

2021 年 3 月 16 日

### [拡張されたポリシー検証オプション](#)

拡張されたポリシー検証は、IAM コンソールで利用できます。AWS API、および AWS CLI IAM Access Analyzer のポリシーチェックを使用して、安全で機能的な JSON ポリシーを作成できるようにします。

2021 年 3 月 15 日

### [IAM リソースのタグ付け](#)

タグキーおよび値のペアを使用して、追加の IAM リソースにタグを付けることができるようになりました。

2021 年 2 月 11 日

### [IAM ユーザーのデフォルトのパスワードポリシー](#)

AWS アカウント にカスタムパスワードポリシーを設定しない場合、IAM ユーザーのパスワードはデフォルトの AWS パスワードポリシーの要件を満たす必要があります。

2020 年 11 月 18 日

### [AWS サービスのアクション、リソース、および条件キーの各ページが移動しました](#)

それぞれの AWS のサービスでは、IAM ポリシーで使用できるように、アクション、リソース、および条件コンテキストキーを定義することができます。これで、サービス認証リファレンスで、AWS サービス、ならびにそのアクション、リソース、および条件コンテキストキーのリストを検索できるようになりました。

2020 年 11 月 16 日

### [より長い IAM ユーザーロールセッション期間](#)

IAM ユーザーは、AWS Management Console でロールを切り替えるときにロールセッション時間を長くできるようになりました。これによりセッションの期限切れによる中断を減らすことができます。ユーザーには、ロールに設定された最大セッション期間、または IAM ユーザーのセッションの残り時間のいずれか短い方が付与されます。

2020 年 7 月 24 日

### [Service Quotas を使用して IAM エンティティのクイック増加をリクエストする](#)

Service Quotas コンソールを使用して、調整可能なクォータの IAM クォータ増加をリクエストできます。現在、いくつかの増加は Service Quotas で自動的に承認され、数分以内にアカウントで利用できるようになります。より大きなリクエストは AWS Support に送信されます。

2020 年 6 月 25 日

## [IAM の最終アクセス時間情報に Amazon S3 管理アクションを追加](#)

サービスの最終アクセス時間情報に加えて、プリンシパルが最後に Amazon S3 アクションを使用した時間に関する情報を IAM コンソールに表示できるようになりました。また、AWS CLI または AWS API を使用して、データレポートを取得することもできます。このレポートには、プリンシパルで許可されているどのサービスとアクションに、いつ最後にアクセスが試みられたかに関する情報が含まれます。この情報を使用して不要なアクセス許可を識別し、最小権限の原則により良く準拠するよう IAM ポリシーを改善することができます。

2020 年 6 月 3 日

## [セキュリティチャプター追加](#)

セキュリティに関する章では、セキュリティとコンプライアンスの目標を達成するために IAM と AWS STS を設定する方法について説明します。また、IAM リソースのモニタリングや保護に役立つ、他の AWS のサービスの使用方法についても説明します。

2020 年 4 月 29 日

## [sts:RoleSessionName](#)

ロールを引き受けるときにプリンシパルが指定するセッション名に基づいてアクセス許可を付与するポリシーを記述できるようになりました。

2020 年 4 月 21 日

## AWS サインインページの更新

メインの AWS サインインページでサインインする場合、AWS アカウントのルートユーザー または IAM ユーザーとしてサインインすることを選択できません。この場合、ページのラベルに、ルートユーザーの E メールアドレス または IAM ユーザー情報を指定する必要があるかどうかが表示されます。このドキュメントには、AWS サインインページを理解するのに役立つ、更新済みの画面キャプチャが含まれています。

2020 年 3 月 4 日

[aws:ViaAWSService および  
aws:CalledVia 条件キー](#)

サービスが IAM プリンシパル (ユーザーまたはロール) に代わってリクエストを実行できるかどうかを制限するポリシーを記述できるようになりました。プリンシパルが AWS サービスに対してリクエストを実行すると、そのサービスはプリンシパルの認証情報を使用して、後続のリクエストを他のサービスに対して実行することがあります。いずれかのサービスがプリンシパルの認証情報を使用してリクエストを実行したときに一致する、aws:ViaAWSService 条件キーを使用します。特定のサービスがプリンシパルの認証情報を使用してリクエストを実行したときに一致する、aws:CalledVia 条件キーを使用します。

2020 年 2 月 20 日

[Policy Simulator でアクセス許可の境界のサポートを追加](#)

IAM Policy Simulator を使用して、IAM エンティティに対するアクセス許可の境界の影響をテストできるようになりました。

2020 年 1 月 23 日

## クロスアカウントポリシーの評価

AWS でクロスアカウントアクセスのポリシーが評価される方法を学習できるようになりました。これは、別のアカウントのプリンシパルがリソースにアクセスすることを信頼するアカウントのリソースに許可する、リソースベースのポリシーが含まれている場合に発生します。リクエストは両方のアカウントで許可されている必要があります。

2020 年 1 月 2 日

## セッションタグ

AWS STS でロールを受け取るとき、またはユーザーをフェデレートするときにタグを含めることができるようになりました。AssumeRole または GetFederationToken オペレーションを実行するときに、セッションタグを属性として渡すことができます。AssumeRoleWithSAML または AssumeRoleWithWebIdentity オペレーションを実行するときに、企業の ID から AWS に属性を渡すことができます。

2019 年 11 月 22 日



## [AWS Organizations の AWS アカウントのグループへのアクセスを制御する](#)

IAM ポリシーで AWS Organizations からレファレンス組織単位 (OU) を参照できるようになりました。Organizations を使用してアカウントを OU に編成する場合、リソースへのアクセスを許可する前に、プリンシパルが特定の OU に属するように要求できます。プリンシパルには、AWS アカウントのルートユーザー、IAM ユーザー、IAM ロールが含まれます。これを行うには、ポリシーの `aws:PrincipalOrgPaths` 条件キーに OU パスを指定します。

2019 年 11 月 20 日

## [最後に使用したロール](#)

ロールが最後に使用された日付、時刻、およびリージョンを表示できるようになりました。この情報は、アカウント内の未使用のロールを特定する際にも役立ちます。AWS Management Console、AWS CLI および AWS API を使用して、ロールが最後にいつ使用されたかに関する情報を表示できます。

2019 年 11 月 19 日

## [グローバル条件コンテキスト キーページの更新](#)

各グローバル条件キーがリクエストのコンテキストに含まれるタイミングを知ることができるようになりました。また、ページの目次 (TOC) を使用して、各キーにさらに簡単に移動することもできます。ページの情報は、より正確なポリシーを記述するのに役立ちます。例えば、従業員が IAM ロールでフェデレーションを使用する場合、aws:userName キーではなく aws:userId キーを使用する必要があります。aws:userName キーは IAM ユーザーにのみ適用され、ロールには適用されません。

2019 年 10 月 6 日

## [AWS の ABAC](#)

タグを使用した AWS での属性ベースのアクセスコントロール (ABAC) の動作と、従来の AWS 認証モデルとの比較について説明します。ABAC チュートリアルを使用して、プリンシパルタグを持つ IAM ロールが一致するタグを持つリソースにアクセスすることを許可するポリシーを作成およびテストする方法を学習します。この戦略により、個人は自分のジョブに必要な AWS リソースのみを表示または編集できます。

2019 年 10 月 3 日

## [AWS STS GetAccessKeyInfo オペレーション](#)

コード内の AWS アクセスキーを確認して、キーが所有するアカウントのものであるかどうかを判断できます。アクセスキー ID は、[aws sts get-access-key-info](#) AWS CLI コマンドまたは [GetAccessKeyInfo](#) AWS API オペレーションを使用して渡すことができます。

2019 年 7 月 24 日

## [IAM での Organizations サービスの最終アクセス時間情報の表示](#)

IAM コンソールの AWS Organizations セクションで、AWS Organizations エンティティまたはポリシーのサービスの最終アクセス時間情報を表示できます。また、AWS CLI または AWS API を使用して、データレポートを取得することもできます。このデータには、Organizations アカウントのプリンシパルで許可されているどのサービスがいつ最後にアクセスを試みたかに関する情報が含まれます。この情報を使用して不要なアクセス許可を識別し、最小権限の原則により良く準拠するよう Organizations ポリシーを改善することができます。

2019年6月20日

### [管理ポリシーをセッションポリシーとして使用する](#)

ロールを引き受ける際に最大 10 の管理ポリシー ARN を渡すことができるようになりました。これにより、ロールの一時的な認証情報のアクセス許可を制限することができます。

2019 年 5 月 7 日

### [グローバルエンドポイントに対するセッショントークンの AWS STS リージョンの互換性](#)

バージョン 1 またはバージョン 2 グローバルエンドポイントのトークンの使用を選択できるようになりました。バージョン 1 トークンは、デフォルトで利用できる AWS リージョンでのみ有効です。これらのトークンは、アジアパシフィック (香港) など、手動で有効になっているリージョンでは動作しません。バージョン 2 のトークンはすべてのリージョンで有効です。ただし、バージョン 2 トークンの方が長く、一時的にトークンを保存するシステムに影響する可能性があります。

2019 年 4 月 26 日

### [AWS リージョンの有効化と無効化の許可](#)

管理者がアジアパシフィック (香港) リージョン (ap-east-1) を有効化および無効化できるようにするポリシーを作成できるようになりました。

2019 年 4 月 24 日

## [IAM ユーザーのセキュリティ 認証情報ページ](#)

IAM ユーザーが [My Security Credentials (セキュリティ認証情報)] ページで自分の認証情報を管理できるようになりました。この AWS Management Console ページには、アカウント ID や正規ユーザー ID などのアカウント情報が表示されます。ユーザーは、自分のパスワード、アクセスキー、X.509 証明書、SSH キー、および Git 認証情報を表示および編集することもできます。

2019 年 1 月 24 日

## [アクセスアドバイザー API](#)

AWS CLI および AWS API を使用して、サービスの最終アクセス時間情報を表示できるようになりました。

2018 年 12 月 7 日

## [IAM ユーザーとロールのタグ付け](#)

タグキーと値のペアを使用して、IAM タグを使い、アイデンティティ (IAM ユーザーまたはロール) にカスタム属性を追加できるようになりました。タグを使用して、アイデンティティのリソースへのアクセスや、アイデンティティにアタッチできるタグを制御することもできます。

2018 年 11 月 14 日

## [U2F セキュリティキー](#)

AWS Management Console にサインインするとき、U2F セキュリティキーを多要素認証 (MFA) オプションとして使用できるようになりました。

2018 年 9 月 25 日

<a href="#">Amazon VPC エンドポイントのサポート</a>	米国西部 (オレゴン) リージョンで、VPC と AWS STS の間のプライベート接続を確立できるようになりました。	2018 年 7 月 31 日
<a href="#">アクセス許可の境界</a>	新機能では、完全な IAM; 管理アクセスを付与することなく、信頼された従業員に IAM 許可を管理できる権限を付与することがより簡単になりました。	2018 年 7 月 12 日
<a href="#">aws:PrincipalOrgID</a>	新しい条件キーでは、IAM プリンシパルの AWS 組織を指定することで、AWS リソースへのアクセスをより簡単にコントロールできます。	2018 年 5 月 17 日
<a href="#">aws:RequestedRegion</a>	新しい条件キーでは、より簡単に IAM ポリシーを使用して AWS リージョンへのアクセスをコントロールできます。	2018 年 4 月 25 日
<a href="#">IAM ロールのセッションの有効期間を増やしました</a>	IAM ロールのセッションの有効期間は 12 時間になりました。	2018 年 3 月 28 日
<a href="#">ロール作成ワークフローを更新しました</a>	新しいワークフローでは、信頼関係を作成するプロセスとロールにアクセス許可をアタッチするプロセスが効率化されます。	2017 年 9 月 8 日

<a href="#">AWS アカウント サインイン プロセス</a>	更新された AWS サインイン エクスペリエンスにより、ルートユーザーと IAM ユーザーの両方が、AWS Management Console のホームページから [Sign In to the Console] (コンソールにサインイン) リンクを使用できるようになりました。	2017 年 8 月 25 日
<a href="#">IAM ポリシーの例</a>	ドキュメントの更新に伴って 30 を超えるポリシー例が追加されました。	2017 年 8 月 2 日
<a href="#">IAM のベストプラクティス</a>	IAM コンソールの [ユーザー] セクションに追加された情報により、IAM のベストプラクティスを実践しやすくなりました。	2017 年 7 月 5 日
<a href="#">Auto Scaling のリソース</a>	リソースレベルのアクセス許可で、Auto Scaling のリソースへのアクセスとアクセス許可をコントロールできます。	2017 年 5 月 16 日
<a href="#">Amazon RDS for MySQL および Amazon Aurora データベース</a>	データベース管理者は、データベースユーザーと IAM ユーザーおよびロールに関連付けることができます。これにより、すべての AWS リソースへのユーザーアクセスを一元管理できます。	2017 年 4 月 24 日
<a href="#">サービスリンクロール</a>	サービスにリンクされたロールでは、より簡単で安全な方法でアクセス許可を AWS のサービスに委任できます。	2017 年 4 月 19 日

## [ポリシー概要](#)

新しいポリシー概要では、IAM ポリシーのアクセス許可をより簡単に理解できます。

2017 年 3 月 23 日