aws

開発者ガイド

Amazon Cognito

Amazon Cognito: 開発者ガイド

Table of Contents

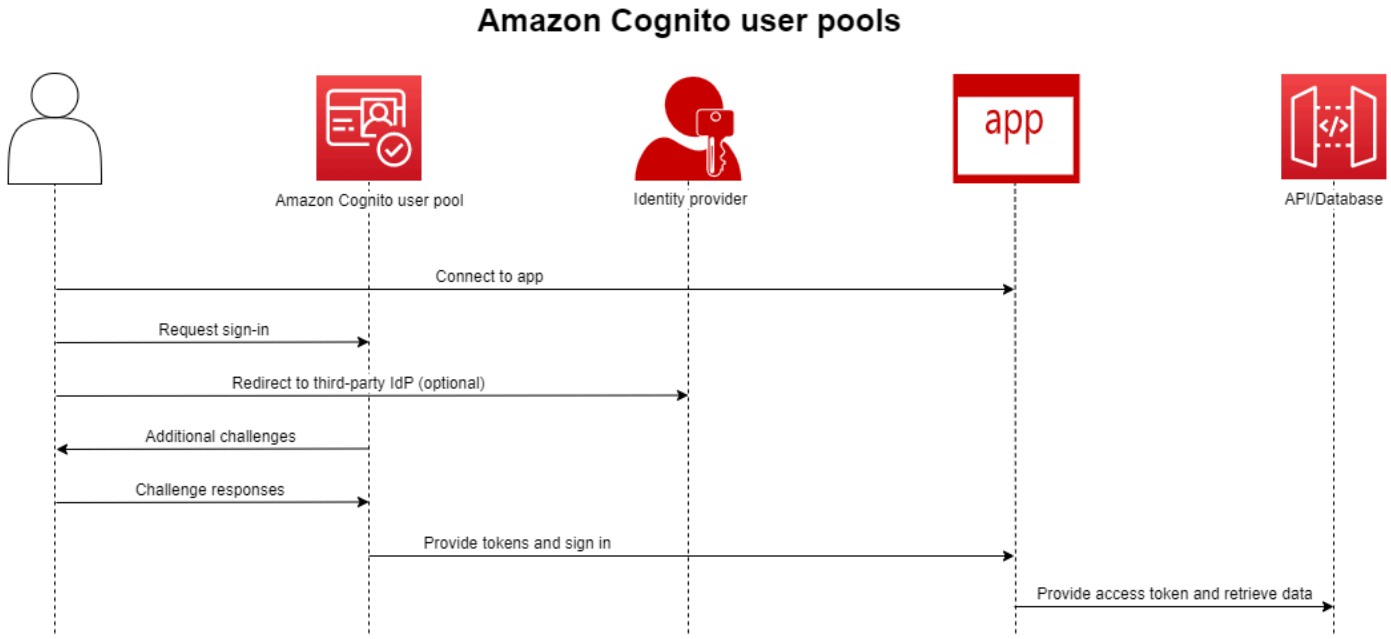
Amazon Cognito とは

Amazon Cognito はウェブアプリとモバイルアプリ用のアイデンティティプラットフォームです。これは、ユーザーディレクトリ、認証サーバー、OAuth 2.0 アクセストークンと AWS 認証情報の承認サービスです。Amazon Cognito を使用すると、組み込みのユーザーディレクトリ、エンタープライズディレクトリ、Google や Facebook などのコンシューマー ID プロバイダーからユーザーを認証および承認できます。

トピック

その後に続く 2 つのコンポーネントが Amazon Cognito を構成します。これらは、ユーザーのアクセスニーズに応じて、独立して動作することも、連携して動作することもできます。

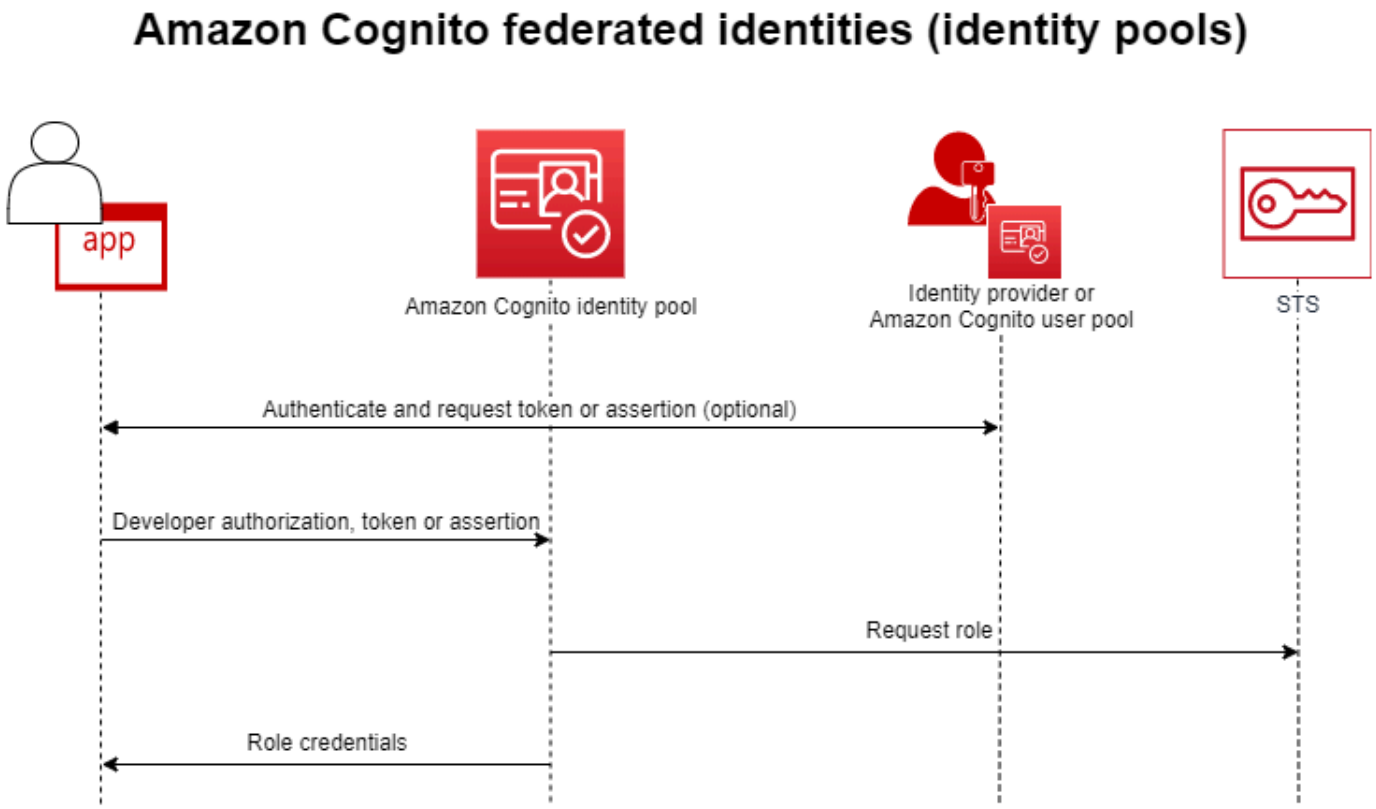
ユーザープール



アプリまたは API のユーザーを認証および承認する場合は、ユーザープールを作成します。ユーザープールは、セルフサービスと管理者主導の両方によるユーザー作成、管理、認証を行うユーザーディレクトリです。ユーザープールは、独立したディレクトリや OIDC ID プロバイダー (IdP) でも、従業員や顧客のアイデンティティを提供するサードパーティプロバイダーの中間サービスプロバイダー (SP) でもかまいません。SAML 2.0 およびユーザープール IdPs を持つ OIDC の組織のワークフォース ID に対して、アプリケーションで Single Sign-On (SSO) を提供できます。また、パブリック OAuth 2.0 アイデンティティストア (Amazon、Google、Apple、および Facebook) 内の組織のカスタマーアイデンティティに対して、アプリで SSO を提供することもできます。カスタマー ID およびアクセス管理 (CIAM) の詳細については、「CIAM とは何ですか?」を参照してください。

ユーザープールはアイデンティティプールとの統合を必要としません。ユーザープールから、認証された JSON ウェブトークン (JWT) をアプリ、ウェブサーバー、または API に直接発行できます。

アイデンティティプール



認証されたユーザーまたは匿名ユーザーが AWS リソースにアクセスすることを許可する場合は、Amazon Cognito ID プールを設定します。ID プールは、ユーザーにリソースを提供するためにアプリの AWS 認証情報を発行します。ユーザープールや SAML 2.0 サービスなど、信頼できる ID プロバイダーでユーザーを認証できます。また、オプションでゲストユーザーに認証情報を発行することもできます。ID プールは、ロールベースのアクセス制御と属性ベースのアクセス制御の両方を使用して、 AWS リソースへのアクセスに対するユーザーの認可を管理します。

アイデンティティプールはユーザープールとの統合を必要としません。アイデンティティプールは、ワークフォースとコンシューマーの両方の ID プロバイダーからの認証済みクレームを直接受け入れることができます。

Amazon Cognito のユーザープールとアイデンティティプールの併用

このトピックの最初の図では、Amazon Cognito を使用してユーザーを認証し、ユーザーにアクセス権を付与します。 AWS のサービス

1. アプリユーザーはユーザープールを介してサインインし、OAuth 2.0 トークンを受け取ります。

2. アプリケーションは、ユーザープールトークンを ID プールと交換して、 AWS APIs および AWS Command Line Interface () で使用できる一時的な AWS 認証情報と交換しますAWS CLI。
3. アプリケーションは認証情報セッションをユーザーに割り当て、Amazon S3 や Amazon DynamoDB AWS のサービス などの に認可されたアクセスを提供します。

アイデンティティプールとユーザープールを使用するその他の例については、「Amazon Cognito の一般的なシナリオ」を参照してください。

Amazon Cognito では、責任分担モデルによるクラウド義務のセキュリティは、SOC 1～3、PCI DSS、ISO 27001 に準拠し、HIPAA-BAA に対応しています。Amazon Cognito では、SOC1～3、ISO 27001、HIPAA-BAA に準拠し、PCI DSS に準拠し、PCI DSS に準拠し、Amazon Cognito でクラウド内のセキュリティを設計できます。詳細については、[AWS 対象範囲内のサービス]を参照してください。リージョンデータに関する考慮事項も参照してください。

Amazon Cognito の機能

ユーザープール

Amazon Cognito ユーザープールは、ユーザーディレクトリです。ユーザープールを使用することで、ユーザーはウェブまたはモバイルアプリに Amazon Cognito 経由でサインインする、またはサードパーティー IdP 経由でフェデレートすることができます。フェデレーションとローカルユーザーには、ユーザープールにユーザープロファイルがあります。

ローカルユーザーとは、ユーザーがユーザープールにサインアップした、または直接作成したユーザーです。これらのユーザープロファイルは、、 AWS SDK AWS Management Console、または AWS Command Line Interface () で管理およびカスタマイズできますAWS CLI。

Amazon Cognito ユーザープールは、サードパーティーの からトークンとアサーションを受け取り IdPs、アプリケーションに発行する JWT にユーザー属性を収集します。Amazon Cognito が とのやり取りを処理し、クレームを中央トークン形式にマッピングしながら IdPs、1 つの JWTs セットでアプリケーションを標準化できます。

Amazon Cognito ユーザープールは、スタンドアロンIdP である可能性があります。Amazon Cognito は OpenID Connect (OIDC) 標準に基づいて、認証と承認のための JWT を生成しています。ローカルユーザーをログインさせると、ユーザープールがそれらのユーザーに対して権限を持つことになります。ローカルユーザーを認証すると、次の機能にアクセスできます。

- Amazon Cognito ユーザープール API を呼び出してユーザーを認証、承認、管理する独自のウェブフロントエンドを実装します。
- ユーザーに多要素認証 (MFA) を設定します。Amazon Cognito は、タイムベースドワンタイムパスワード (TOTP) と SMS メッセージ MFA をサポートしています。
- 悪意のある管理下にあるユーザーアカウントからのアクセスから保護します。
- 独自のカスタムマルチステップ認証フローを作成します。
- 別のディレクトリでユーザーを検索し、Amazon Cognito に移行します。

Amazon Cognito ユーザープールは、 へのサービスプロバイダー (SP) としてのデュアルロールと IdPs、アプリへの IdP を果たすこともできます。Amazon Cognito ユーザープールは、Facebook や Google IdPs などのコンシューマー、または Okta や Active Directory フェデレーションサービス (ADFS) IdPs などのワークフォースに接続できます。

Amazon Cognito ユーザープールが発行する OAuth 2.0 トークンと OpenID Connect (OIDC) トークンを使用すると、次のことができるようになります。

- ユーザーを認証し、ユーザープロファイルの設定に必要な情報を提供する ID トークンをアプリ内で受け入れる
- ユーザーの API 呼び出しを許可する OIDC スコープを使用して、API でアクセストークンを受け入れます。
- Amazon Cognito ID プールから AWS 認証情報を取得します。

Amazon Cognito ユーザープールの機能

機能	説明
OIDC IdP	ユーザーを認証するための ID トークンを発行する
認証サーバー	API へのユーザーアクセスを許可するアクセス APIs
SAML 2.0 SP	SAML アサーションを ID トークンとアクセストークンに変換する
OIDC SP	OIDC トークンを ID トークンとアクセストークンに変換する

OAuth 2.0 SP	Apple、Facebook、Amazon、Google の ID トークンを独自の ID トークンとアクセストークンに変換する
認証フロントエンドサービス	ホストされた UI でユーザーをサインアップ、管理、認証する
独自の UI の API サポート	サポートされている AWS SDKs 1 で API リクエストを使用してユーザーを作成、管理、認証する
MFA	追加の認証要素として SMS メッセージ、TOTPs、またはユーザーのデバイスを使用する1
セキュリティのモニタリングと対応	悪意のあるアクティビティや安全でないパスワードからの保護1
認証フローをカスタマイズする	独自の認証メカニズムを構築するか、既存のフローにカスタムステップを追加する1
グループ	トークンを ID プールに渡すときに、ユーザーの論理グループと IAM ロールクレームの階層を作成する
ID トークンをカスタマイズする	新規、変更、抑制されたクレームで ID トークンをカスタマイズする
ユーザー属性をカスタマイズする	ユーザー属性に値を割り当て、独自のカスタム属性を追加する

[1] この機能はローカルユーザーのみが使用できます。

ユーザープールの詳細については、「ユーザープールの開始方法」とAmazon Cognito ユーザープール API リファレンスを参照してください。

アイデンティティプール

ID プールは、ユーザーまたはゲストに割り当て、一時的な AWS 認証情報の受信を許可する一意の識別子または ID のコレクションです。SAML 2.0、OpenID Connect (OIDC)、または OAuth 2.0 ソー

シャル ID プロバイダー (IdP) からの信頼できるクレームという形でアイデンティティプールに認証証明書を提示すると、ユーザーをアイデンティティプールの ID に関連付けます。ID プールが ID 用に作成するトークンは、 AWS Security Token Service () から一時的なセッション認証情報を取得できますAWS STS。

認証された ID を補完するために、IdP 認証なしで AWS アクセスを許可するように ID プールを設定することもできます。独自の認証証明を提供することも、認証なしで提供することもできます。認証されていない ID を使用して、リクエストしたすべてのアプリユーザーに一時的な AWS 認証情報を付与できます。アイデンティティプールでは、開発者が認証したアイデンティティを使用して、独自のカスタムスキーマに基づいてクレームを受け付けて認証情報を発行することもできます。

Amazon Cognito アイデンティティプールでは、 AWS アカウント 内の IAM ポリシーと統合する方法が 2 つあります。これら 2 つの機能は一緒に使用することも、個別に使用することもできます。

ロールベースアクセスコントロール

ユーザーがアイデンティティプールにクレームを渡すと、Amazon Cognito はリクエストする IAM ロールを選択します。ロールの権限をニーズに合わせてカスタマイズするには、各ロールに IAM ポリシーを適用します。たとえば、ユーザーがマーケティング部門に所属していることを証明すると、マーケティング部門のアクセスニーズに合わせたポリシーが設定されたロールの認証情報を受け取ります。Amazon Cognito では、デフォルトロール、ユーザーのクレームをクエリするルールに基づくロール、またはユーザープール内のユーザーのグループメンバーシップに基づくロールをリクエストできます。IAM がユーザーのアイデンティティプールのみを信頼して一時的なセッションを生成するようにロールの信頼ポリシーを設定することもできます。

アクセスコントロールの属性

アイデンティティプールはユーザーのクレームから属性を読み取り、ユーザーの一時セッションのプリンシパルタグにマッピングします。次に、アイデンティティプールからセッションタグを伝達する IAM プリンシパルに基づいてリソースへのアクセスを許可または拒否するように IAM リソースベースのポリシーを設定できます。例えば、ユーザーがマーケティング部門にいることを実証した場合、 はセッション に AWS STS タグを付けますDepartment: marketing。Amazon S3 バケットは、Departmentタグmarketingに の値を必要とする aws:PrincipalTag 条件に基づいて読み取りオペレーションを許可します。

Amazon Cognito アイデンティティプールの機能

機能	説明

Amazon Cognito ユーザープール SP	ユーザープールの ID トークンを のウェブ ID 認証情報と交換する AWS STS
SAML 2.0 SP	からのウェブ ID 認証情報に SAML アサーションを交換する AWS STS
OIDC SP	OIDC トークンを からのウェブ ID 認証情報と交換する AWS STS
OAuth 2.0 SP	Amazon、Facebook、Google、Apple、および Twitter の OAuth トークンを のウェブ ID 認証情報と交換する AWS STS
カスタム SP	AWS 認証情報を使用して、任意の形式のクレームを のウェブ ID 認証情報と交換します。AWS STS
非認証アクセス	認証なしで から AWS STS アクセスが制限されたウェブ ID 認証情報を発行する
ロールベースアクセスコントロール	クレームに基づいて認証されたユーザーの IAM ロールを選択し、ID プールのコンテキストでのみ引き受けるようにロールを設定します。
単一ドメイン内の属性ベースの	クレームを AWS STS 一時セッションのプリンシパルタグに変換し、IAM ポリシーを使用してプリンシパルタグに基づいてリソースアクセスをフィルタリングする

アイデンティティプールの詳細については、「Amazon Cognito ID プールの開始方法」と「Amazon Cognito ID プール API リファレンス」を参照してください。

Amazon Cognito ユーザープールとアイデンティティプールの比較

機能	説明	ユーザープール	ID プール

OIDC IdP	OIDC ID トークンを発行してアプリユーザーを認証する	✓	
API 認証サーバー	OAuth 2.0 APIs、データベース、その他のリソースへのユーザーアクセスを許可するアクセストークンを発行する	✓	
IAM ウェブ ID 認証サーバー	一時的な AWS 認証情報 AWS STS と交換できるトークンを生成する		✓
SAML 2.0 SP と OIDC IdP	SAML 2.0 IdP からのクレームに基づいてカスタマイズされた OIDC トークンを発行する	✓	
OIDC SP と OIDC IdP	OIDC IdP からのクレームに基づいてカスタマイズされた OIDC トークンを発行する	✓	
OAuth 2.0 SP と OIDC IdP	Apple や Google などの OAuth 2.0 ソーシャルプロバイダーのスコープに基づいてカスタマイズされた OIDC トークンを発行する	✓	

SAML 2.0 SP および認証情報ブローカー	SAML 2.0 IdP からのクレームに基づいて一時的な AWS 認証情報を発行する		✓
OIDC SP と認証情報ブローカー	OIDC IdP からのクレームに基づいて一時的な AWS 認証情報を発行する		✓
OAuth 2.0 SP および認証情報ブローカー	Apple や Google などの OAuth 2.0 ソーシャルプロバイダーのスコープに基づいて一時的な AWS 認証情報を発行する		✓
Amazon Cognito ユーザープール SP と認証情報ブローカー	Amazon Cognito ユーザープールからの OIDC クレームに基づいて一時的な AWS 認証情報を発行する		✓
カスタム SP および認証情報ブローカー	デベロッパーの IAM 認証に基づいて一時的な AWS 認証情報を発行する		✓
認証フロントエンドサービス	ホストされた UI でユーザーをサインアップ、管理、認証する	✓	
独自の認証 UI の API サポート	サポートされている AWS SDKs 1 で API リクエストを使用してユーザーを作成、管理、認証する	✓	

MFA	追加の認証要素として SMS メッセージ、TOTPs、またはユーザーのデバイスを使用する1	✓	
セキュリティのモニタリングと対応	悪意のあるアクティビティや安全でないパスワードからの保護1	✓	
認証フローをカスタマイズする	独自の認証メカニズムを構築するか、既存のフローにカスタムステップを追加する1	✓	
グループ	トークンを ID プールに渡すときに、ユーザーの論理グループと IAM ロールクレームの階層を作成する	✓	
ID トークンをカスタマイズする	新規、変更、抑制されたクレームで ID トークンをカスタマイズする	✓	
AWS WAF ウェブ ACLs	による認証環境へのリクエストのモニタリングと制御 AWS WAF	✓	
ユーザー属性をカスタマイズする	ユーザー属性に値を割り当て、独自のカスタム属性を追加する	✓	
非認証アクセス	認証なしで から AWS STS アクセスが制限されたウェブ ID 認証情報を発行する		✓

ロールベースアクセスコントロール	クレームに基づいて認証されたユーザーの IAM ロールを選択し、ID プールのコンテキストでのみ引き受けるようにロールを設定します。	✓
単一ドメイン内の属性ベースの	ユーザークレームを一時セッションのプリンシパルタグに変換し、IAM ポリシーを使用してプリンシパルタグに基づいてリソースアクセスをフィルタリングする AWS STS	✓

[1] この機能はローカルユーザーのみが使用できます。

Amazon Cognito の使用開始方法

ユーザープールアプリケーションの例については、「」を参照してくださいユーザープールの開始方法。

ID プールの概要については、「」を参照してくださいAmazon Cognito ID プールの開始方法。

ユーザープールと ID プールを使用したガイド付きセットアップエクスペリエンスへのリンクについては、「」を参照してくださいAmazon Cognito のガイド付きセットアップオプション。

動画、記事、ドキュメント、その他のサンプルアプリケーションについては、Amazon Cognito デベロッパーリソース」を参照してください。

Amazon Cognito を使用するには、 AWS アカウントが必要です。詳細については、「の開始方法 AWS」を参照してください。

リージョナルな可用性

Amazon Cognito は、世界中の複数の AWS リージョンで利用できます。各リージョンで、Amazon Cognito は複数のアベイラビリティーゾーンに分散されています。これらのアベイラビリティーゾーンは物理的に相互に分離されていますが、低レイテンシーで高スループットの冗長性に優れたプライベートネットワーク接続で統合されています。これらのアベイラビリティーゾーンにより、 AWS はレイテンシーを最小限に抑えながら、Amazon Cognito を含む のサービスを非常に高レベルの可用性と冗長性で提供できます。

現在 Amazon Cognito を利用できるすべてのリージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「AWS のリージョンとエンドポイント」を参照してください。各リージョンで利用できるアベイラビリティーゾーンの数の詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon Cognito の料金

Amazon Cognitoの料金については、「Amazon Cognito の料金」を参照してください。

Amazon Cognito ユーザープールと ID プールでの認証の仕組み

顧客が Amazon Cognito ユーザープールにサインインすると、アプリケーションは JSON ウェブトークン (JWTsを受け取ります。

ユーザーがユーザープールトークンまたは別のプロバイダーを使用して ID プールにサインインすると、アプリケーションは一時的な AWS 認証情報を受け取ります。

ユーザープールのサインインを使用すると、 AWS SDK を使用して認証と認可を完全に実装できます。独自のユーザーインターフェイス (UI) コンポーネントを構築しない場合は、構築済みのウェブ UI (ホストされた UI) またはサードパーティー ID プロバイダー (IdP) のサインインページを呼び出すことができます。

このトピックでは、アプリケーションが Amazon Cognito とやり取りして ID トークンで認証し、アクセストークンで認証し、ID プール認証情報 AWS のサービス で にアクセスする方法の概要を説明します。

トピック

- AWS SDK によるユーザープール API 認証と認可

- ホストされた UI によるユーザープール認証
- サードパーティー ID プロバイダーによるユーザープール認証
- ID プール認証

AWS SDK によるユーザープール API 認証と認可

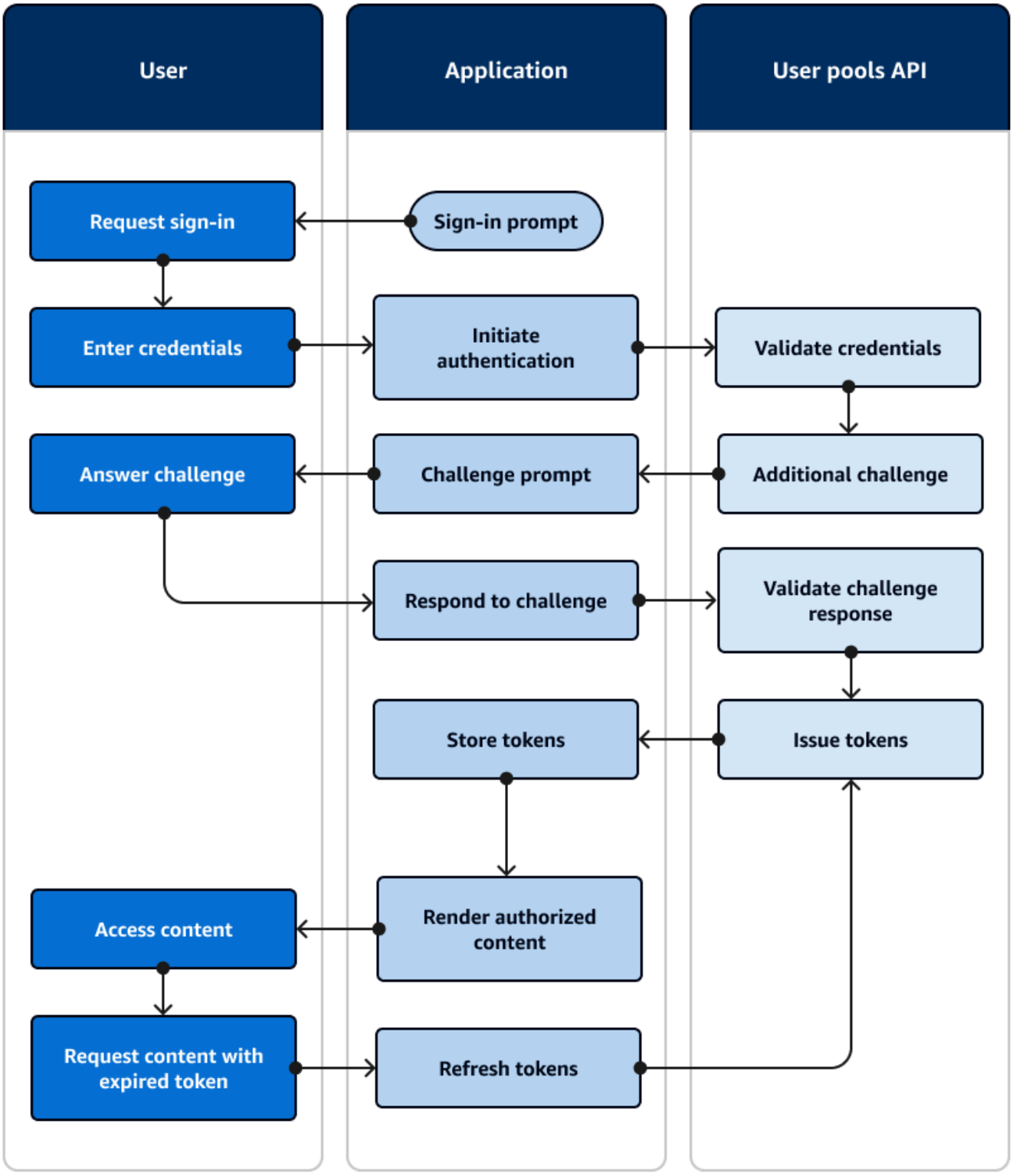
AWS は、さまざまなデベロッパーフレームワーク で Amazon Cognito ユーザープールまたは Amazon Cognito ID プロバイダー のコンポーネントを開発しました。 ???これらの SDKsAmazon Cognito ユーザープール API を呼び出します。同じユーザープール API 名前空間には、ユーザープールの設定とユーザー認証のためのオペレーションがあります。詳細な概要については、「」を参照してくださいAmazon Cognito ユーザープール API とユーザープールのエンドポイントの使用。

API 認証は、アプリケーションに既存の UI コンポーネントがあり、主にユーザープールをユーザーディレクトリとして使用するモデルに適合します。この設計により、Amazon Cognito がより大きなアプリケーション内のコンポーネントとして追加されます。複雑なチャレンジとレスポンスのチェーンを処理するには、プログラムによるロジックが必要です。

このアプリケーションは、OpenID Connect (OIDC) 証明書利用者の完全な実装を実装する必要はありません。代わりに、JWTsをデコードして使用できます。ローカルユーザーのユーザープール機能のすべてにアクセスする場合は、開発環境で Amazon Cognito SDK を使用して認証を構築します。

カスタム OAuth スコープを使用した API 認証は、外部 API 認証に対する指向性が低くなります。API 認証からアクセストークンにカスタムスコープを追加するには、実行時に を使用してトークンを変更しますトークン生成前の Lambda トリガー。

次の図は、API 認証の一般的なサインインセッションを示しています。


User
Application
User pools API
Request sign-in
Sign-in prompt
Enter credentials
Initiate authentication
Validate credentials
Answer challenge
Challenge prompt
Additional challenge
Respond to challenge
Validate challenge response
Store tokens
Issue tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

API 認証フロー

1. ユーザーはアプリケーションにアクセスします。
2. 「サインイン」リンクを選択します。
3. ユーザー名とパスワードを入力します。
4. アプリケーションは API InitiateAuthリクエストを行う メソッドを呼び出します。リクエストは、ユーザーの認証情報をユーザープールに渡します。
5. ユーザープールはユーザーの認証情報を検証し、ユーザーが多要素認証 (MFA) をアクティブ化したと判断します。
6. ユーザープールは、MFA コードをリクエストするチャレンジで応答します。
7. アプリケーションは、ユーザーから MFA コードを収集するプロンプトを生成します。
8. アプリケーションは API RespondToAuthChallengeリクエストを行う メソッドを呼び出します。リクエストはユーザーの MFA コードを渡します。
9. ユーザープールはユーザーの MFA コードを検証します。
10. ユーザープールはユーザーの JWTs。
11. アプリケーションは、ユーザーの JWTs。
12. アプリケーションには、リクエストされたアクセスコントロールコンポーネントが表示されます。
13. ユーザーは自分のコンテンツを表示します。
14. 後で、ユーザーのアクセストークンの有効期限が切れ、アクセスコントロール対象コンポーネントの表示をリクエストします。
15. アプリケーションは、ユーザーのセッションを永続する必要があると判断します。更新トークンを使用して InitiateAuthメソッドを再度呼び出し、新しいトークンを取得します。

バリアントとカスタマイズ

このフローは、独自のカスタム認証チャレンジなど、追加のチャレンジで強化できます。パスワードが侵害されたユーザー、または予期しないサインイン特性が悪意のあるサインインの試みを示している可能性があるユーザーのアクセスを自動的に制限できます。このフローは、サインアップ、ユーザー属性の更新、パスワードのリセットを行うオペレーションでもほぼ同じように見えます。これらのフローのほとんどには、パブリック (クライアント側) と機密 (サーバー側) の API オペレーションが重複しています。

関連リソース

- Amazon Cognito ユーザープール API
- ユーザープールの開始方法
- Amazon Cognito の認証と承認を、ウェブアプリケーションとモバイルアプリケーションに統合する
- Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用

ホストされた UI によるユーザープール認証

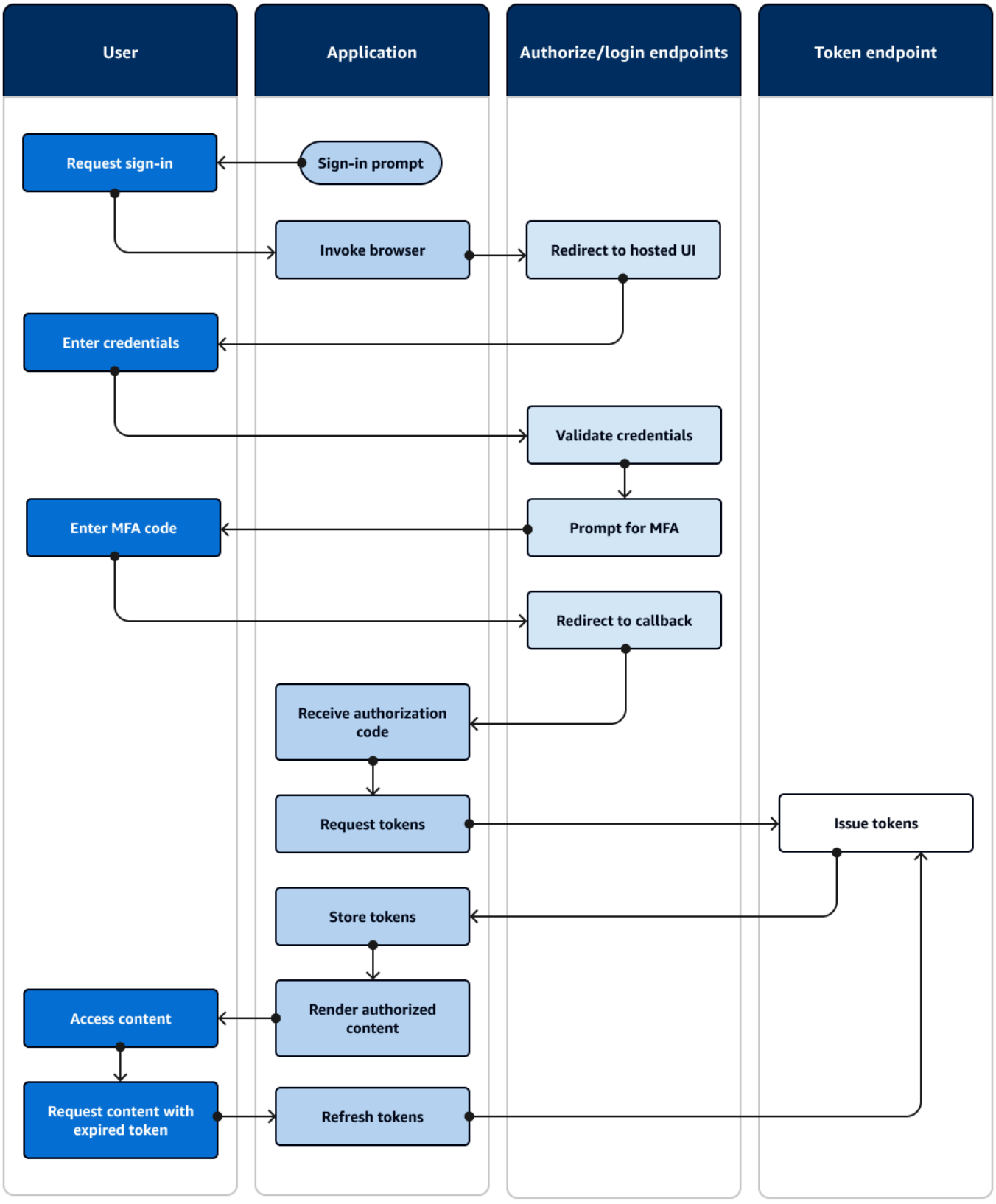
ホストされた UI は、ユーザープールとアプリクライアントにリンクされているウェブサイトです。ユーザーのサインイン、サインアップ、パスワードリセットの各オペレーションを実行できます。認証用のホストされた UI コンポーネントを使用するアプリケーションでは、実装に必要なデベロッパーの労力が軽減されます。アプリケーションは認証用の UI コンポーネントをスキップし、ユーザーのブラウザでホストされた UI を呼び出すことができます。

アプリケーションは、ウェブまたはアプリケーションのリダイレクト場所を使用してユーザーの JWTs を収集します。ホストされた UI を実装するアプリケーションは、OpenID Connect (OIDC) IdP であるかのようにユーザープールに接続して認証できます。

ホストされた UI 認証は、アプリケーションが認証サーバーを必要とするが、カスタム認証、ID プール統合、ユーザー属性セルフサービスなどの機能を必要としないモデルに適合します。これらの高度なオプションの一部を使用する場合は、 SDK のユーザープールコンポーネントを使用して実装できます。

OIDC 実装に主に依存するホストされた UI およびサードパーティーの IdP 認証モデルは、OAuth 2.0 スコープを使用する高度な認証モデルに最適です。

次の図は、ホストされた UI 認証の一般的なサインインセッションを示しています。


User
Application
Authorize/login endpoints
Token endpoint
Request sign-in
Sign-in prompt
Invoke browser
Redirect to hosted UI
Enter credentials
Validate credentials
Enter MFA code
Prompt for MFA
Redirect to callback
Receive authorization code
Request tokens
Issue tokens
Store tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

ホストされた UI 認証フロー

1. ユーザーはアプリケーションにアクセスします。
2. 「サインイン」リンクを選択します。
3. アプリケーションは、ホストされた UI サインインプロンプトにユーザーを誘導します。
4. ユーザー名とパスワードを入力します。
5. ユーザープールはユーザーの認証情報を検証し、ユーザーが多要素認証 (MFA) をアクティブ化したと判断します。
6. ホストされた UI は、ユーザーに MFA コードの入力を求めます。
7. ユーザーが MFA コードを入力します。
8. ホストされた UI は、ユーザーをアプリケーションにリダイレクトします。
9. アプリケーションは、ホストされた UI がコールバック URL に追加した URL リクエストパラメータから認証コードを収集します。
10. アプリケーションは認証コードを使用してトークンをリクエストします。
11. トークンエンドポイントは JWTsをアプリケーションに返します。
12. アプリケーションは、ユーザーの JWTs。
13. アプリケーションには、リクエストされたアクセスコントロールコンポーネントが表示されます。
14. ユーザーは自分のコンテンツを表示します。
15. 後で、ユーザーのアクセストークンの有効期限が切れ、アクセスコントロール対象コンポーネントの表示をリクエストします。
16. アプリケーションは、ユーザーのセッションを永続する必要があると判断します。更新トークンを使用して、トークンエンドポイントに新しいトークンをリクエストします。

バリアントとカスタマイズ

ホストされた UI のルックアンドフィールは、任意のアプリクライアントで CSS を使用してカスタマイズできます。また、独自の ID プロバイダー、スコープ、ユーザー属性へのアクセス、高度なセキュリティ設定を使用してアプリクライアントを設定することもできます。

関連リソース

- Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用
- ホストされた UI でのサインアップとサインイン

- スコープ、M2M 、およびリソースサーバーによる API 認証
- ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス

サードパーティー ID プロバイダーによるユーザープール認証

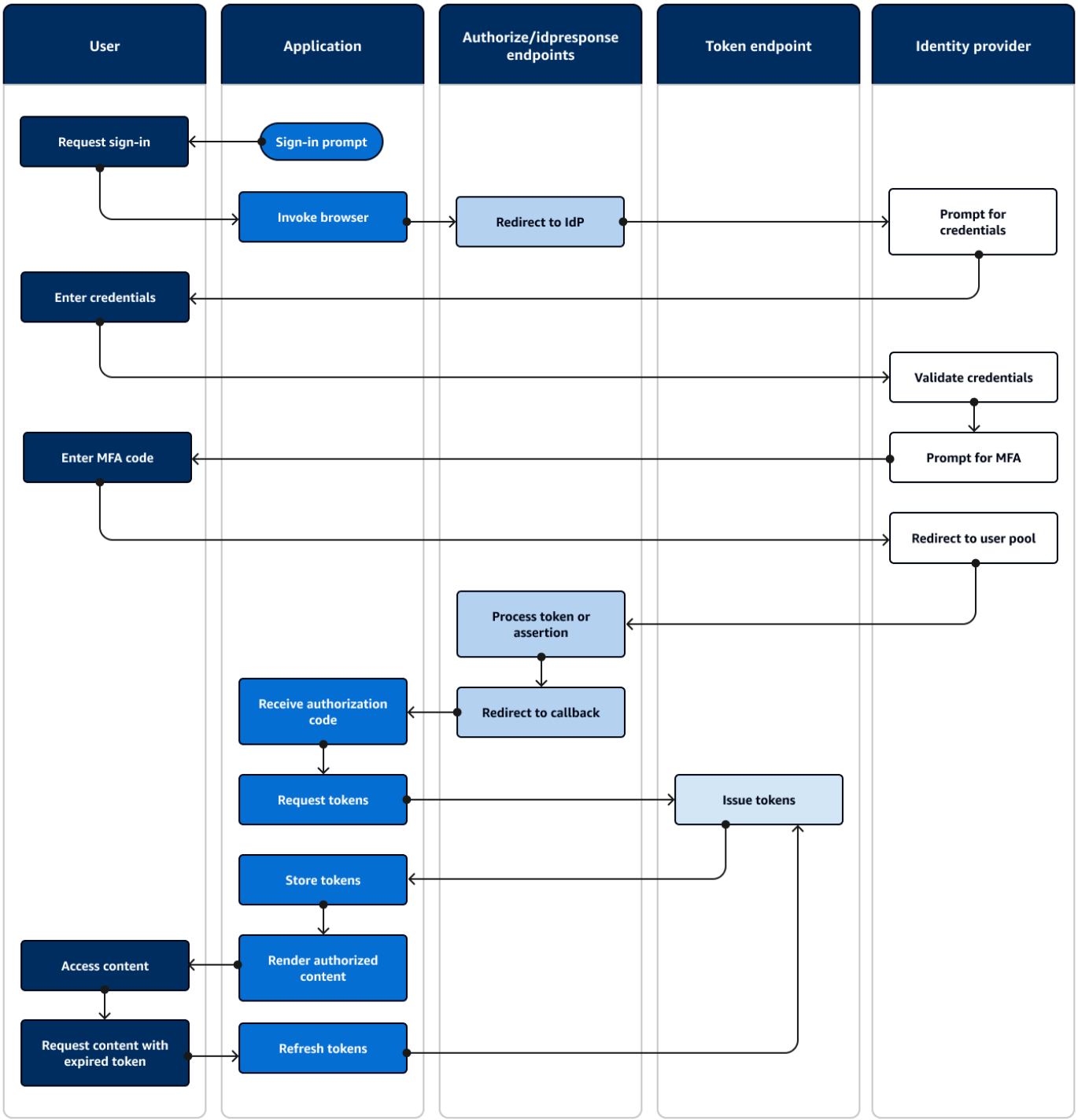
外部 ID プロバイダー (IdP) またはフェデレーション認証 を使用したサインインは、ホストされた UI と同様のモデルです。アプリケーションはユーザープールの OIDC 依存パーティであり、ユーザープールは IdP へのパススルーとして機能します。IdP は、Facebook や Google などのコンシューマーユーザーディレクトリでも、Azure などの SAML 2.0 や OIDC エンタープライズディレクトリでもかまいません。

ユーザーのブラウザでホストされた UI の代わりに、アプリケーションはユーザープール認証サーバー でリダイレクトエンドポイントを呼び出します。ユーザービューから、アプリケーションのサインインボタンを選択します。次に、IdP はサインインするように促します。ホストされた UI 認証と同様に、アプリケーションはアプリケーションのリダイレクト場所に JWTs を収集します。

サードパーティーの IdP による認証は、ユーザーがアプリケーションにサインアップするときに新しいパスワードを入力したくない場合があるモデルに適合します。ホストされた UI 認証が実装されているアプリケーションには、サードパーティー認証を少ない労力で追加できます。実際には、ホストされた UI とサードパーティは、ユーザーのブラウザで呼び出す内容のわずかなバリエーションから一貫した認証結果 IdPs を生成します。

ホストされた UI 認証と同様に、OAuth 2.0 スコープの高度な認証モデルにはフェデレーション認証が最適です。

次の図は、フェデレーション認証の一般的なサインインセッションを示しています。



フェデレーティッド認証フロー

1. ユーザーはアプリケーションにアクセスします。
2. 「サインイン」リンクを選択します。

3. アプリケーションは、IdP を使用してサインインプロンプトをユーザーに指示します。
4. ユーザー名とパスワードを入力します。
5. IdP はユーザーの認証情報を検証し、ユーザーが多要素認証 (MFA) をアクティブ化したと判断します。
6. IdP は、ユーザーに MFA コードの入力を求めます。
7. ユーザーが MFA コードを入力します。
8. IdP は、SAML レスポンスまたは認証コードを使用してユーザーをユーザープールにリダイレクトします。
9. ユーザーが認証コードを渡した場合、ユーザープールはコードを IdP トークンとサイレントに交換します。ユーザープールは IdP トークンを検証し、新しい認証コードを使用してユーザーをアプリケーションにリダイレクトします。
10. アプリケーションは、ユーザープールがコールバック URL に追加した URL リクエストパラメータから認証コードを収集します。
11. アプリケーションは認証コードを使用してトークンをリクエストします。
12. トークンエンドポイントは JWTsをアプリケーションに返します。
13. アプリケーションは、ユーザーの JWTs。
14. アプリケーションには、リクエストされたアクセスコントロールコンポーネントが表示されます。
15. ユーザーは自分のコンテンツを表示します。
16. 後で、ユーザーのアクセストークンの有効期限が切れ、アクセスコントロール対象コンポーネントの表示をリクエストします。
17. アプリケーションは、ユーザーのセッションを永続する必要があると判断します。更新トークンを使用して、トークンエンドポイントに新しいトークンをリクエストします。

バリアントとカスタマイズ

ホストされた UI でフェデレーション認証を開始できます。ユーザーは、アプリクライアント に割り当て IdPs た のリストから選択できます。ホストされた UI は、E メールアドレスの入力を求め、ユーザーのリクエストを対応する SAML IdP に自動的にルーティングすることもできます。 IdP サードパーティーの ID プロバイダーによる認証では、ホストされた UI とユーザーとのやり取りは必要ありません。アプリケーションは、ユーザーの認証サーバーリクエストに リクエストパラメータを追加し、ユーザーがサイレントで IdP サインインページにリダイレクトするようにできます。

関連リソース

- サードパーティー経由のユーザープールへのサインインの追加
- シナリオ例: エンタープライズダッシュボードで Amazon Cognito アプリケーションをブックマークする
- スコープ、M2M 、およびリソースサーバーによる API 認証
- ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス

ID プール認証

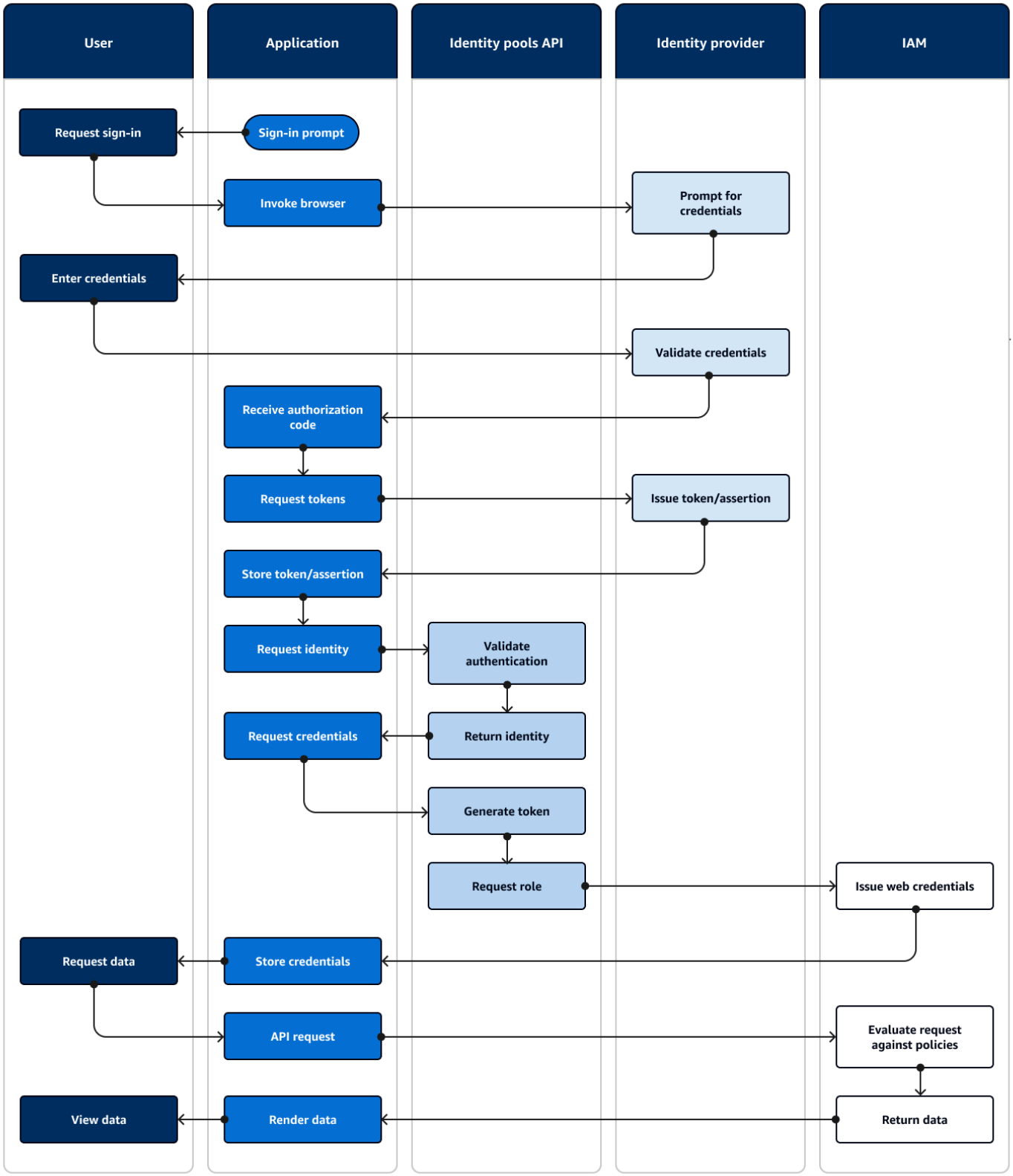
ID プールはアプリケーションのコンポーネントであり、関数、API 名前空間、および SDK モデルのユーザープールとは異なります。ユーザープールがトークンベースの認証と承認を提供する場合、アイデンティティプールは AWS Identity and Access Management (IAM) の認証を提供します。

のセットを ID プール IdPs に割り当てて、それらを使用してユーザーにサインインできます。ユーザープールは ID プールと密接に統合 IdPs されており、ID プールにアクセスコントロールのほとんどのオプションを提供します。同時に、ID プールには幅広い認証オプションがあります。ユーザープールは、SAML、OIDC、ソーシャル、デベロッパー、ゲスト ID ソースを ID プールからの一時的な AWS 認証情報へのルートとして結合します。

ID プールによる認証は外部です。前に示したユーザープールフローのいずれか、または別の IdP で個別に開発するフローに従います。アプリケーションが最初の認証を実行すると、証明を ID プールに渡し、代わりに一時セッションを受け取ります。

ID プールによる認証は、IAM 認証 AWS のサービス を使用して のアプリケーションアセットとデータにアクセスコントロールを適用するモデルに適合します。ユーザープール の API 認証と同様に、成功したアプリケーションには、ユーザーの利益のためにアクセスする各サービスの AWS SDKs が含まれます。 AWS SDKs、ID プール認証からの認証情報を API リクエストの署名として適用します。

次の図は、IdP による ID プール認証の一般的なサインインセッションを示しています。


User
Application
Identity pools API
Identity provider
IAM
Request sign-in
Sign-in prompt
Invoke browser
Prompt for credentials
Enter credentials
Validate credentials
Receive authorization code
Request tokens
Issue token/assertion
Store token/assertion
Request identity
Validate authentication
Request credentials
Return identity
Generate token
Request role
Issue web credentials
Request data
Store credentials
API request
Evaluate request against policies
View data
Render data
Return data

フェデレーティッド認証フロー

1. ユーザーはアプリケーションにアクセスします。
2. 「サインイン」リンクを選択します。
3. アプリケーションは、IdP を使用してサインインプロンプトをユーザーに指示します。
4. ユーザー名とパスワードを入力します。
5. IdP はユーザーの認証情報を検証します。
6. IdP は、SAML レスポンスまたは認証コードを使用してユーザーをアプリケーションにリダイレクトします。
7. ユーザーが認証コードを渡した場合、アプリケーションはコードを IdP トークンと交換します。
8. アプリケーションは、ユーザーの JWTs、キャッシュします。
9. アプリケーションは API GetIdリクエストを行う メソッドを呼び出します。ユーザーのトークンまたはアサーションを渡し、アイデンティティ ID をリクエストします。
10. ID プールは、設定された ID プロバイダーに対してトークンまたはアサーションを検証します。
11. ID プールは ID を返します。
12. アプリケーションは API GetCredentialsForIdentityリクエストを行う メソッドを呼び出します。ユーザーのトークンまたはアサーションを渡し、IAM ロールをリクエストします。
13. ID プールは新しい JWT を生成します。新しい JWT には、IAM ロールをリクエストするクレームが含まれています。ID プールは、IdP の ID プール設定のユーザーのリクエストとロール選択基準に基づいてロールを決定します。
14. AWS Security Token Service （ AWS STS) は ID プールからのAssumeRoleWithWebIdentityリクエストに応答します。レスポンスには、IAM ロールを持つ一時セッションの API 認証情報が含まれています。
15. アプリケーションはセッション認証情報を保存します。
16. ユーザーは、 のアクセス保護されたリソースを必要とするアクションをアプリで実行します AWS。
17. アプリケーションは、必要な の API リクエストに署名として一時的な認証情報を適用します AWS のサービス。
18. IAM は、認証情報のロールにアタッチされたポリシーを評価します。リクエストと比較します。
19. はリクエストされたデータ AWS のサービス を返します。
20. アプリケーションは、ユーザーのインターフェイスでデータをレンダリングします。
21. ユーザーはデータを表示します。

バリアントとカスタマイズ

ユーザープールによる認証を視覚化するには、「トークン/アサーションの問題」ステップの後に、前のユーザープールの概要のいずれかを挿入します。デベロッパー認証は、リクエスト ID の前にあるすべてのステップを、デベロッパー認証情報によって署名されたリクエストに置き換えます。また、ゲスト認証では、リクエスト ID に直接スキップし、認証を検証せず、アクセスが制限されたIAM ロールの認証情報を返します。

関連リソース

- Amazon Cognito アイデンティティプール
- ユーザー IAM ロール
- ID プールの概念
- ID プール (フェデレーティッドアイデンティティ) の認証フロー

Amazon Cognito の用語

Amazon Cognito は、ウェブアプリとモバイルアプリの認証情報を提供します。アイデンティティとアクセス管理で共通の用語から抽出され、構築されます。ユニバーサルアイデンティティとアクセス条件に関する多くのガイドが用意されています。いくつか例を挙げます。

- IDPro ナレッジボディの用語
- AWS ID サービス
- NIST CSRC の用語集

次のリストは、Amazon Cognito に固有の用語、または Amazon Cognito に特定のコンテキストを持つ用語を示しています。

トピック

- 全般
- ユーザープール
- アイデンティティプール

全般

このリストの用語は Amazon Cognito に固有のものではなく、ID とアクセス管理の実務者の間で広く認識されています。以下は、用語の完全なリストではありませんが、このガイドの特定の Amazon Cognito コンテキストのガイドです。

アプリケーション

通常、モバイルアプリケーションです。このガイドでは、アプリケーションは Amazon Cognito に接続するウェブアプリケーションまたはモバイルアプリの省略形であることがよくあります。

属性ベースのアクセスコントロール (ABAC)

アプリケーションのジョブタイトルや部門など、ユーザーのプロパティに基づいてリソースへのアクセスを決定するモデル。ABAC を適用する Amazon Cognito ツールには、ユーザープールの ID トークンと ID プールのプリンシパルタグが含まれます。

認証サーバー

JSON ウェブトークン を生成するウェブベースのシステム。Amazon Cognito ユーザープールフェデレーションエンドポイントは、ユーザープールの 2 つの認証方法と認可方法の認可サーバーコンポーネントです。もう 1 つの方法は、ユーザープール API です。

機密アプリケーション、サーバー側のアプリ

ユーザーがアプリケーションサーバー上のコードとシークレットへのアクセスを使用してリモートに接続するアプリケーション。これは通常、ウェブアプリケーションです。

ID プロバイダー (IdP)

ユーザー ID を保存および検証するサービス。Amazon Cognito は、外部プロバイダーからの認証をリクエストし、アプリへの IdP にすることができます。

JSON ウェブトークン (JWT)

認証されたユーザーに関するクレームを含む JSON 形式のドキュメント。ID トークンはユーザーを認証し、アクセストークンはユーザーを認証し、更新トークンは認証情報を更新します。Amazon Cognito は外部プロバイダーからトークンを受け取り、アプリケーションまたは にトークンを発行します AWS STS。

多要素認証 (MFA)

ユーザーがユーザー名とパスワードを入力した後に追加の認証を提供する要件。Amazon Cognito ユーザープールには、ローカルユーザー 用の MFA 機能があります。

OAuth 2.0 (マニフェスト) プロバイダー

JWT アクセストークンと更新トークンを提供するユーザープールまたは ID プールへの IdP。Amazon Cognito ユーザープールは、ユーザーの認証後にソーシャルプロバイダーとのやり取りを自動化します。

OpenID Connect (OIDC) プロバイダー

OAuth 仕様を拡張して ID トークンを提供するユーザープールまたは ID プールへの IdP。Amazon Cognito ユーザープールは、ユーザーの認証後に OIDC プロバイダーとのやり取りを自動化します。

パブリックアプリ

デバイスに自己完結型で、コードがローカルに保存され、シークレットにアクセスできないアプリケーション。これは通常、モバイルアプリです。

リソースサーバー

アクセスコントロールを持つ API。Amazon Cognito ユーザープールは、リソースサーバーを使用して、API とやり取りするための設定を定義するコンポーネントを記述します。

ロールベースのアクセスコントロール (RBAC)

ユーザーの機能指定に基づいてアクセス権を付与するモデル。Amazon Cognito アイデンティティプールは、IAM ロール間に差異がある RBAC を実装します。

サービスプロバイダー (SP)、証明書利用者 (RP)

IdP に依存して、ユーザーが信頼できるとアサートするアプリケーション。Amazon Cognito は IdPs、外部 の SP、およびアプリベースの SP の IdP として機能します。 SPs

SAMLprovider

ユーザーが Amazon Cognito に渡すデジタル署名されたアサーションドキュメントを生成するユーザープールまたは ID プールへの IdP。

ユニバーサル一意識別子 (UUID)

オブジェクトに適用される 128 ビットラベル。Amazon Cognito UUIDsは、ユーザープールまたは ID プールごとに一意です。

ユーザーディレクトリ

他のシステムにその情報を提供するユーザーとその属性のコレクション。Amazon Cognito ユーザープールはユーザーディレクトリであり、外部ユーザーディレクトリからユーザーを統合するためのツールでもあります。

ユーザープール

このガイドの次のリストにある用語は、特定の機能またはユーザープールの設定を指します。

Amazon Cognito ユーザープール API

AWS SDK を使用してアプリに追加できる認証および認可 API オペレーションのセット。API は、ローカルユーザーとリンクされたユーザー にサインインできます。

アダプティブ認証

潜在的な悪意のあるアクティビティを検出し、ユーザープロファイル に追加のセキュリティを適用するアドバンストセキュリティの機能。

アドバンストセキュリティ機能

ユーザーセキュリティ用のツールを追加するオプションコンポーネント。

アプリクライアント

ユーザープールの設定を 1 つのアプリへの IdP として定義するコンポーネント。

コールバック URL、リダイレクト URI

アプリクライアントの 設定と、ユーザープールフェデレーションエンドポイントへのリクエストの パラメータ。コールバック URL は、アプリ内の認証済みユーザーの初期送信先です。

漏えいした認証情報

攻撃者が知る可能性のあるユーザーパスワードを検出し、ユーザープロファイルに追加のセキュリティを適用するアドバンストセキュリティの機能。

確認

新しいユーザーがサインインできるように、前提条件が満たされていると判断するプロセス。確認は通常、E メールアドレスまたは電話番号検証を通じて行われます。

カスタム認証

追加のユーザーチャレンジとレスポンスを定義する Lambda トリガーによる認証プロセスの拡張。

デバイス認証

MFA を信頼できるデバイスの ID を使用するサインインに置き換える認証プロセス。

外部プロバイダー、サードパーティープロバイダー

ユーザープールとの信頼関係を持つ IdP。

フェデレーティッドユーザー

外部プロバイダー によって認証されたユーザープール内のユーザー。

フェデレーションエンドポイント

IdPs および アプリケーションとのやり取りのために サービスをホストするユーザープールドメイン上のウェブページのセット。

ホストされた UI

ユーザー認証のために サービスをホストするユーザープールドメイン上の一連のインタラクティブなウェブページ。

Lambda トリガー

ユーザープール AWS Lambda がユーザー認証プロセスのキーポイントで自動的に を呼び出すことができる の関数。Lambda トリガーを使用して認証結果をカスタマイズできます。

ローカルユーザー

外部プロバイダーとの認証によって作成されなかったユーザープールユーザーディレクトリ内のユーザープロファイル。

リンクされたユーザー

ID がローカルユーザー とマージされた外部プロバイダーのユーザー。 ???

トークンのカスタマイズ

実行時にユーザーの ID またはアクセストークンを変更するトークン生成前の Lambda トリガーの結果。

ユーザープール、Amazon Cognito ID プロバイダー、**cognito-idp**、Amazon Cognito ユーザープール

OIDC と連携するアプリケーションの認証および認可サービスを含む AWS リソース IdPs。

ユーザープールドメイン

ユーザープールに追加するウェブサイト名。ドメインは、ホストされた UI とフェデレーションエンドポイント のベース URL です。

検証

ユーザーが E メールアドレスまたは電話番号を所有していることを確認するプロセス。ユーザープールは、新しい E メールアドレスまたは電話番号を入力したユーザーにコードを送信しま

す。Amazon Cognito にコードを送信すると、メッセージ送信先の所有権が検証され、ユーザープールから追加のメッセージを受信できます。また、「確認」も参照してください。

ユーザープロファイル、ユーザーアカウント

ユーザーディレクトリ 内のユーザーのエントリ。すべてのユーザーは、ユーザープールにプロファイルを持っています。

アイデンティティプール

このガイドの次のリストにある用語は、特定の機能または ID プールの設定を指します。

アクセスコントロールの属性

ID プールでの属性ベースのアクセスコントロールの実装。ID プールは、ユーザー属性をユーザー認証情報にタグとして適用します。

基本 (クラシック) 認証

ユーザー認証情報 のリクエストをカスタマイズできる認証プロセス。

デベロッパーが認証した ID

デベロッパー認証情報 を使用して ID プールのユーザー認証情報を承認する認証プロセス。 ???

デベロッパー認証情報

ID プール管理者の IAM API キー。

拡張認証

IAM ロールを選択し、ID プールで定義したロジックに従ってプリンシパルタグを適用する認証フロー。

アイデンティティ

アプリユーザーとそのユーザー認証情報を、ID プールとの信頼関係を持つ外部ユーザーディレクトリのプロファイルにリンクする UUID。

ID プール、Amazon Cognito フェデレーティッド ID、Amazon Cognito ID、 **`cognito-identity`**

一時的な AWS 認証情報 を使用するアプリケーションの認証および認可サービスを含む AWS リソース。

認証されていない ID

ID プール IdP でサインインしていないユーザー。認証する前に、単一の IAM ロールに対して制限付きユーザー認証情報を生成することをユーザーに許可できます。

ユーザー認証情報

ID プール認証後にユーザーが受け取る一時的な AWS API キー。

AWS SDK でこのサービスを使用する

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

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例

SDK ドキュメント	コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

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

の開始方法 AWS

Amazon Cognito の使用を開始する前に、必要な AWS リソースをいくつか用意してください。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

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

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

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [アカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

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

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

ユーザープールの開始方法

このセクションのガイドを使用して、最初のユーザープールリソースを作成できます。 step-by-step チュートリアルでは、React JavaScript デベロッパー環境の基本的なウェブアプリケーションから始めます。そこから、ホストされたユーザーインターフェイス (ホストされた UI) や、外部のソーシャルプロバイダーまたは SAML 2.0 ID プロバイダーとのフェデレーションサインイン () などの機能を引き続き追加できますIdPs。

機能セットを拡張し、Amazon Cognito のより多くのコンポーネントを組み込むときは、「Amazon Cognito ユーザープール」の章を読んで、ユーザープールでできることの詳細を確認してください。

このセクションのユーザープールとアプリケーションの例は、アプリケーションリソースと Amazon Cognito ユーザープールの基本的な統合を示しています。後で、使用可能なオプションをさらに使用できるようにユーザープールを調整できます。その後、アプリケーションを更新して新しい APIs し、ホストされた UI と とやり取りできます IdPs。

このセクションのチュートリアルでは、カスタム UI と AWS SDK による API ベースの認証を使用してアプリケーションを作成します。この方法で構築するアプリケーションは、ローカルユーザーの認証に最適です。事前構築された UI、一部のユーザープール機能の自動処理、フェデレーティッドユーザーの認証を使用してアプリケーションを開始するには、「」に進みますホストされた UI でアプリクライアントを追加する。

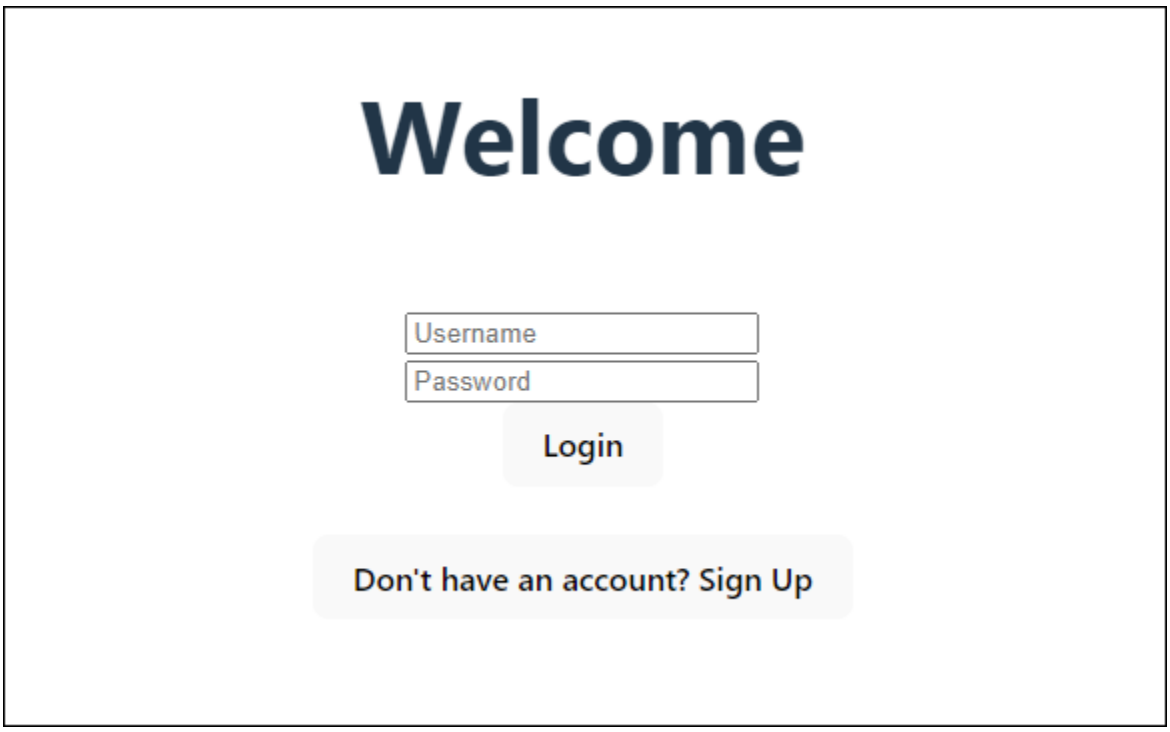
トピック

- React 単一ページアプリケーションの例をセットアップする
- Flutter でサンプル Android アプリをセットアップする
- 次のステップ

React 単一ページアプリケーションの例をセットアップする

このチュートリアルでは、ユーザーのサインアップ、確認、サインインをテストできる React 単一ページアプリケーションを作成します。React は、ウェブおよびモバイルアプリ用の JavaScriptベースのライブラリで、ユーザーインターフェイス (UI) に焦点を当てています。このサンプルアプリケーションは、Amazon Cognito ユーザープールの基本的な機能をいくつか示します。React を使用したウェブアプリケーション開発をすでに経験している場合は、 からサンプルアプリケーションをダウンロードしてください GitHub。

次のスクリーンショットは、作成するアプリケーションの最初の認証ページを示しています。



「ユーザープールの作成」の手順では、サンプルアプリケーションで動作するユーザープールを設定します。以下の要件を満たすユーザープールがある場合は、このステップをスキップできます。

- ユーザーは自分の E メールアドレスでサインインできます。Cognito ユーザープールのサインインオプション： E メール。
- ユーザー名では大文字と小文字は区別されません。ユーザー名の要件: ユーザー名の大文字と小文字を区別しない。
- 多要素認証 (MFA) は必要ありません。MFA 強制: オプションの MFA 。
- ユーザープールは、ユーザープロファイルの確認の属性を E メールメッセージで検証します。を検証する属性: E メールメッセージを送信し、E メールアドレス を検証します。
- E メールは唯一の必須属性です。必須属性： E メール。
- ユーザーはユーザープールにサインアップできます。自己登録 : 自己登録を有効にする が選択されています。
- 初期アプリクライアントは、ユーザー名とパスワードによるサインインを許可するパブリッククライアントです。アプリタイプ : パブリッククライアント 、認証フロー : `ALLOW_USER_PASSWORD_AUTH`。

ユーザープールを作成する

新しいユーザープールを作成する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ユーザープールの作成ボタンを選択します。このオプションを表示するには、左側のナビゲーションペインからユーザープールを選択する必要がある場合があります。
3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。
4. 「サインインエクスペリエンスの設定」で、このユーザープールで使用する ID プロバイダー (IdPs) を選択できます。詳細については、「サードパーティー経由のユーザープールへのサインインの追加」を参照してください。

   a. 認証プロバイダー で、プロバイダータイプ で、Cognito ユーザープールのみが選択されていることを確認します。

   b. Cognito ユーザープールのサインインオプション で、ユーザー名 を選択します。追加のユーザー名要件 を選択しないでください。

   c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。
5. セキュリティ要件の設定 では、パスワードポリシー、多要素認証 (MFA) 要件、ユーザーアカウント復旧オプションを選択できます。詳細については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。

   a. パスワードポリシー の場合、パスワードポリシーモードが Cognito のデフォルト に設定されていることを確認します。

   b. 多要素認証 で、MFA 強制 にオプションの MFA を選択します。

   c. MFA メソッド で、認証アプリケーション と SMS メッセージ を選択します。

   d. ユーザーアカウント復旧 で、セルフサービスアカウントの復旧を有効にする が選択され、ユーザーアカウント復旧メッセージの配信方法が E メールのみ に設定されていることを確認します。

   e. 他のすべてのオプションをデフォルトのままにして、次へを選択します。
6. 「サインアップエクスペリエンスの設定」では、新規ユーザーとしてサインアップするときに新規ユーザーが ID を検証する方法と、ユーザーのサインアップフローで必須またはオプションとなる属性を決定できます。詳細については、「ユーザープール内のユーザーを管理する」を参照してください。

a. 自己登録を有効にするが選択されていることを確認します。この設定により、ユーザープールが開き、インターネット上の全員からサインアップできます。これはサンプルアプリケーションを目的としていますが、この設定は本番環境では慎重に適用してください。

b. Cognito が支援する検証および確認 で、Cognito がメッセージを自動的に送信して検証および確認チェックボックスが選択されていることを確認します。

c. 検証する属性が E メールメッセージの送信に設定されていることを確認し、E メールアドレス を確認します。

d. 属性の変更の検証で、デフォルトのオプションが選択されていることを確認します。更新が保留中の場合は元の属性値を保持し、更新が保留中の場合はアクティブな属性値が E メールアドレス に設定されます。

e. 「必須属性」で、以前の選択に基づく必須属性に E メール が表示されていることを確認します。

> **Important**
>
> このサンプルアプリケーションでは、ユーザープールで phone_number を必須属性として設定しないでください。phone_number が必須属性として表示される場合は、前の選択肢を確認して更新します。
>
> - オプションの MFA、ユーザーアカウント復旧メッセージの配信方法の E メールのみ
> - E メールメッセージを送信し、検証する属性の E メールアドレスを確認する

f. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

7. メッセージ配信の設定 では、サインアップ、アカウント確認、MFA、アカウント復旧のために E メールおよび SMS メッセージをユーザーに送信するように Amazon Simple Email Service および Amazon Simple Notification Service との統合を設定できます。詳細については、「Amazon Cognito ユーザープールの E メール設定」および「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。

   a. E メールプロバイダー で、Cognito で E メールを送信 を選択し、Amazon Cognito が提供するデフォルトの E メール送信者を使用します。E メール量が少ない場合のこの設定は、アプリケーションのテストには十分です。Amazon Simple Email Service (Amazon SES) で E メールアドレスを検証し、Amazon SES で E メールを送信 を選択すると、 を返すことができます。

b. SMS の場合、新しい IAM ロールの作成を選択し、IAM ロール名を入力します。これにより、SMS メッセージを送信するためのアクセス許可を Amazon Cognito に付与するロールが作成されます。

c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

8. 「アプリの統合」では、ユーザープールに名前を付け、ホストされた UI を設定し、アプリクライアントを作成できます。詳細については、「ホストされた UI でアプリクライアントを追加する」を参照してください。サンプルアプリケーションはホストされた UI を使用しません。

a. ユーザープール名 に、ユーザープール名 を入力します。

b. Cognito でホストされた UI の使用 は選択しないでください。

c. 「初期アプリケーションクライアント」で、アプリタイプがパブリッククライアント に設定されていることを確認します。

d. クライアントシークレット で、クライアントシークレットを生成しない が選択されていることを確認します。

e. [App client name] (アプリケーションクライアント名) を入力します。

f. アドバンストアプリケーションクライアント設定 を展開します。認証フローのリストALLOW_USER_PASSWORD_AUTHに を追加します。

g. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

9. 確認と作成 画面で選択内容を確認し、必要に応じて選択内容を変更します。ユーザープールの設定に問題がなければ、ユーザープールの作成を選択して続行します。

10. ユーザープール ページで、新しいユーザープールを選択します。

11. ユーザープールの概要 で、ユーザープール ID を書き留めます。この文字列は、サンプルアプリケーションを作成するときに指定します。

12. アプリ統合 タブを選択し、アプリクライアントと分析 セクションを見つけます。新しいアプリクライアントを選択します。クライアント ID を書き留めます。

関連リソース

- Amazon Cognito user pools
- ユーザープール認証フロー
- ユーザープールでのトークンの使用

アプリケーションの作成

このアプリケーションを構築するには、デベロッパー環境を設定する必要があります。デベロッパー環境の要件は次のとおりです。

1. Node.js がインストールされ、更新されます。
2. ノードパッケージマネージャー (npm) がインストールされ、バージョン 10.2.3 以上に更新されます。
3. 環境には、ウェブブラウザの TCP ポート 5173 からアクセスできます。

React ウェブアプリケーションの例を作成するには

1. デベロッパー環境にサインインし、アプリケーションの親ディレクトリに移動します。

```
cd ~/path/to/project/folder/
```

2. 新しい React サービスを作成します。

```
npm create vite@latest frontend-client -- --template react-ts
```

3. の AWS コード例リポジトリからcognito-developer-guide-react-exampleプロジェクトフォルダのクローンを作成します GitHub。

```
cd ~/some/other/path
```

```
git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

```
cp -r ./aws-doc-sdk-examples/javascriptv3/example_code/cognito-identity-provider/
scenarios/cognito-developer-guide-react-example/frontend-client ~/path/to/project/
folder/frontend-client
```

4. プロジェクトの src ディレクトリに移動します。

```
cd ~/path/to/project/folder/frontend-client/src
```

5. 次の値を編集config.tsして置き換えます。

   a. を AWS リージョン コードYOUR_AWS_REGIONに置き換えます。例: us-east-1。

b. を、テスト用に指定したユーザープールの ID YOUR_COGNITO_USER_POOL_IDに置き換えます。例: us-east-1_EXAMPLE。ユーザープールは、前のステップで AWS リージョン 入力した にある必要があります。

c. を、テスト用に指定したアプリクライアントの ID YOUR_COGNITO_APP_CLIENT_IDに置き換えます。例: 1example23456789。アプリクライアントは、前のステップのユーザープールにある必要があります。

6. 以外の IP からサンプルアプリケーションにアクセスする場合はlocalhost、行を編集package.jsonして "dev": "vite", に変更します"dev": "vite --host 0.0.0.0",。

7. アプリケーションをインストールします。

```
npm install
```

8. アプリケーションを起動します。

```
npm run dev
```

9. http://localhost:5173 または のウェブブラウザでアプリケーションにアクセスしますhttp://[IP address]:5173。

10. 有効な E メールアドレスで新しいユーザーをサインアップします。

11. E メールメッセージから確認コードを取得します。アプリケーションに確認コードを入力します。

12. ユーザー名とパスワードを使用してサインインします。

Amazon Lightsail を使用した React デベロッパー環境の作成

このアプリケーションの使用を開始する簡単な方法は、Amazon Lightsail を使用して仮想クラウドサーバーを作成することです。

Lightsail を使用すると、このサンプルアプリケーションの前提条件があらかじめ設定された小さなサーバーインスタンスをすばやく作成できます。ブラウザベースのクライアントを使用してインスタンスに SSH 接続し、パブリックまたはプライベート IP アドレスでウェブサーバーに接続できます。

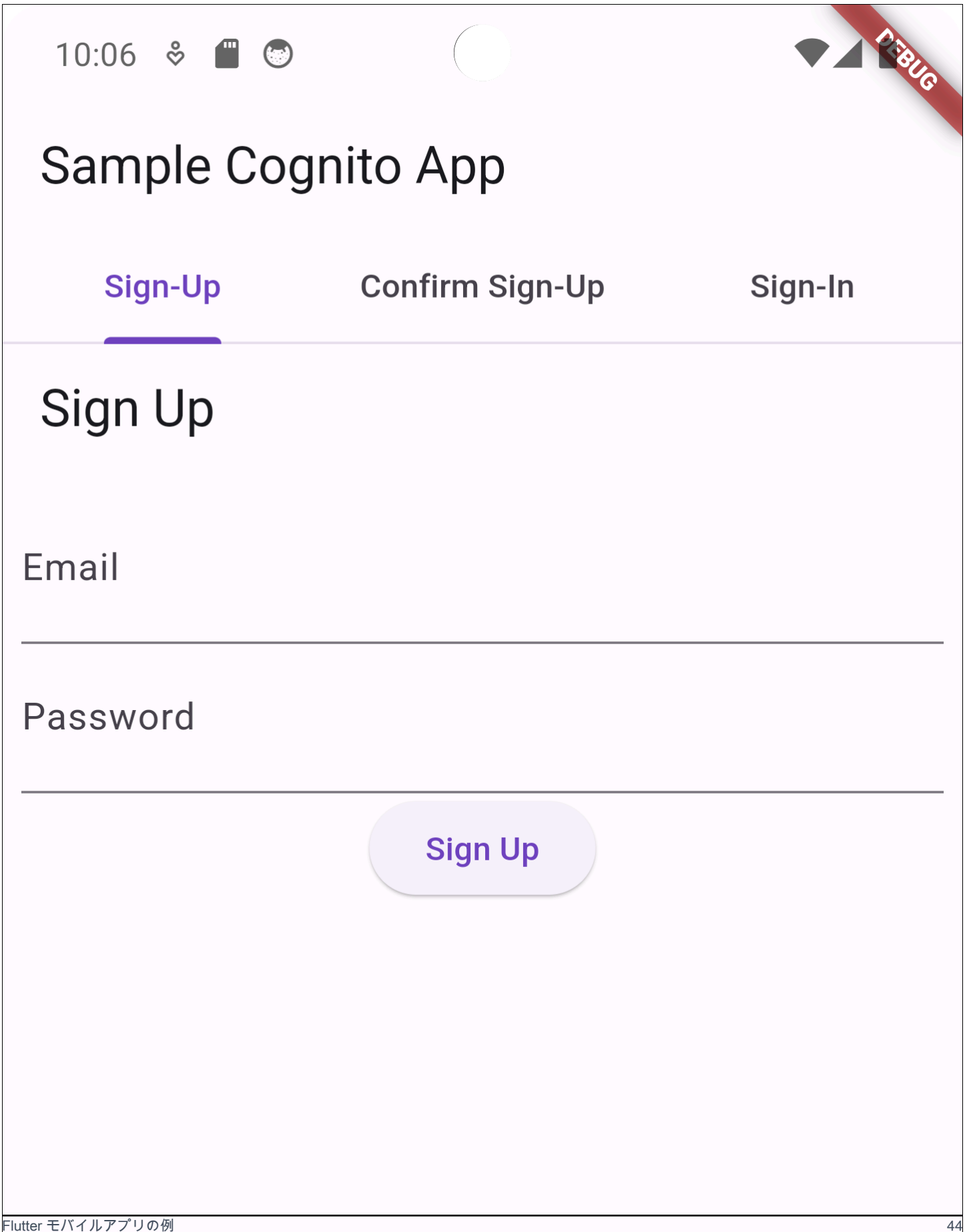
このサンプルアプリケーションの Lightsail インスタンスを作成するには

1. Lightsail コンソール に移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [インスタンスの作成] を選択します。
3. プラットフォームの選択 で、Linux/Unix を選択します。
4. 設計図 を選択する で、Node.js を選択します。
5. 「インスタンスを識別する」で、開発環境にわかりやすい名前を付けます。
6. [インスタンスの作成] を選択します。
7. Lightsail がインスタンスを作成したら、そのインスタンスを選択し、接続 タブから SSH を使用して接続 を選択します。
8. ブラウザウィンドウで SSH セッションが開きます。`node -v` と `npm -v`を実行して、インスタンスが Node.js と 10.2.3 の最小 npm バージョンでプロビジョニングされたことを確認します。
9. React アプリケーションの設定に進みます。

Flutter でサンプル Android アプリをセットアップする

このチュートリアルでは、デバイスをエミュレートし、ユーザーのサインアップ、確認、サインインをテストできるモバイルアプリケーションを Android Studio に作成します。このサンプルアプリケーションは、Flutter で Android 用の基本的な Amazon Cognito ユーザープールモバイルクライアントを作成します。Flutter を使用したモバイルアプリ開発をすでに経験している場合は、 からサンプルアプリケーションをダウンロードしてください GitHub。

次のスクリーンショットは、仮想 Android デバイスで実行されているアプリを示しています。


10:06
DEBUG
Sample Cognito App
Sign-Up
Confirm Sign-Up
Sign-In
Sign Up
Email
Password
Sign Up

「ユーザープールの作成」の手順では、サンプルアプリケーションで動作するユーザープールを設定します。以下の要件を満たすユーザープールがある場合は、このステップをスキップできます。

- ユーザーは自分の E メールアドレスでサインインできます。Cognito ユーザープールのサインインオプション： E メール。
- ユーザー名では大文字と小文字は区別されません。ユーザー名の要件: ユーザー名の大文字と小文字を区別しない。
- 多要素認証 (MFA) は必要ありません。MFA の適用: オプションの MFA 。
- ユーザープールは、ユーザープロファイルの確認の属性を E メールメッセージで検証します。を検証する属性: E メールメッセージを送信し、E メールアドレス を検証します。
- E メールは唯一の必須属性です。必須属性: E メール。
- ユーザーはユーザープールにサインアップできます。自己登録 : 自己登録を有効にする が選択されています。
- 初期アプリクライアントは、ユーザー名とパスワードによるサインインを許可するパブリッククライアントです。アプリタイプ : パブリッククライアント 、認証フロー : `ALLOW_USER_PASSWORD_AUTH`。

ユーザープールを作成する

新しいユーザープールを作成する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ユーザープールの作成ボタンを選択します。このオプションを表示するには、左側のナビゲーションペインからユーザープールを選択する必要がある場合があります。
3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。
4. 「サインインエクスペリエンスの設定」で、このユーザープールで使用する ID プロバイダー (IdPs) を選択できます。詳細については、「サードパーティー経由のユーザープールへのサインインの追加」を参照してください。

    a. 認証プロバイダー では、プロバイダータイプ で、Cognito ユーザープールのみが選択されていることを確認します。

    b. Cognito ユーザープールのサインインオプション で、ユーザー名 を選択します。追加のユーザー名要件 を選択しないでください。

c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

5. セキュリティ要件の設定 では、パスワードポリシー、多要素認証 (MFA) 要件、ユーザーアカウント復旧オプションを選択できます。詳細については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。

   a. パスワードポリシー の場合、パスワードポリシーモードが Cognito のデフォルト に設定されていることを確認します。

   b. 多要素認証 で、MFA 強制 にオプションの MFA を選択します。

   c. MFA メソッド で、認証アプリケーション と SMS メッセージ を選択します。

   d. ユーザーアカウント復旧 で、セルフサービスアカウントの復旧を有効にする が選択され、ユーザーアカウント復旧メッセージの配信方法が E メールのみ に設定されていることを確認します。

   e. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

6. 「サインアップエクスペリエンスの設定」では、新規ユーザーとしてサインアップするときに新規ユーザーが ID を検証する方法と、ユーザーのサインアップフローで必須またはオプションとなる属性を決定できます。詳細については、「ユーザープール内のユーザーを管理する」を参照してください。

   a. 自己登録を有効にするが選択されていることを確認します。この設定により、ユーザープールが開き、インターネット上の全員からサインアップできます。これはサンプルアプリケーションを目的としていますが、この設定は本番環境では慎重に適用してください。

   b. Cognito が支援する検証および確認 で、Cognito がメッセージを自動的に送信して検証および確認チェックボックスが選択されていることを確認します。

   c. 検証する属性が E メールメッセージの送信に設定されていることを確認し、E メールアドレス を確認します。

   d. 属性の変更の検証で、デフォルトのオプションが選択されていることを確認します。更新が保留中の場合は元の属性値を保持し、更新が保留中の場合はアクティブな属性値が E メールアドレス に設定されます。

   e. 「必須属性」で、以前の選択に基づく必須属性に E メール が表示されていることを確認します。

> **Important**
>
> このサンプルアプリケーションでは、ユーザープールで phone_number を必須属性として設定しないでください。phone_number が必須属性として表示される場合は、前の選択肢を確認して更新します。
>
> - オプションの MFA、ユーザーアカウント復旧メッセージの配信方法の E メールのみ
> - E メールメッセージを送信し、検証する属性の E メールアドレスを確認する

f. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

7. メッセージ配信の設定 では、サインアップ、アカウント確認、MFA、アカウント回復のために E メールおよび SMS メッセージをユーザーに送信するように Amazon Simple Email Service および Amazon Simple Notification Service との統合を設定できます。詳細については、「Amazon Cognito ユーザープールの E メール設定」および「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。

   a. E メールプロバイダー で、Cognito で E メールを送信 を選択し、Amazon Cognito が提供するデフォルトの E メール送信者を使用します。E メール量が少ない場合のこの設定は、アプリケーションのテストには十分です。Amazon Simple Email Service (Amazon SES) で E メールアドレスを検証し、Amazon SES で E メールを送信 を選択すると、 を返すことができます。

   b. SMS の場合、新しい IAM ロールの作成を選択し、IAM ロール名を入力します。これにより、SMS メッセージを送信するためのアクセス許可を Amazon Cognito に付与するロールが作成されます。

   c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

8. 「アプリの統合」では、ユーザープールに名前を付け、ホストされた UI を設定し、アプリクライアントを作成できます。詳細については、「ホストされた UI でアプリクライアントを追加する」を参照してください。サンプルアプリケーションはホストされた UI を使用しません。

   a. ユーザープール名 に、ユーザープール名 を入力します。

   b. Cognito でホストされた UI の使用 は選択しないでください。

   c. 「初期アプリケーションクライアント」で、アプリタイプがパブリッククライアント に設定されていることを確認します。

d. クライアントシークレット で、クライアントシークレットを生成しない が選択されていることを確認します。

e. [App client name] (アプリケーションクライアント名) を入力します。

f. アプリクライアントの詳細設定 を展開します。認証フロー のリストALLOW_USER_PASSWORD_AUTHに を追加します。

g. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

9. 確認と作成 画面で選択内容を確認し、必要に応じて選択内容を変更します。ユーザープールの設定に問題がなければ、ユーザープールの作成を選択して続行します。

10. ユーザープール ページで、新しいユーザープールを選択します。

11. ユーザープールの概要 で、ユーザープール ID を書き留めます。この文字列は、サンプルアプリケーションを作成するときに指定します。

12. アプリ統合 タブを選択し、アプリクライアントと分析 セクションを見つけます。新しいアプリクライアントを選択します。クライアント ID を書き留めます。

関連リソース

- Amazon Cognito user pools
- ユーザープール認証フロー
- ユーザープールでのトークンの使用

アプリケーションの作成

Android アプリの例を作成するには

1. Android スタジオとコマンドラインツールをインストールします。
2. Android Studio で、Flutter プラグイン をインストールします。
3. このサンプルアプリケーション の cognito_flutter_mobile_app ディレクトリの内容から新しい Android Studio プロジェクトを作成します。

   - assets/config.json と を編集<<YOUR USER POOL ID>>し、以前に作成したユーザープールとアプリクライアントの IDs << YOUR CLIENT ID>>に置き換えます。 ???

4. Flutter をインストールします。

   a. PATH 変数に Flutter を追加します。

b. 次のコマンドでライセンスを受け入れます。

```
flutter doctor --android-licenses
```

c. Flutter 環境を確認し、不足しているコンポーネントをインストールします。

```
flutter doctor
```

- コンポーネントが見つからない場合は、`flutter doctor -v`を実行して問題を解決する方法を確認してください。

d. 新しい Flutter プロジェクトの ディレクトリに移動し、依存関係をインストールします。

- `flutter pub add amazon_cognito_identity_dart_2` を実行します。

e. `flutter pub add flutter_secure_storage` を実行します。

5. 仮想 Android デバイスを作成します。

1. Android スタジオ GUI で、デバイスマネージャー を使用して新しいデバイスを作成します。
2. CLI で、 を実行しますflutter emulators --create --name android-device。

6. 仮想 Android デバイスを起動します。

1. Android Studio GUI で、仮想デバイスの横にある開始 アイコンを選択します。
2. CLI で、 を実行しますflutter emulators --launch android-device。

7. 仮想デバイスでアプリを起動します。

1. Android Studio GUI で、デプロイ アイコンを選択します。
2. CLI で、 を実行しますflutter run。

8. Android Studio で実行中の仮想デバイスに移動します。
9. 有効な E メールアドレスで新しいユーザーをサインアップします。
10. E メールメッセージから確認コードを取得します。アプリケーションに確認コードを入力します。
11. ユーザー名とパスワードを使用してサインインします。

次のステップ

チュートリアルに従ってサンプルアプリケーションを完了したら、ユーザープールの実装範囲を広げることができます。追加のユーザープール を作成したり、他のアプリケーション のユーザープール機能をカスタマイズしたり、外部 ID プロバイダー を追加したりできます。Amazon Cognito ユーザープールを本番稼働用アプリケーションに配置するための移行を計画する際に、追加の例やチュートリアルを評価できます。

Amazon Cognito ユーザープールのその他の機能は以下のとおりです。

- 組み込みのサインインおよびサインアップウェブページのカスタマイズ
- ユーザープールに MFA を追加します
- ユーザープールにアドバンストセキュリティを追加する
- Lambda トリガーを使用したユーザープールワークフローのカスタマイズ
- Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用

Amazon Cognito の認証および承認モデルの概要については、「」を参照してくださいAmazon Cognito ユーザープールと ID プールでの認証の仕組み。

ユーザープール認証が成功 AWS のサービス した後に他の にアクセスするには、「」を参照してくださいサインイン後の ID プール AWS のサービス を使用した へのアクセス。

AWS Management Console とユーザープール SDKsの使用に加えて、 を使用してユーザープールを管理することもできますAWS Command Line Interface。

トピック

- 新しいユーザープールを作成する
- ホストされた UI でアプリクライアントを追加する
- ユーザープールにソーシャルサインインを追加する (オプション)
- SAML ID プロバイダーを使用したサインインをユーザープールに追加する (オプション)

新しいユーザープールを作成する

ユーザープールを使用すると、ユーザーは Amazon Cognito 経由でウェブまたはモバイルアプリに口グインできます。

新しいユーザープールを作成する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ユーザープールの作成ボタンを選択します。このオプションを表示するには、左側のナビゲーションペインからユーザープールを選択する必要がある場合があります。
3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。
4. 「サインインエクスペリエンスの設定」で、このユーザープールで使用する ID プロバイダー (IdPs) を選択できます。詳細については、「サードパーティー経由のユーザープールへのサインインの追加」を参照してください。

   a. 認証プロバイダー で、プロバイダータイプ で、Cognito ユーザープールのみが選択されていることを確認します。

   b. Cognito ユーザープールのサインインオプション で、ユーザー名 を選択します。追加のユーザー名要件 を選択しないでください。

   c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。
5. セキュリティ要件の設定 では、パスワードポリシー、多要素認証 (MFA) 要件、ユーザーアカウント復旧オプションを選択できます。詳細については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。

   a. パスワードポリシー の場合、パスワードポリシーモードが Cognito のデフォルト に設定されていることを確認します。

   b. 多要素認証 で、MFA 強制 にオプションの MFA を選択します。

   c. MFA メソッド で、認証アプリケーション と SMS メッセージ を選択します。

   d. ユーザーアカウント復旧 で、セルフサービスアカウントの復旧を有効にする が選択され、ユーザーアカウント復旧メッセージの配信方法が E メールのみ に設定されていることを確認します。

   e. 他のすべてのオプションをデフォルトのままにして、次へを選択します。
6. 「サインアップエクスペリエンスの設定」では、新規ユーザーとしてサインアップするときに新規ユーザーが ID を検証する方法と、ユーザーのサインアップフローで必須またはオプションとなる属性を決定できます。詳細については、「ユーザープール内のユーザーを管理する」を参照してください。

a. 自己登録を有効にするが選択されていることを確認します。この設定により、ユーザープールが開き、インターネット上の全員からサインアップできます。これはサンプルアプリケーションを目的としていますが、この設定は本番環境では慎重に適用してください。

b. Cognito が支援する検証および確認 で、Cognito がメッセージを自動的に送信して検証および確認チェックボックスが選択されていることを確認します。

c. 検証する属性が E メールメッセージの送信に設定されていることを確認し、E メールアドレス を確認します。

d. 属性の変更の検証で、デフォルトのオプションが選択されていることを確認します。更新が保留中の場合は元の属性値を保持し、更新が保留中の場合はアクティブな属性値が E メールアドレス に設定されます。

e. 「必須属性」で、以前の選択に基づく必須属性に E メール が表示されていることを確認します。

> **Important**
>
> このサンプルアプリケーションでは、ユーザープールで phone_number を必須属性として設定しないでください。phone_number が必須属性として表示される場合は、前の選択肢を確認して更新します。
>
> - オプションの MFA、ユーザーアカウント復旧メッセージの配信方法の E メールのみ
> - E メールメッセージを送信し、検証する属性の E メールアドレスを確認する

f. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

7. メッセージ配信の設定 では、サインアップ、アカウント確認、MFA、およびアカウント回復のために E メールおよび SMS メッセージをユーザーに送信するように、Amazon Simple Email Service および Amazon Simple Notification Service との統合を設定できます。詳細については、「Amazon Cognito ユーザープールの E メール設定」および「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。

   a. E メールプロバイダー で、Cognito で E メールを送信 を選択し、Amazon Cognito が提供するデフォルトの E メール送信者を使用します。E メール量が少ない場合のこの設定は、アプリケーションのテストには十分です。Amazon Simple Email Service (Amazon SES) で E メールアドレスを検証し、Amazon SES で E メールを送信 を選択すると、 を返すことができます。

b. SMS の場合、新しい IAM ロールの作成を選択し、IAM ロール名を入力します。これにより、SMS メッセージを送信するためのアクセス許可を Amazon Cognito に付与するロールが作成されます。

c. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

8. 「アプリの統合」では、ユーザープールに名前を付け、ホストされた UI を設定し、アプリクライアントを作成できます。詳細については、「ホストされた UI でアプリクライアントを追加する」を参照してください。サンプルアプリケーションはホストされた UI を使用しません。

a. ユーザープール名 に、ユーザープール名 を入力します。

b. Cognito でホストされた UI の使用 は選択しないでください。

c. 初期アプリケーションクライアント で、アプリケーションタイプがパブリッククライアント に設定されていることを確認します。

d. クライアントシークレット で、クライアントシークレットを生成しない が選択されていることを確認します。

e. [App client name] (アプリケーションクライアント名) を入力します。

f. アプリクライアントの詳細設定 を展開します。認証フローのリストALLOW_USER_PASSWORD_AUTHに を追加します。

g. 他のすべてのオプションをデフォルトのままにして、次へを選択します。

9. 「確認と作成」画面で選択内容を確認し、必要に応じて選択内容を変更します。ユーザープールの設定に問題がなければ、ユーザープールの作成を選択して続行します。

10. ユーザープール ページで、新しいユーザープールを選択します。

11. ユーザープールの概要 で、ユーザープール ID を書き留めます。この文字列は、サンプルアプリケーションを作成するときに指定します。

12. アプリ統合 タブを選択し、アプリクライアントと分析 セクションを見つけます。新しいアプリクライアントを選択します。クライアント ID を書き留めます。

ユーザープールを作成する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。

2. [User Pools] (ユーザープール) を選択します。

3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。

4. [Configure sign-in experience] (サインインエクスリエンスを構成する) で、このユーザープールで使用するフェデレーションプロバイダーを選択します。詳細については、「サードパーティー経由のユーザープールへのサインインの追加」を参照してください。
5. [Configure security requirements] (セキュリティ要件の設定) で、パスワードポリシー、多要素認証 (MFA) 要件、ユーザーアカウントの回復オプションを選択します。詳細については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。
6. [Configure sign-up experience] (サインアップエクスペリエンスの設定) で、新規ユーザーがサインアップ時に自分の ID を確認する方法と、ユーザーのサインアップフローで必須またはオプションにする属性を決定します。詳細については、「ユーザープール内のユーザーを管理する」を参照してください。

   > **Important**
   >
   > ユーザープールでユーザーサインアップを有効にすると、インターネット上のすべてのユーザーがアカウントにサインアップしてアプリケーションにサインインできるようになります。アプリケーションをパブリックサインアップで開く場合を除き、ユーザープールで自己登録を有効にしないでください。この設定を変更するには、ユーザープールコンソールのサインアップエクスペリエンスタブでセルフサービスサインアップを更新するか、 CreateUserPool または UpdateUserPool API リクエスト AllowAdminCreateUserOnlyで の値を更新します。
   > ユーザープールに設定できるセキュリティ機能については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。

7. [Configure message delivery] (メッセージ配信の設定) で、Amazon Simple Email Service および Amazon Simple Notification Service との統合を構成して、サインアップ、アカウント確認、MFA、およびアカウントの復元用に E メールおよび SMS メッセージをユーザーに送信します。詳細については、「Amazon Cognito ユーザープールの E メール設定」および「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。
8. [Integrate your app] (アプリケーションの統合) で、ユーザープールに名前を付け、ホストされた UI を設定し、アプリケーションクライアントを作成します。詳細については、「ホストされた UI でアプリクライアントを追加する」を参照してください。
9. 「確認と作成」画面で選択内容を確認し、必要に応じて選択内容を変更します。ユーザープールの設定に問題がなければ、ユーザープールの作成を選択して続行します。

関連リソース

ユーザープールの詳細については、「Amazon Cognito user pools」を参照してください。

ユーザープール認証フロー 「」および「」も参照してくださいユーザープールでのトークンの使用。

ホストされた UI でアプリクライアントを追加する

ユーザープールを作成したら、ホストされた UI の組み込みウェブページを起動するアプリケーションのアプリケーションクライアントを作成できます。ホストされた UI では、ユーザーは次のことができます。

- ユーザープロファイルにサインアップします。
- サードパーティーの ID プロバイダーでサインインします。
- 多要素認証の有無にかかわらずサインインします。
- パスワードをリセットします。

ホストされた UI サインイン用のアプリクライアントを作成するには

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。新しいユーザープールを作成すると、ウィザード中にアプリケーションクライアントを設定し、ホストされた UI を構成するように求められます。
4. [App integration] (アプリケーションの統合) タブをユーザープールに設定します。
5. [ドメイン] の横で、[アクション] を選択し、[カスタムドメインの作成] または [Amazon Cognito ドメインの作成] のどちらかを選択します。ユーザープールドメインを既に設定している場合は、新しいカスタムドメインを作成する前に、[Amazon Cognitoドメインの削除] または [カスタムドメインの削除] を選択します。
6. Amazon Cognito ドメインで使用するための使用可能なドメインプレフィックスを入力します。[カスタムドメイン] のセットアップに関する詳細については、「ホストされた UI への独自のドメインの使用」を参照してください。
7. [作成] を選択します。

8. 同じユーザープールの [App integration] (アプリケーションの統合) タブに移動し、[App clients] (アプリケーションクライアント) を検索します。[Create app client] (アプリケーションクライアントの作成) を選択します。
9. [Application type] (アプリケーション) を選択します。選択した内容に基づいて、いくつかの推奨設定が提供されます。ホストされた UI を使用するアプリケーションは、[Public client] (パブリッククライアント) です。
10. [App client name] (アプリケーションクライアント名) を入力します。
11. この演習では、[Don't generate client secret] (クライアントシークレットを生成しない) を選択します。クライアントシークレットは、一元化されたアプリケーションからユーザーを認証する機密アプリケーションによって使用されます。この演習では、ホストされた UI サインインページをユーザーに提示し、クライアントシークレットは必要ありません。
12. アプリケーションで許可する認証フローを選択します。`USER_SRP_AUTH` が選択されていることを確認します。
13. 必要に応じて、[token expiration] (トークン有効期限)、[Advanced security configuration] (高度なセキュリティ設定)、および[Attribute read and write permissions] (属性の読み取りと書き込みのアクセス許可) をカスタマイズします。詳細については、「アプリケーションクライアントのセッティングの構成」を参照してください。
14. アプリケーションクライアントの [Add a callback URL] (コールバック URL を追加する) 。これは、ホストされた UI 認証の後に指示される場所です。アプリにサインアウトを実装できるようになるまで、許可されたサインアウト URL を追加する必要はありません。

    iOS または Android アプリの場合は、`myapp://` などのコールバック URL を使用できます。
15. アプリケーションクライアントの [ID providers] (ID プロバイダー) を選択します。少なくとも、[Amazon Cognito ユーザープール] をプロバイダーとして有効にします。

    > **Note**
    >
    > Facebook、Amazon、Google、Apple などの外部 ID プロバイダー (IdPs) 、および OpenID Connect (OIDC) や SAML 経由でサインインするには IdPs、まず、「サードパーティー によるユーザープールサインインの追加」に示すように設定します。次に、アプリクライアント設定ページに戻り、有効にします。

16. [OAuth 2.0 Grant Types] (OAuth 2.0 グラントタイプ) を選択します。[Authorization code grant (認証コードの付与)] を選択して、ユーザープールトークンと交換される認証コードが返されます。トークンはエンドユーザーに直接公開されないため、侵害される可能性は低くなります。ただし、ユーザープールトークンの認証コードを交換するために、バックエンドでカスタムアプリ

ケーションが必要です。セキュリティ上の理由から、モバイルアプリ用の認証コード付与フローとコード交換用証明キー (PKCE) を併用することをお勧めします。

[Implicit grant] (暗黙の付与)を選択して、Amazon Cognito からユーザープールの JSON Web トークン (JWT) が返されるようにします。このフローは、トークンの認証コードを交換できるバックエンドがない場合に使用でき、トークンのデバッグにも役立ちます。

> **Note**
>
> [Authorization code grant (認証コードの付与)] と [Implicit code grant (暗黙的コードの付与)] の両方を有効にして、必要に応じて各権限を使用することができます。
> アプリがユーザーの代わりではなく自身の代わりとしてアクセストークンをリクエストする必要がある場合にのみ、[クライアント認証情報] を選択します。

17. 特に除外する必要がない限り、すべての [OpenID Connect scopes] (OpenID Connect スコープ) を選択します。
18. 設定したカスタムスコープを選択します。カスタムスコープは、通常、機密クライアントで使用されます。
19. [作成] を選択します。

サインインページを表示するには

アプリクライアントページ から、ホストされた UI を表示 を選択して、アプリクライアント ID、スコープ、権限、コールバック URL パラメータがあらかじめ入力されているサインインページに新しいブラウザタブを開きます。

ホストされた UI サインインウェブページは、以下の URL を使用して手動で表示できます。`response_type` を書き留めます。この場合は、認証コード付与の response_type=code です。

```
https://your_domain/login?
response_type=code&client_id=your_app_client_id&redirect_uri=your_callback_url
```

response_type=token である暗黙のコード付与の次の URL を使用して、ホストされた UI サインインウェブページを表示できます。サインインが正常に行われると、Amazon Cognito がユーザープールトークンをウェブブラウザのアドレスバーに返します。

```
https://your_domain/login?
response_type=token&client_id=your_app_client_id&redirect_uri=your_callback_url
```

JSON ウェブトークン (JWT) ID トークンは、レスポンスの #idtoken= パラメータの後にあります。

ここでは、暗黙的な付与リクエストからのレスポンスの例を示します。ID トークン文字列はこれ以上に長くなります。

```
https://www.example.com/
#id_token=123456789tokens123456789&expires_in=3600&token_type=Bearer
```

Amazon Cognito ユーザープールトークンは、RS256 アルゴリズムを使用して署名されます。を使用して、ユーザープールトークンをデコードして検証できます AWS Lambda。詳細については、 AWS GitHub ウェブサイトの「Amazon Cognito JWT トークンをデコードして検証する」を参照してください。

ドメインは [ドメイン名] ページに表示されています。アプリクライアント ID およびコールバック URL は [全般設定] ページに表示されています。コンソールで行った変更がすぐに表示されない場合は、数分待ってからブラウザを更新します。

ユーザープールにソーシャルサインインを追加する (オプション)

アプリユーザーが Facebook、Google、Amazon、Apple などのソーシャル ID プロバイダー (IdP) を介してサインインするようにできます。ユーザーが直接サインインしても、サードパーティーを介してサインインしても、すべてのユーザーにはユーザープールにプロファイルがあります。ソーシャルサインイン ID プロバイダーを通じてサインインを追加しない場合は、この手順を省略してください。

ソーシャル IdP に登録する

Amazon Cognito でソーシャル IdP を作成する前に、アプリケーションをソーシャル IdP に登録して、クライアント ID とクライアントシークレットを取得する必要があります。

アプリケーションを Facebook に登録する

1. Facebook の開発者アカウントを作成します。
2. Facebook 認証情報を使用してサインインします。

3. [My Apps] (マイアプリ) メニューから、[新しいアプリを作成] (新しいアプリを作成) を選択します。

   既存の Facebook アプリがない場合は、別のオプションが表示されます。[Create App (アプリの作成)] を選択します。
4. [アプリの作成] ページで、アプリのユースケースを選択し、[次へ] を選択します。
5. Facebook アプリケーションに名前を入力して、[アプリの作成] を選択します。
6. 左のナビゲーションバーで、[設定]、[基本] の順に選択します。
7. [App ID] (アプリ ID)] と [App Secret (アプリシークレット) を書き留めます。これらは次のセクションで使用します。
8. ページの下部で、[+ プラットフォームを追加] を選択します。
9. プラットフォームの選択画面で、プラットフォームを選択し、次へを選択します。
10. [変更を保存] を選択します。
11. [App Domains] (アプリケーションドメイン) で、ユーザープールのドメインを入力します。

    ```
    https://your_user_pool_domain
    ```
12. [変更を保存] を選択します。
13. ナビゲーションバーから製品 を選択し、Facebook ログイン から設定 を選択します。
14. [Facebook でログイン] の [設定] メニューから [設定] を選択します。

    [Valid OAuth Redirect URIs] (有効な OAuth リダイレクト URI) にリダイレクト URL を入力します。リダイレクト URL は、 `/oauth2/idpresponse`エンドポイントを持つユーザープールドメインで構成されます。

    ```
    https://your_user_pool_domain/oauth2/idpresponse
    ```
15. [変更を保存] を選択します。

アプリケーションを Amazon に登録する

1. Amazon の開発者アカウントを作成します。
2. Amazon 認証情報を使用してサインインします。
3. Amazon クライアント ID およびクライアントシークレットを受け取るには、Amazon セキュリティプロファイルを作成する必要があります。

ページ上部のナビゲーションバーからアプリとサービスを選択し、Login with Amazon を選択します。

4. [Create a New Security Profile (新しいセキュリティプロファイルの作成)] を選択します。
5. [Security Profile Name] (セキュリティプロファイル名)、[Security Profile Description] (セキュリティプロファイルの説明)、[Consent Privacy Notice URL] (プライバシー規約 URL の同意) に入力します。
6. [Save] を選択します。
7. [クライアント ID] および [クライアントシークレット] を選択して、クライアント ID およびシークレットを表示します。これらは次のセクションで使用します。
8. 歯車アイコンにマウスカーソルを合わせ、[Web Settings] (ウェブ設定)、[Edit] (編集) の順に選択します。
9. [Allowed Origins] (許可されたオリジン) にユーザープールのドメインを入力します。

```
https://<your-user-pool-domain>
```

10. `/oauth2/idpresponse` エンドポイントを使用するユーザープールドメインを [Allowed Return URLs] (許可されたリターン URL) に入力します。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

11. [Save] を選択します。

アプリケーションを Google に登録する

Google Cloud プラットフォームの OAuth 2.0 の詳細については、「Google Workspace for Developers」(Google Workspace デベロッパーガイド) ドキュメントの「Learn about authentication & authorization」(認証と認可について学ぶ) を参照してください。

1. Google の開発者アカウントを作成します。
2. Google Cloud Platform コンソールにサインインします。
3. 上部のナビゲーションバーから、[Select a project] (プロジェクトの選択) を選択します。Google プラットフォームにプロジェクトが既にある場合は、このメニューには代わりにデフォルトのプロジェクトが表示されます。
4. [NEW PROJECT] (新しいプロジェクト) を選択します。
5. 製品の名前を入力し、[CREATE] (作成) を選択します。

6. 左側のナビゲーションバーで、APIsとサービス を選択し、認証同意画面 を選択します。
7. アプリ情報、アプリドメイン 、承認済みドメイン 、デベロッパーの連絡先情報を入力します。承認されたドメインには、 amazoncognito.comとカスタムドメインのルートが含まれている必要があります。例: example.com。[SAVE AND CONTINUE] (保存して続行) を選択します。
8. 1. スコープ で、スコープ を追加または削除を選択し、少なくとも次の OAuth スコープを選択します。

    1. .../auth/userinfo.email
    2. .../auth/userinfo.profile
    3. openid
9. [Test users] (テストユーザー) で、[Add users] (ユーザーの追加) を選択します。E メールアドレスとその他の承認されたテストユーザーを入力し、SAVE AND CONTINUE を選択します。
10. 左のナビゲーションバーを再度展開し、APIsとサービス を選択し、認証情報 を選択します。
11. 「認証情報の作成」を選択し、「OAuth クライアント ID」を選択します。
12. [Application type] (アプリケーションタイプ) を選択し、クライアントに [Name] (名前) を入力します。
13. 承認オ JavaScript リジン で、ADD URI を選択します。ユーザープールのドメインを入力します。

    ```
    https://<your-user-pool-domain>
    ```
14. [Authorized redirect URIs] (承認済みのリダイレクト URI) で、[ADD URI] (URI の追加) を選択します。ユーザープールのドメインの /oauth2/idpresponse エンドポイントへのパスを入力します。

    ```
    https://<your-user-pool-domain>/oauth2/idpresponse
    ```
15. [CREATE] (作成) を選択します。
16. Google が [クライアント ID] と [クライアントシークレット] の下に表示する値を安全に保存します。Google IdP を追加するときに、Amazon Cognito にこれらの値を指定します。

アプリケーションを Apple に登録する

「Apple でサインイン」を設定する方法についての詳細は、Apple 開発者ドキュメントの「Configuring Your Environment for Sign in with Apple」(Apple でサインインするための環境を設定する) を参照してください。

1. Apple の開発者アカウントを作成します。
2. Apple 認証情報を使用してサインインします。
3. 左のナビゲーションバーで、[Certificates, Identifiers & Profiles] (証明書、ID & プロファイル) を選択します。
4. 左のナビゲーションペインで、[Identifiers] (識別子) を選択します。
5. [Identifiers] (識別子) ページで、[+] アイコンを選択します。
6. [Register a New Identifier] (新しい識別子の登録) ページで、[App IDs] (アプリ ID)、[Continue] (続行) の順に選択します。
7. タイプを選択 ページで、アプリ を選択し、続行 を選択します。
8. [Register an App ID] ページで、次の操作を行います。
    1. [Description] (説明) で、説明を入力します。
    2. [App ID Prefix] (アプリ ID プレフィックス) に、[Bundle ID] (バンドル ID) を入力します。[App ID Prefix] (アプリケーション ID プレフィックス) にある値を書き留めておきます。この値は、ステップ 2: ユーザープールにソーシャル IdP を追加する で Apple を ID プロバイダーとして選択した後で使用します。
    3. [Capabilities] (機能) で、[Sign In with Apple] (Apple でサインイン) を選択してから [Edit] (編集) を選択します。
    4. [Sign in with Apple: App ID Configuration] (Apple でサインイン: アプリ ID の設定) ページで、アプリをプライマリとして設定するか、他のアプリ ID とグループ化するかを選択し、次に [Save] (保存) を選択します。
    5. [Continue] を選択します。
9. [Confirm your App ID] ページで、[登録] を選択します。
10. [Identifiers] (識別子) ページで、[+] アイコンを選択します。
11. [Register a New Identifier] (新しい識別子の登録) ページで、[Services IDs] (サービス ID)、[Continue] (続行) の順に選択します。
12. [Register a Services ID] ページで、次の操作を行います。
    1. [Description] (説明) で、説明を入力します。

2. [Identifier] (識別子) で、識別子を入力します。このサービス ID をメモしておきます。この値は、 で Apple を ID プロバイダーとして選択した後に必要になりますステップ 2: ユーザープールにソーシャル IdP を追加する。
3. [Continue (続行)] を選択し、[Register (登録)] を選択します。

13. 識別子ページから作成したサービス ID を選択します。

1. [Sign In with Apple] (Apple でサインイン) を選択後、[Configure] (設定) を選択します。
2. [Web Authentication Configuration] (ウェブ認証の設定) ページで、[Primary App ID] (プライマリアプリ ID) として前に作成したアプリ ID を選択します。
3. [Website URLs] (ウェブサイトの URL) の横の [+] アイコンを選択します。
4. [Domains and subdomains] (ドメインとサブドメイン) で、`https://` プレフィックスなしでユーザープールのドメインを入力します。

```
<your-user-pool-domain>
```

5. [Return URLs] (URL を返す) で、ユーザープールのドメインの `/oauth2/idpresponse` エンドポイントへのパスを入力します。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

6. Next を選択し、Done を選択します。ドメインを検証する必要はありません。
7. [Continue] (続行) を選択し、次に [Save] (保存) を選択します。

14. 左のナビゲーションペインで [Keys] (キー) を選択します。
15. [Keys (キー)] ページで、[+] アイコンを選択します。
16. [Register a New Key] ページで、次の操作を行います。

1. [Key Name] (キー名) に、キー名を入力します。
2. [Sign In with Apple] (Apple でサインイン) を選択後、[Configure] (設定) を選択します。
3. 「キーの設定」ページで、以前にプライマリアプリ ID として作成したアプリ ID を選択します。[保存] を選択します。
4. [Continue] (続行) を選択し、[Register] (登録) を選択します。

17. 「キーのダウンロード」ページで、「ダウンロード」を選択してプライベートキーをダウンロードし、表示されるキー ID をメモして、「完了」を選択します。このプライベートキーと、このページに表示されている [Key ID] (キー ID) の値は、ステップ 2: ユーザープールにソーシャル IdP を追加する で Apple を ID プロバイダーとして選択した後で必要になります。

ユーザープールにソーシャル IdP を追加する

このセクションでは、前のセクションで取得したクライアント ID およびクライアントシークレットを使用してユーザープールにソーシャル IdP を設定します。

を使用してユーザープールのソーシャル ID プロバイダーを設定するには AWS Management Console

1. Amazon Cognito コンソールに移動します。 AWS 認証情報の入力を求められる場合があります。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を検索し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. ソーシャル ID プロバイダー ([Facebook]、[Google]、[Login with Amazon]、[Sign in with Apple]) を選択します。
6. ソーシャル ID プロバイダーの選択に基づいて、次のステップから選択します。

   - Google と Login with Amazon – 前のセクションで生成されたアプリクライアント ID とアプリクライアントシークレットを入力します。
   - Facebook – 前のセクションで生成されたアプリクライアント ID とアプリクライアントシークレットを入力し、API バージョン (バージョン 2.12 など) を選択します。可能な限り最新のバージョンを選択することをお勧めします。各 Facebook API にはライフサイクルと非推奨日があります。Facebook のスコープと属性は API バージョンによって異なる場合があります。Facebook でソーシャル ID ログインをテストして、フェデレーションが意図したとおりに機能することを確認することをお勧めします。
   - Sign in with Apple – 前のセクションで生成されたサービス ID、チーム ID、キー ID、プライベートキーを入力します。
7. 使用する承認済みスコープの名前を入力します。スコープは、アプリでアクセスするユーザー属性 (`name` や `email` など) を定義します。Facebook の場合は、コンマで区切る必要があります。Google および Login with Amazon の場合は、スペースで区切って指定します。Sign in with Apple の場合は、アクセスするスコープのチェックボックスをオンにします。

ソーシャル ID プロバイダー	スコープ例
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Apple でのサインイン	`email name`

アプリケーションユーザーは、これらの属性をアプリケーションに提供すること
に同意するよう求められます。ソーシャルプロバイダーのスコープの詳細について
は、Google、Facebook、Login with Amazon、および Sign in with Apple のドキュメントを参照
してください。

以下は、Sign in with Apple の場合にスコープが返らない可能性があるユーザーシナリオです。

- エンドユーザーが Apple のサインインページを離れるとエラーが発生します (これらは Amazon Cognito の内部障害または開発者が記述した内容が原因である可能性があります)。
- サービス ID 識別子は、ユーザープールやその他の認証サービス全体で使用されます。
- ユーザーがサインインした後に、デベロッパーがスコープを追加している。ユーザーは認証時およびトークンの更新時にのみ、新しい情報を取得する。
- デベロッパーはユーザーを削除し、ユーザーは Apple ID プロファイルからアプリを削除せずに再度サインインします。

8. ID プロバイダーからユーザープールに属性をマッピングします。詳細については、「マッピングについて知っておくべきこと」を参照してください
9. [Create] (作成) を選択します。
10. [App client integration] (アプリケーションクライアント統合) タブから、[App clients] (アプリケーションクライアント) のリストより 1 つ選択して、[Edit hosted UI settings] (ホストされた UI 設定の編集) をクリックします。[Identity providers] (ID プロバイダー) で、新しいソーシャル ID プロバイダーをアプリケーションクライアントに追加します。
11. [変更を保存] を選択します。

ソーシャル IdP の設定をテストする

前の 2 つのセクションの要素を使用してログイン URL を作成できます。この URL を使用してソーシャル IdP の設定をテストします。

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

ドメインは、コンソールにあるユーザープールの [ドメイン名] ページで確認できます。クライアント ID は [アプリクライアントの設定] ページにあります。redirect_uri パラメータのコールバック URL を使用します。これは、ユーザーが認証に成功した後でリダイレクトされるページの URL です。

> **Note**
>
> Amazon Cognito は 5 分以内に完了しない認証リクエストをキャンセルし、ホストされた UI にユーザーをリダイレクトします。ページには、`Something went wrong` というエラーメッセージが表示されます。

SAML ID プロバイダーを使用したサインインをユーザープールに追加する (オプション)

アプリユーザーが SAML ID プロバイダー (IdP) を通じてサインインするようにできます。ユーザーが直接サインインしても、サードパーティーを介してサインインしても、すべてのユーザーにはユーザープールにプロファイルがあります。SAML ID プロバイダーを通じてサインインを追加しない場合は、この手順を省略してください。

詳細については、「ユーザープールでの SAML ID プロバイダーの使用」を参照してください。

SAML ID プロバイダーを更新し、ユーザープールを設定する必要があります。SAML 2.0 ID プロバイダーの証明書利用者またはアプリケーションとしてユーザープールを追加する方法については、SAML ID プロバイダーのドキュメントを参照してください。

また、SAML ID プロバイダーにアサーションコンシューマーサービス (ACS) エンドポイントを提供する必要があります。SAML ID プロバイダーの SAML 2.0 POST バインド用に、ユーザープールドメインで次のエンドポイントを設定します。ユーザープールドメインの詳細については、「」を参照してくださいユーザープールのドメインを設定する。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://Your custom domain/saml2/idpresponse
```

ドメインプレフィックスとユーザープールのリージョン値は、Amazon Cognito コンソールのドメイン名タブにあります。

一部の SAML ID プロバイダーでは、オーディエンス URI または SP エンティティ ID urnとも呼ばれるサービスプロバイダー (SP) を次の形式で指定する必要があります。

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

ユーザープール ID は、Amazon Cognito コンソールの [General settings] (全般設定) タブにあります。

また、SAML ID プロバイダーを設定して、ユーザープールに必要なすべての属性の属性値を提供する必要もあります。通常、email はユーザープールの必須属性です。その場合、SAML ID プロバイダーは、SAML アサーションの email 値 (クレーム) を指定する必要があります。

Amazon Cognito ユーザープールはバインディング後のエンドポイントで SAML 2.0 フェデレーションをサポートします。これにより、ユーザープールはユーザーエージェントを通じて ID プロバイダーから直接 SAML レスポンスを受信するため、アプリが SAML アサーションレスポンスを取得または解析する必要がなくなります。

ユーザープールに SAML 2.0 ID プロバイダーを設定する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を検索し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. [SAML] ソーシャル ID プロバイダーを選択します。

6. カンマで区切られた [Identifiers] (識別子) を入力します。識別子は Amazon Cognito に、ユーザーがサインインしたときに入力した E メールアドレスを確認するように指示します。次に、ドメインに対応するプロバイダーに指示します。

7. ユーザーがログアウトしたときに、Amazon Cognito が署名されたサインアウト要求をプロバイダーに送信するためには、[Add sign-out flow] (サインアウトフローの追加) を選択します。SAML 2.0 ID プロバイダーを設定して、ホストされた UI を構成するときに作成される `https://<your Amazon Cognito domain>/saml2/logout` エンドポイントにサインアウト応答を送信する必要があります。`saml2/logout` エンドポイントは POST バインディングを使用します。

   > **Note**
   >
   > このオプションが選択されていて、SAML ID プロバイダーが署名付きログアウトリクエストを期待する場合は、Amazon Cognito が提供する署名証明書を SAML IdP で設定する必要もあります。
   > SAML IdP は署名されたログアウトリクエストを処理し、Amazon Cognito セッションからユーザーをログアウトします。

8. [Metadata document source] (メタデータドキュメントソース) を選択します。ID プロバイダーがパブリック URL で SAML メタデータを提供する場合は、[Metadata document URL] (メタデータドキュメント URL) を選択してそのパブリック URL を入力できます。それ以外の場合は、[Upload metadata document] (メタデータドキュメントをアップロード) を選択し、プロバイダーから以前ダウンロードしたメタデータファイルを選択します。

   > **Note**
   >
   > プロバイダーにパブリックエンドポイントがある場合は、ファイルをアップロードするのではなく、メタデータドキュメント URL を入力することをお勧めします。これにより、Amazon Cognito はメタデータを自動的に更新できます。通常、メタデータの更新は 6 時間ごとまたはメタデータの有効期限が切れる前のいずれか早いタイミングで発生します。

9. [Map attributes between your SAML provider and your appS] (AML プロバイダーとアプリケーション間で属性をマッピングする) をクリックして、SAML プロバイダー属性をユーザープールのユーザープロファイルにマッピングします。ユーザープールの必須属性を属性マップに含めます。

たとえば、[User pool attribute] (ユーザープール属性) emailを選択して、ID プロバイダーからの SAML アサーションに表示される SAML 属性名を入力します。ID プロバイダーは、参考として SAML アサーションのサンプルを提供する場合があります。ID プロバイダーの中には、email などのような単純な名前を使用している ID プロバイダーもあります。次の例のように、URL 形式の属性名を使用するものもあります。

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

10. [作成] を選択します。

Amazon Cognito ID プールの開始方法

Amazon Cognito ID プールを使用すると、一意のアイデンティティを作成し、ユーザーに許可を割り当てることができます。ID プールには以下を含めることができます。

- Amazon Cognito ユーザープール内のユーザー
- Facebook、Google、Apple、または SAML ID プロバイダーなどの外部 ID プロバイダーを使って認証するユーザー。
- 独自の既存の認証プロセスによって認証されたユーザー

ID プールを使用すると、他の に直接アクセス AWS のサービス したり、Amazon API Gateway を介してリソースにアクセスしたりするために定義したアクセス許可を持つ一時的な AWS 認証情報を取得できます。

トピック

Amazon Cognito で ID プールを作成する

ID プールは、Amazon Cognito コンソール経由で、または、 AWS Command Line Interface (CLI) か Amazon Cognito API を使用して作成することができます。

コンソールで新しい ID プールを作成する

1. Amazon Cognito コンソールにサインインし、アイデンティティプールを選択します。
2. [ID プールを作成] を選択します。
3. [ID プールの信頼を設定] で、アイデンティティプールを認証アクセス、ゲストアクセス、またはその両方に設定することを選択します。

    - [認証アクセス] を選択した場合、アイデンティティプールの認証済み ID のソースとして設定する ID タイプを 1 つ以上選択します。[カスタムデベロッパープロバイダー] を設定した

場合、アイデンティティプールの作成後にそのプロバイダーを変更したり削除したりすることはできません。

4. [許可を設定] で、アイデンティティプール内の認証済みユーザーまたはゲストユーザーのデフォルトの IAM ロールを選択します。

   a. Amazon Cognito に、基本的な権限とアイデンティティプールとの信頼関係を持つ新しいロールを作成する場合は、[新しい IAM ロールを作成] を選択します。新しいロールを識別するための IAM ロール名を入力します (たとえば `myidentitypool_authenticatedrole`)。[ポリシードキュメントを表示] を選択して、Amazon Cognito が新しい IAM ロールに割り当てるアクセス権限を確認します。

   b. 使用する にロールが既にある場合は、既存の IAM ロール AWS アカウント の使用を選択できます。`cognito-identity.amazonaws.com` を含めるように IAM ロールの信頼ポリシーを設定する必要があります。リクエストが特定のアイデンティティプール内の認証されたユーザーから送信されたという証拠を提示した場合にのみ、Amazon Cognito がロールを引き継ぐことを許可するようにロールの信頼ポリシーを設定します。詳細については、「ロールの信頼とアクセス権限」を参照してください。

5. Connect ID プロバイダー で、「ID プールの信頼を設定する」で選択した ID プロバイダー (IdPs) の詳細を入力します。OAuth アプリケーションクライアント情報の提供、Amazon Cognito ユーザープールの選択、IAM IdP の選択、またはディベロッパープロバイダーのカスタム識別子の入力を求められる場合があります。

   a. 各 IdP のロール設定を選択します。その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。Amazon Cognito ユーザープール IdP では、トークンに preferred_role を含むロールを選択することもできます。`cognito:preferred_role` クレームの詳細については、「グループへの優先順位の値の割り当て」を参照してください。

      i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

      ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

   b. アクセスコントロールの属性は、IdP ごとに個別に設定できます。アクセスコントロールの属性は、Amazon Cognito がユーザーの一時セッションに適用するプリンシパルタグにユー

ザーのクレームをマッピングします。セッションに適用するタグに基づいてユーザーアクセスをフィルタリングするIAMポリシーを作成できます。

i. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

ii. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

iii. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

6. [プロパティの設定] の [ID プール名] に [名前] を入力します。
7. [基本 (クラシック) 認証] で、ベーシックフローを有効にするかどうかを選択します。基本的なフローをアクティブにすると、 に対して行ったロールの選択をバイパス IdPs して、AssumeRoleWithWebIdentity を直接呼び出すことができます。詳細については、「ID プール (フェデレーティッドアイデンティティ) の認証フロー」を参照してください。
8. アイデンティティプールにタグを適用する場合は、[タグ] で [タグの追加] を選択します。
9. [確認および作成] で、新しいアイデンティティプールに対して行った選択を確認します。[編集] を選択してウィザードに戻り、設定を変更します。終了したら、[ID プールの作成] を選択します。

SDKE の設定

Amazon Cognito ID プールを使用するには、 AWS Amplify、 AWS SDK for Java、または を設定します AWS SDK for .NET。詳細については、以下のトピックを参照してください。

- AWS SDK for Java デベロッパーガイドの SDK for のセットアップ JavaScript
- Amplify Dev Center の「Amplify のドキュメント」
- AWS SDK for .NET ディベロッパーガイド の「Amazon Cognito 認証情報プロバイダー」

ID プロバイダーを統合する

Amazon Cognito アイデンティティプール (フェデレーティッド ID) は、Amazon Cognito ユーザープール、フェデレーティッド ID プロバイダー (Amazon、Facebook、Google、および SAML ID プロバイダーなど) に加えて、認証されていない ID 経由でのユーザー認証をサポートしています。また、この機能は独自のバックエンド認証プロセスを通じてユーザーを認証し登録することができる、デベロッパーが認証したアイデンティティ (アイデンティティプール) をサポートしています。

独自のユーザーディレクトリを作成するための Amazon Cognito ユーザープールの使用に関する詳細については、「Amazon Cognito user pools」および「サインイン後の ID プール AWS のサービス を使用した へのアクセス」を参照してください。

外部 ID プロバイダーについては、ID プール外部 ID プロバイダー を参照してください。

独自のバックエンド認証プロセスの統合に関する詳細については、「デベロッパーが認証したアイデンティティ (アイデンティティプール)」を参照してください。

認証情報を取得する

Amazon Cognito ID プールは、ゲストユーザー (未認証) 、および認証してトークンを受け取ったユーザーに一時的な AWS 認証情報を提供します。これらの AWS 認証情報を使用して、アプリケーションは Amazon API Gateway AWS を介して AWS 内または外部にあるバックエンドに安全にアクセスできます。「認証情報の取得」を参照してください。

Amazon Cognito のガイド付きセットアップオプション

構造化されたガイド付きエクスペリエンスで Amazon Cognito の機能を評価することをお勧めします。ここでは、ユーザープールと ID プールでカスタマイズされたエクスペリエンスを提供する外部リソースをいくつか紹介します。

ワークショップを完了する

AWS ワークショップスタジオは、Amazon Cognito の機能の大部分をセットアップするワークショップをホストします。 Amazon Cognito これらの機能には、ユーザープール API、ユーザープールでホストされる UI、ID プール、セキュリティ設定が含まれます。

例からアプリケーションコードを追加する

このガイドのコード例の章には、ユーザープールと ID プールで使用できるアプリケーションコードが含まれています。コード例の章のユーザープールセクションには、個々の操作をカバーする短いスニペットと、さまざまなプログラミング言語のアプリケーション end-to-end 例の長い例があります。

でフルスタックアプリケーションを作成する AWS Amplify

AWS Amplify は、アプリケーションとユーザーインターフェイスを開発およびホストしたいデベロッパー AWS のサービス 向けの です。Amazon Cognito は Amplify の認証コンポーネントです。アプリケーションに認証を追加すると、Amplify は Amazon Cognito ユーザープールと ID プールリソースのデプロイを自動化できます。また、Amazon Cognito の認証と承認を、ウェブアプリケーションとモバイルアプリケーションに統合する も参照してください。

のその他の Amazon Cognito アプリケーションリソース GitHub

- .NET for Amazon Cognito を使用した認証フローの例
- Amazon Cognito パスワードレス認証
- PetStore Amazon Verified Permissions を使用した の例
- ABAC と ID プールを使用して AWS リソースにアクセスする React アプリのサンプル
- AWS CDK を使用した Amazon Cognito および API Gateway ベースのマシンツーマシン認証
- Amazon Cognito 、API Gateway、IAM を使用したきめ細かな認証の構築
- CloudFront authorization@edge

その他のワークショップ

- Amazon Cognito と を使用してパスワードレス認証を実装する WebAuthn
- Amazon Cognito ユーザープールを使用したマルチテナント SaaS ID
- Amazon Cognito JWT Deep Dive

Amazon Cognito の認証と承認を、ウェブアプリケーションとモバイルアプリケーションに統合する

アプリケーションを Amazon Cognito アプリケーションクライアントと統合すると、API オペレーションを呼び出してユーザーの認証と承認を行うことができます。を使用して Amazon Cognito AWS Amplifyをウェブおよびモバイルアプリと統合することをお勧めします。 AWS Amplify は、フロントエンドのウェブおよびモバイルデベロッパーが でフルスタックアプリケーションを簡単に構築、接続、ホストできる完全なソリューションであり AWS、ユースケースの進化 AWS のサービスに合わせて の幅広い範囲を柔軟に活用できます。Amplify Auth は、主に Amazon Cognito を使用して認証機能を構築します。

トピック

- による認証 AWS Amplify
- AWS SDK による認証
- Amazon Verified Permissions による承認

Amazon Cognito の一般的な実装では、ビジュアルツールと API を組み合わせて使用します。Amazon Cognito コンソールは、Amazon Cognito ユーザープールとアイデンティティプールを設定および管理するためのビジュアルインターフェイスです。ホストされた UI は、Amazon Cognito ユーザープールの迅速なテストとデプロイのための ready-to-use ウェブベースのサインインアプリケーションです。さらに、ほとんどの Amazon Cognito デプロイでは、ユーザープールやアイデンティティプールを操作するコードをアプリケーションに追加する必要があります。例えば、アプリケーションは、ホストされた UI を呼び出してユーザーのサインインを行い、次にアプリケーションコードからトークンエンドポイントを呼び出して、ユーザーの認証コードをトークンと交換する場合があります。この場合、アプリケーションはユーザーのトークンを解釈して保存し、認証と認可のための適切なコンテキストでトークンを提示する必要があります。Amplify は、これらのプロセス用の組み込み機能を備えたガイド付き統合ツールを追加します。

Amazon Cognito リソースをすべてコードで構築することもできます。独自のカスタムアプリケーションコードの使用を開始するには、AWS SDKを使用した Amazon Cognito のコード例を参照してください。OpenID Connect ID プロバイダーとして Amazon Cognito と統合するには、OpenID Connect デベロッパーツールを使用します。

Amazon Cognito の認証と認可を使用する前に、アプリプラットフォームを選択し、サービスと統合するためのコードを準備します。使用可能なプラットフォームについては、「AWS SDK による認

証」を参照してください。 AWS CLI は、Amazon Cognito およびその他の 用のコマンドライン SDK であり AWS のサービス、Amazon Cognito API に慣れるための貴重な場所です。

> **Note**
>
> Amazon Cognito の一部のコンポーネントは、API でのみ設定できます。例えば、ユーザープールのカスタム SMS または E メール送信者の Lambda トリガーは、 CreateUserPool または UpdateUserPool API リクエストの UserPool クラスの `LambdaConfig`プロパティを更新するリクエストでのみ設定できます。

Amazon Cognito ユーザープール API は、その名前空間を複数の API オペレーションクラスと共有します。1 つのクラスでは、ユーザープールとそのプロセス、ID プロバイダー、ユーザーを設定します。別のクラスでは、パブリッククライアントでユーザーがサインイン、サインアウト、プロファイルの管理を行うための認証されていないオペレーションを提供します。API オペレーションの最後のクラスは、機密サーバー側のクライアントで独自の AWS 認証情報を使用して承認するユーザーオペレーションを実行します。アプリケーションコードの実装を始める前に、目的のアプリケーションアーキテクチャを知っておく必要があります。詳細については、「Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用」を参照してください。

による認証 AWS Amplify

AWS Amplify は、ウェブおよびモバイルアプリケーションを構築するための完全なソリューションです。Amplify を使用すると、Amplify ライブラリを使用して既存のリソースに接続したり、Amplify コマンドラインインターフェイス (CLI) を使用して新しいリソースを作成および設定したりできます。Amplify には、Authenticator などの UI コンポーネントも接続されており、アプリケーションでのサインインやサインアップエクスペリエンスの設定とカスタマイズに使用できます。

フロントエンドアプリケーションで Amplify の認証機能を使用するには、プラットフォーム別の以下のドキュメントを参照してください。

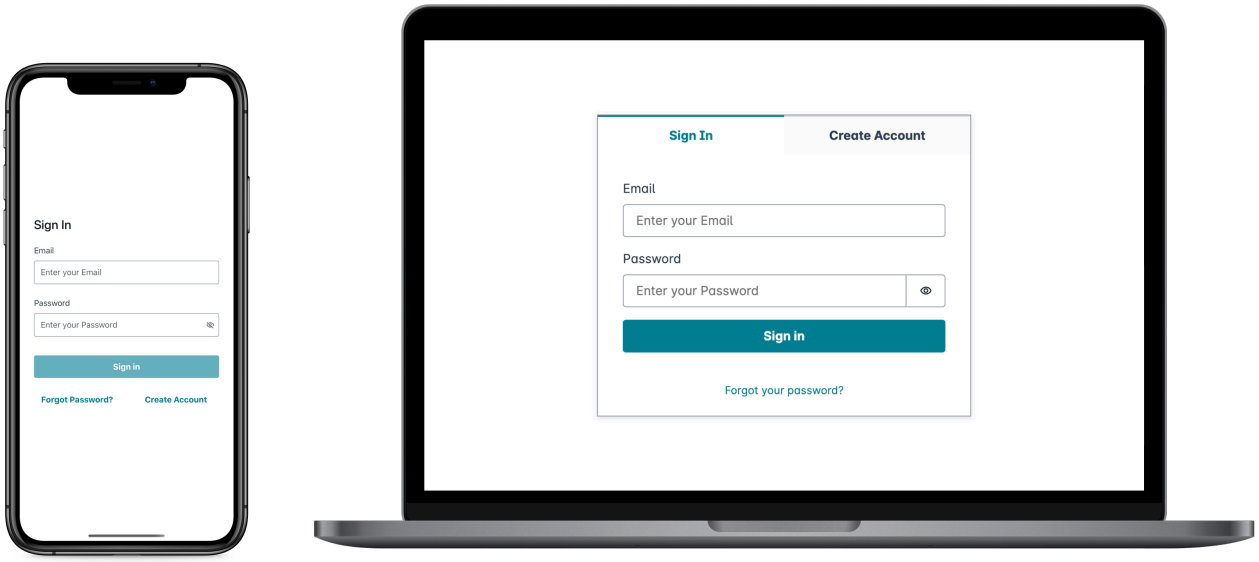
- の Amplify 認証 JavaScript
- Amplify authentication for iOS
- Amplify authentication for Android
- Amplify authentication for Flutter

Amplify ライブラリはオープンソースであり、 で利用できますGitHub。Amplify Auth が Amazon Cognito 認証をどのように実装するかの詳細については、以下のライブラリを参照してください。

- amplify-js
- amplify-swift
- amplify-flutter
- amplify-android

Amplify によるユーザーインターフェイス (UI) の作成

Amazon Cognito ユーザープールでホストされた UI は、ウェブアプリケーションやモバイルアプリケーションの認証フロントエンドの基本的なニーズを満たすことができます。ホストされた UI が対応しているパラメータの範囲を超えてユーザーインターフェイス (UI) をカスタマイズするには、カスタムアプリケーションを作成します。Amplify UI は、さまざまな言語でのカスタマイズ可能なフロントエンドコンポーネントのコレクションです。



カスタム認証コンポーネントの使用を開始するには、Authenticator コンポーネントに関する以下のドキュメントを参照してください。

- Authenticator for Android
- Authenticator for Angular

- Authenticator for Flutter
- Authenticator for React
- Authenticator for React Native
- Authenticator for Swift
- Authenticator for Vue

AWS SDK による認証

安全なバックエンドを使用して Amazon Cognito とやり取りする独自の ID マイクロサービスを構築するには、任意の言語で AWS SDK を使用して Amazon Cognito ユーザープールと Amazon Cognito ID プール API に接続します。

各 API オペレーションの詳細については、「Amazon Cognito ユーザープール API リファレンス」と「Amazon Cognito API リファレンス」を参照してください。これらのドキュメントには、サポートされているプラットフォームでさまざまな SDK を使用するためのリソースに関する、「以下の資料も参照してください」セクションが含まれています。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS の SDK JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Verified Permissions による承認

Amazon Verified Permissions は、お客様が構築するアプリケーションの認証サービスです。Amazon Cognito ユーザープールを ID ソースとして追加すると、アプリケーションはユーザープールアクセスまたはアイデンティティ (ID) トークンを Verified Permissions に渡して、許可または拒否を決定できます。Verified Permissions は、Cedar ポリシー言語で記述したポリシーに基づいてユーザーのプロパティとリクエストコンテキストを考慮します。リクエストコンテキストには、リクエストしたド

キュメント、画像、その他のリソースの ID と、ユーザーがリソースに対して実行するアクションを含めることができます。

アプリは、 または BatchIsAuthorizedWithToken API リクエストで Verified Permissions にユーザーの ID トークンIsAuthorizedWithTokenまたはアクセストークンを提供できます。これらの API オペレーションは、ユーザーを として受け入れPrincipal、アクセスResourceする Actionの の承認決定を行います。追加のカスタムContextは、詳細なアクセス決定に役立ちます。

アプリケーションが IsAuthorizedWithToken API リクエストでトークンを提示すると、Verified Permissions は次の検証を実行します。

1. ユーザープールは、リクエストされたポリシーストアに設定された Verified Permissions の ID ソースです。
2. アクセストークンまたは ID トークンの client_id または aud クレームは、それぞれ Verified Permissions に提供したユーザープールアプリケーションのクライアント ID と一致します。このクレームを検証するには、Verified Permissions の ID ソースでクライアント ID 検証を設定する必要があります。
3. トークンの有効期限は切れていません。
4. トークン内の token_useクレームの値は、 に渡したパラメータと一致しますIsAuthorizedWithToken。token_use クレームは、 accessTokenパラメータに渡accessした場合、および identityTokenパラメータに渡idした場合、 である必要があります。
5. トークンの署名は、ユーザープールの公開された JSON ウェブキー (JWK) から取得されます。JWK は、https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/jwks.json で確認できます。

失効したトークンと削除されたユーザー

Verified Permissions は、ID ソースとユーザートークンの有効期限からの既知の情報のみを検証します。Verified Permissions では、トークンの失効やユーザーの存在は確認されません。ユーザーのトークンを取り消したり、ユーザープールからユーザープロファイルを削除したりした場合でも、Verified Permissions は有効期限が切れるまでトークンを有効と見なします。

ポリシーの評価

ユーザープールをポリシーストアの ID ソースとして設定します。Verified Permissions へのリクエストでユーザーのトークンを送信するようにアプリケーションを設定します。Verified Permissions はリクエストごとに、トークンのクレームをポリシーと比較します。Verified Permissions は、 AWS

の IAM ポリシーに似ていて、プリンシパル、リソース、およびアクションを宣言します。Verified Permissions は、許可されたアクションと一致し、明示的なDenyアクションと一致しAllowない場合、リクエストに で応答します。それ以外の場合は、 で応答しますDeny。詳細については、「Amazon Verified Permissions ユーザーガイド」の「Amazon Verified Permissions のポリシー」を参照してください。

トークンをカスタマイズする

Verified Permissions に提示するユーザークレームを変更、追加、削除するには、 アクセストークンと ID トークンのコンテンツを でカスタマイズしますトークン生成前の Lambda トリガー。プレトークンを生成するトリガーを使用すると、トークンでクレームを追加および変更できます。例えば、データベースで追加のユーザー属性をクエリし、それらを ID トークンにエンコードできます。

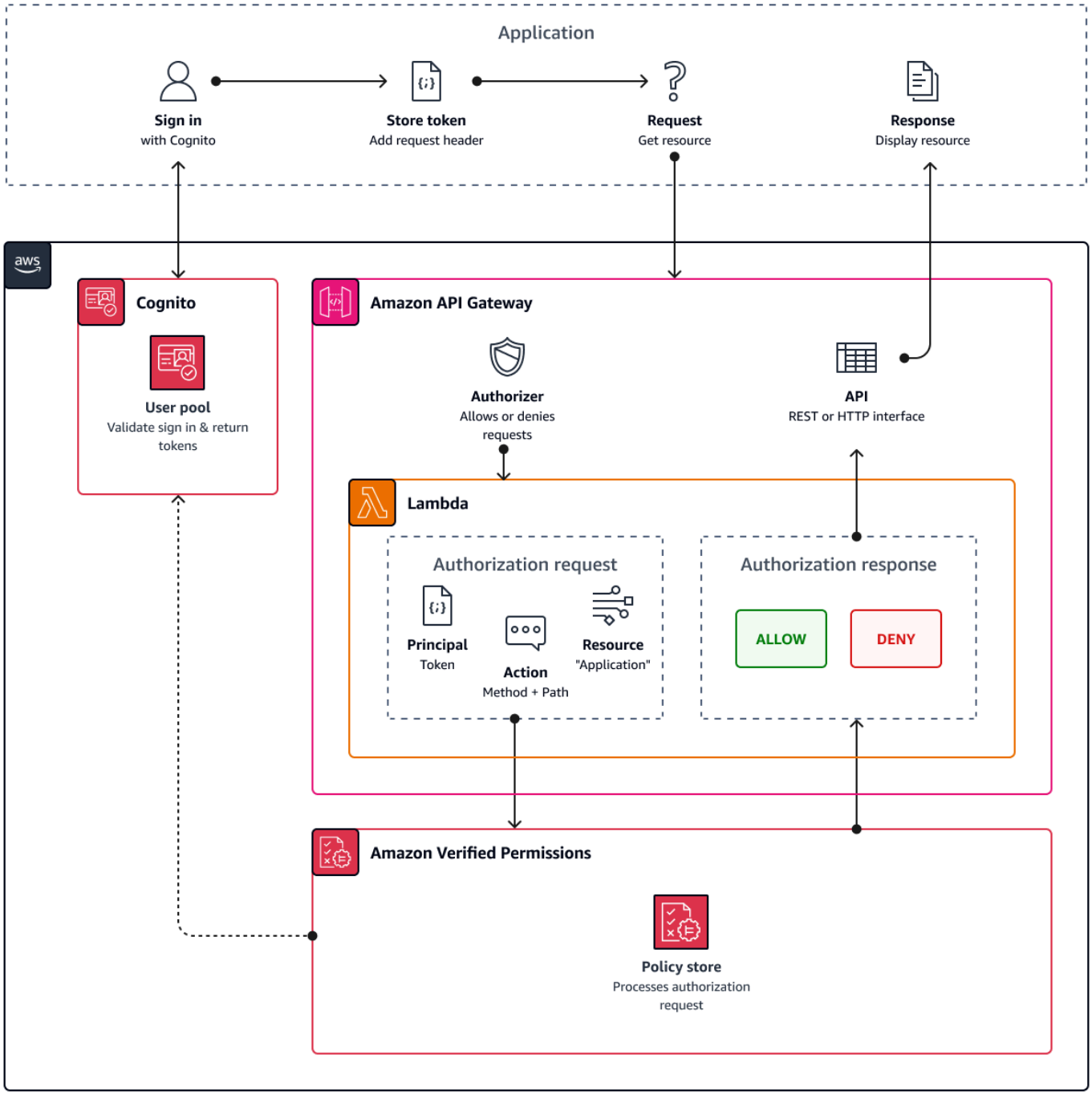
> **Note**
>
> Verified Permissions がクレームを処理する方法を考慮して、cognito、dev、または custom という名前のクレームをプレトークンの生成機能に追加しないでください。これらの予約済みクレームのプレフィックスを、cognito:username のようにコロンで区切られた形式ではなく、完全なクレーム名として提示すると、承認リクエストは失敗します。

Verified Permissions が Amazon Cognito トークンのクレームを認可ポリシーにマッピングする方法の詳細については、「Amazon Cognito トークンを Verified Permissions スキーマにマッピングする」を参照してください。

Verified Permissions による API 認証

ID トークンまたはアクセストークンは、Verified Permissions を使用して、バックエンド Amazon API Gateway REST APIs へのリクエストを承認できます。ユーザープールと API への即時リンクを含むポリシーストアを作成できます。Cognito と API Gateway でのセットアップの開始オプションを使用すると、Verified Permissions はユーザープール ID ソースをポリシーストアに追加し、Lambda オーソライザーを API に追加します。アプリケーションがユーザープールベアラートークンを API に渡すと、Lambda オーソライザーは Verified Permissions を呼び出します。オーソライザーはトークンをプリンシパルとして渡し、リクエストパスとメソッドをアクションとして渡します。

次の図は、Verified Permissions を使用した API Gateway API の承認フローを示しています。詳細については、「Amazon Verified Permissions ユーザーガイド」の「API リンクポリシーストア」を参照してください。



Verified Permissions は、[ユーザープールグループ に関する API]認証を構造化します。ID トークンとアクセストークンの両方に`cognito:groups`クレームが含まれているため、ポリシーストアはさまざまなアプリケーションコンテキストで APIs のロールベースのアクセスコントロール (RBAC) を管理できます。

ポリシーストア設定の選択

ポリシーストアで ID ソースを設定するときは、アクセストークンと ID トークンのどちらを処理するかを選択する必要があります。この決定は、ポリシーエンジンの動作方法にとって重要です。ID トークンにはユーザー属性が含まれます。アクセストークンには、OAuth スコープ というユーザーアクセスコントロール情報が含まれています。どちらのトークンタイプにもグループメンバー情報がありますが、通常、Verified Permissions ポリシーストアを持つ RBAC のアクセストークンをお勧めします。アクセストークンは、承認の決定に貢献できるスコープを持つグループメンバーシップに追加します。アクセストークンのクレームは、認証リクエストのコンテキストになります。

ユーザープールを ID ソースとして設定する場合は、ユーザーエンティティタイプとグループエンティティタイプも設定する必要があります。エンティティタイプは、Verified Permissions ポリシーで参照できるプリンシパル、アクション、およびリソース識別子です。ポリシーストア内のエンティティはメンバーシップ関係を持つことができ、1 つのエンティティを親エンティティのメンバーにすることができます。メンバーシップを使用すると、プリンシパルグループ、アクショングループ、リソースグループを参照できます。ユーザープールグループの場合、指定するユーザーエンティティタイプはグループエンティティタイプのメンバーである必要があります。API にリンクされたポリシーストアを設定するか、Verified Permissions コンソールでガイド付きセットアップに従うと、ポリシーストアにはこの親メンバー関係が自動的に設定されます。

ID トークンは、RBAC と属性ベースのアクセスコントロール (ABAC) を組み合わせることができます。API にリンクされたポリシーストアを作成したら、ユーザー属性とグループメンバーシップを使用してポリシーを強化できます。ID トークンの属性クレームは、認証リクエストのプリンシパル属性になります。ポリシーは、プリンシパル属性に基づいて承認を決定できます。

また、指定した許容可能なアプリケーションクライアントのリストと一致する `aud`または `client_id`クレームを持つトークンを受け入れるようにポリシーストアを設定することもできます。

ロールベースの API 認証のポリシーの例

次のポリシー例は、REST API 例の Verified Permissions PetStore ポリシーストアのセットアップによって作成されました。

```
permit(
  principal in PetStore::UserGroup::"us-east-1_EXAMPLE|MyGroup",
  action in [ PetStore::Action::"get /pets", PetStore::Action::"get /pets/{petId}" ],
  resource
  );
```

Verified Permissions は、次の場合にアプリケーションから認証リクエストにAllow決定を返します。

1. アプリケーションは、 Authorizationヘッダー内の ID またはアクセストークンをベアラートークンとして渡しました。
2. アプリケーションは、文字列 を含む cognito:groupsクレームでトークンを渡しましたMyGroup。
3. アプリケーションが https://myapi.example.com/petsまたは などの にHTTP GETリクエストを行ったhttps://myapi.example.com/pets/scrappy。

Amazon Cognito ユーザーのポリシーの例。

ユーザープールは、API リクエスト以外の条件で Verified Permissions への認証リクエストを生成することもできます。アプリケーション内のアクセスコントロールの決定は、ポリシーストアに送信できます。例えば、リクエストがネットワークを通過する前に、Amazon DynamoDB または Amazon S3 セキュリティを属性ベースのアクセスコントロールで補完し、クォータの使用を削減できます。

次の例では、Cedar ポリシー言語を使用して、1 つのユーザープールアプリケーションクライアントで認証する Finance ユーザーに example_image.png の読み取りと書き込みを許可しています。アプリケーションのユーザーである John は、アプリケーションクライアントから ID トークンを受け取り、それを GET リクエストで認証が必要な URL https://example.com/images/example_image.png に渡します。John の ID トークンには、ユーザープールアプリケーションのクライアント ID 1234567890example の aud クレームが含まれています。また、プレトークンを生成する Lambda 関数によって、John 用に新しいクレーム costCenter (値は Finance1234) が挿入されています。

```
permit (
   principal,
   actions in [ExampleCorp::Action::"readFile", "writeFile"],
   resource == ExampleCorp::Photo::"example_image.png"
)
when {
   principal.aud == "1234567890example" &&
   principal.custom.costCenter like "Finance*"
};
```

次のリクエスト本文の結果は Allow レスポンスになります。

```
{
   "accesstoken": "[John's ID token]",
   "action": {
      "actionId": "readFile",
      "actionType": "Action"
   },
   "resource": {
      "entityId": "example_image.png",
      "entityType": "Photo"
   }
}
```

Verified Permissions でプリンシパルを指定する場合は、次の形式を使用してください。

```
permit (
   principal == [Namespace]::[Entity]::"[user pool ID]"|"[user sub]",
   action,
   resource
);
```

以下は、サブ またはユーザー ID が us-east-1_Exampleの ID を持つユーザープール内のユーザーのプリンシパルの例です973db890-092c-49e4-a9d0-912a4c0a20c7。

```
principal == ExampleCorp::User::"us-east-1_Example|973db890-092c-49e4-
a9d0-912a4c0a20c7",
```

Verified Permissions ポリシーでユーザーグループを指定する場合は、次の形式を使用します。

```
permit (
   principal in [Namespace]::[Group Entity]::"[Group name]",
   action,
   resource
);
```

以下は、 の例です。

単一ドメイン内の属性ベースの

アプリの Verified Permissions による認可、および AWS 認証情報用の Amazon Cognito ID プールのアクセスコントロール機能の属性は、どちらも属性ベースのアクセスコントロール (ABAC) の形式

です。以下は、Verified Permissions と Amazon Cognito ABAC の機能を比較したものです。ABAC では、システムがエンティティの属性を検査し、定義した条件に基づいて承認の決定を行います。

サービス	プロセス	結果
Amazon Verified Permissions	ユーザープール JWT の分析から `Allow`または `Deny`の決定を返します。	アプリケーションリソースへのアクセスは、Cedar ポリシーの評価に基づいて成功または失敗します。
Amazon Cognito ID プール (アクセスコントロールの属性)	属性に基づいてセッションタグをユーザーに割り当てます。IAM ポリシー条件は、タグ`Allow`または `Deny`へのユーザーアクセスをチェックできます AWS のサービス。	IAM ロールの一時的な AWS 認証情報を含むタグ付けされたセッション。

AWS SDK を使用した Amazon Cognito のコード例。

次のコード例は、Amazon Cognito を AWS Software Development Kit (SDK) と使用する方法を示しています。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「AWS SDK でこのサービスを使用する」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コード例

- SDK を使用した Amazon Cognito ID のコード例 AWS SDKs
  - SDK を使用した Amazon Cognito ID のアクション AWS SDKs
    - AWS SDK または CLI CreateIdentityPoolで を使用する
    - AWS SDK または CLI DeleteIdentityPoolで を使用する
    - AWS SDK または CLI DescribeIdentityPoolで を使用する
    - AWS SDK または CLI GetCredentialsForIdentityで を使用する
    - AWS SDK または CLI GetIdentityPoolRolesで を使用する
    - AWS SDK または CLI ListIdentityPoolsで を使用する
    - AWS SDK または CLI SetIdentityPoolRolesで を使用する
    - AWS SDK または CLI UpdateIdentityPoolで を使用する
  - SDK を使用した Amazon Cognito Identity のクロスサービスの例 AWS SDKs
    - Amazon Transcribe アプリを構築する
    - Amazon Textract エクスプローラーアプリケーションを作成する
- SDK を使用した Amazon Cognito ID プロバイダーのコード例 AWS SDKs
  - AWS SDKs を使用した Amazon Cognito ID プロバイダーのアクション
    - AWS SDK または CLI AdminCreateUserで を使用する
    - AWS SDK または CLI AdminGetUserで を使用する
    - AWS SDK または CLI AdminInitiateAuthで を使用する
    - AWS SDK または CLI AdminRespondToAuthChallengeで を使用する
    - AWS SDK または CLI AdminSetUserPasswordで を使用する
    - AWS SDK または CLI AssociateSoftwareTokenで を使用する
    - AWS SDK または CLI ConfirmDeviceで を使用する

SDK を使用した Amazon Cognito ID のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon Cognito Identity を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SDK を使用した Amazon Cognito ID のアクション AWS SDKs
  - AWS SDK または CLI CreateIdentityPoolで を使用する
  - AWS SDK または CLI DeleteIdentityPoolで を使用する
  - AWS SDK または CLI DescribeIdentityPoolで を使用する
  - AWS SDK または CLI GetCredentialsForIdentityで を使用する
  - AWS SDK または CLI GetIdentityPoolRolesで を使用する
  - AWS SDK または CLI ListIdentityPoolsで を使用する
  - AWS SDK または CLI SetIdentityPoolRolesで を使用する
  - AWS SDK または CLI UpdateIdentityPoolで を使用する
- SDK を使用した Amazon Cognito Identity のクロスサービスの例 AWS SDKs
  - Amazon Transcribe アプリを構築する
  - Amazon Textract エクスプローラーアプリケーションを作成する

SDK を使用した Amazon Cognito ID のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の Amazon Cognito ID アクションを実行する方法を示しています。これらは Amazon Cognito Identity API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。各例には GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、Amazon Cognito ID API Reference (Amazon Cognito ID API リファレンス) を参照してください。

例

- AWS SDK または CLI CreateIdentityPoolで を使用する
- AWS SDK または CLI DeleteIdentityPoolで を使用する
- AWS SDK または CLI DescribeIdentityPoolで を使用する
- AWS SDK または CLI GetCredentialsForIdentityで を使用する
- AWS SDK または CLI GetIdentityPoolRolesで を使用する
- AWS SDK または CLI ListIdentityPoolsで を使用する
- AWS SDK または CLI SetIdentityPoolRolesで を使用する
- AWS SDK または CLI UpdateIdentityPoolで を使用する

AWS SDK または CLI **CreateIdentityPool**で を使用する

以下のコード例は、CreateIdentityPool の使用方法を示しています。

CLI

AWS CLI

Cognito アイデンティティプールプロバイダーを含むアイデンティティプールを作成するには

この例では、 という名前の ID プールを作成します MyIdentityPool。これには Cognito アイデンティティプールプロバイダーが含まれます。認証されていないアイデンティティは許可されません。

コマンド:

```
aws cognito-identity create-identity-pool --identity-pool-name
 MyIdentityPool --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

出力:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
```

```
            "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
            "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
            "ServerSideTokenCheck": false
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスCreateIdentityPool」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
```

```
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API の詳細については、「 API リファレンスCreateIdentityPool」の「」を参照してください。 AWS SDK for Java 2.x

PowerShell

のツール PowerShell

例 1: 認証されていない ID を許可する新しい ID プールを作成します。

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

出力:

```
LoggedAt                        : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities  : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName                : CommonTests13
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 136
HttpStatusCode                  : OK
```

- API の詳細については、「 コマンドレットリファレンスCreateIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

新しい ID プールを作成します。

```
    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
"com.exampleco.CognitoIdentityDemo",
                                                       identityPoolName: name)

        let result = try await cognitoIdentityClient.createIdentityPool(input:
cognitoInputCall)
        guard let poolId = result.identityPoolId else {
            return nil
        }

        return poolId
    }
```

- 詳細については、「AWS SDK for Swift デベロッパーガイド」を参照してください。
- API の詳細については、CreateIdentityPoolAWS 「 SDK for Swift API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteIdentityPool**で を使用する

以下のコード例は、DeleteIdentityPool の使用方法を示しています。

CLI

AWS CLI

アイデンティティプールを削除するには

次の delete-identity-pool 例では、指定したアイデンティティプールを削除します。

コマンド:

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスDeleteIdentityPool」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");
```

```
        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスDeleteIdentityPool」の「」を参照してください。 AWS SDK for Java 2.x

PowerShell

のツール PowerShell

例 1: 特定の ID プールを削除します。

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API の詳細については、「 コマンドレットリファレンスDeleteIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

指定した ID プールを削除します。

```
    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        let input = DeleteIdentityPoolInput(
            identityPoolId: id
        )

        _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
    }
```

- 詳細については、「AWS SDK for Swift デベロッパーガイド」を参照してください。
- API の詳細については、DeleteIdentityPoolAWS 「 SDK for Swift API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeIdentityPool**で を使用する

以下のコード例は、DescribeIdentityPool の使用方法を示しています。

CLI

AWS CLI

ID プールを記述するには

この例では、ID プールについて説明します。

コマンド:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

出力:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- API の詳細については、「 コマンドリファレンスDescribeIdentityPool」の「」を参照してください。 AWS CLI

PowerShell

のツール PowerShell

例 1: 特定の ID プールに関する情報を ID で取得します。

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

出力:

```
LoggedAt                       : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : CommonTests1
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 142
HttpStatusCode                 : OK
```

- API の詳細については、「 コマンドレットリファレンスDescribeIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetCredentialsForIdentity**で を使用する

次の例は、GetCredentialsForIdentity を使用する方法を説明しています。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
```

```
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスGetCredentialsForIdentity」の「」を参照してください。 AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetIdentityPoolRoles**で を使用する

以下のコード例は、GetIdentityPoolRoles の使用方法を示しています。

CLI

AWS CLI

ID プールロールを取得するには

この例では、ID プールロールを取得します。

コマンド:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

出力:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- API の詳細については、「 コマンドリファレンスGetIdentityPoolRoles」の「」を参照してください。 AWS CLI

PowerShell

のツール PowerShell

例 1: 特定の ID プールのロールに関する情報を取得します。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

出力:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- API の詳細については、「 コマンドレットリファレンスGetIdentityPoolRoles」の「」を参照してください。 AWS Tools for PowerShell

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListIdentityPools**で を使用する

以下のコード例は、ListIdentityPools の使用方法を示しています。

CLI

AWS CLI

アイデンティティプールを一覧表示するには

この例ではアイデンティティプールを一覧表示します。最大 20 個のアイデンティティが一覧表示されます。

コマンド:

```
aws cognito-identity list-identity-pools --max-results 20
```

出力:

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- API の詳細については、「 コマンドリファレンスListIdentityPools」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスListIdentityPools」の「」を参照してください。 AWS SDK for Java 2.x

PowerShell

のツール PowerShell

例 1: 既存の ID プールのリストを取得します。

```
Get-CGIIdentityPoolList
```

出力:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- API の詳細については、「 コマンドレットリファレンスListIdentityPools」の「」を参照してください。 AWS Tools for PowerShell

Swift

SDK for Swift

> **Note**
>
> これはプレビューリリースの SDK に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

名前を指定してアイデンティティプールの ID を検索します。

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        var token: String? = nil

        // Iterate over the identity pools until a match is found.

        repeat {
            /// `token` is a value returned by `ListIdentityPools()` if the
            /// returned list of identity pools is only a partial list. You
            /// use the `token` to tell Amazon Cognito that you want to
            /// continue where you left off previously. If you specify `nil`
            /// or you don't provide the token, Amazon Cognito will start at
            /// the beginning.

            let listPoolsInput = ListIdentityPoolsInput(maxResults: 25,
nextToken: token)

            /// Read pages of identity pools from Cognito until one is found
            /// whose name matches the one specified in the `name` parameter.
            /// Return the matching pool's ID. Each time we ask for the next
            /// page of identity pools, we pass in the token given by the
            /// previous page.

            let output = try await cognitoIdentityClient.listIdentityPools(input:
listPoolsInput)

            if let identityPools = output.identityPools {
                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }

            token = output.nextToken
```

```
        } while token != nil

        return nil
    }
```

既存のアイデンティティプールの ID を取得します。存在しない場合は、作成します。

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        guard let poolId = try await self.getIdentityPoolID(name: name) else {
            return try await self.createIdentityPool(name: name)
        }

        return poolId
    }
```

- 詳細については、「AWS SDK for Swift デベロッパーガイド」を参照してください。
- API の詳細については、ListIdentityPoolsAWS 「 SDK for Swift API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetIdentityPoolRoles**で を使用する

以下のコード例は、SetIdentityPoolRoles の使用方法を示しています。

CLI

AWS CLI

ID プールロールを設定するには

次のset-identity-pool-roles例では、ID プールロールを設定します。

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- API の詳細については、「 コマンドリファレンスSetIdentityPoolRoles」の「」を参照してください。 AWS CLI

PowerShell

のツール PowerShell

例 1: 認証されていない IAM ロールを持つように特定の ID プールを設定します。

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API の詳細については、「 コマンドレットリファレンスSetIdentityPoolRoles」の「」を参照してください。 AWS Tools for PowerShell

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UpdateIdentityPool**で を使用する

以下のコード例は、UpdateIdentityPool の使用方法を示しています。

CLI

AWS CLI

ID プールを更新するには

この例では、ID プールを更新します。名前を に設定します MyIdentityPool。Cognito を ID プロバイダーとして追加します。認証されていない ID は許可されません。

コマンド:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-name
 "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

出力:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- API の詳細については、「 コマンドリファレンスUpdateIdentityPool」の「」を参照してください。 AWS CLI

PowerShell

のツール PowerShell

例 1: ID プールプロパティの一部を更新します。この場合は、ID プールの名前を更新します。

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

出力:

```
LoggedAt                        : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities  : False
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : NewPoolName
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 135
HttpStatusCode                  : OK
```

- API の詳細については、「 コマンドレットリファレンスUpdateIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon Cognito Identity のクロスサービスの例 AWS SDKs

次のサンプルアプリケーションでは、 AWS SDKsを使用して Amazon Cognito Identity を他の と組み合わせます AWS のサービス。各例には GitHub、アプリケーションのセットアップと実行の手順を示す へのリンクが含まれています。

例

- Amazon Transcribe アプリを構築する
- Amazon Textract エクスプローラーアプリケーションを作成する

Amazon Transcribe アプリを構築する

次のコード例は、Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示する方法を示しています。

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示するアプリを作成します。このアプリは 2 つの Amazon Simple Storage Service (Amazon S3) バケットを使用します。1 つはアプリケーションコードをホストし、もう 1 つは文字起こしを保存します。このアプリは、Amazon Cognito ユーザープールを使用してユーザーを認証します。認証されたユーザーには、必要な AWS サービスにアクセスするための AWS Identity and Access Management (IAM) アクセス許可があります。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract エクスプローラーアプリケーションを作成する

次のコード例は、インタラクティブアプリケーションで Amazon Textract の出力を調べる方法を示しています。

JavaScript

SDK for JavaScript (v3)

を使用して、Amazon Textract を使用してドキュメントイメージからデータを抽出し、インタラクティブなウェブページに表示する React アプリケーション AWS SDK for JavaScript を構築する方法を示します。この例はウェブブラウザで実行され、認証情報に認証された Amazon Cognito ID が必要です。Amazon Simple Storage Service (Amazon S3) をストレージに使用し、通知のために、Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブした Amazon Simple Queue Service (Amazon SQS) キューをポーリングします。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon Cognito ID プロバイダーのコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon Cognito ID プロバイダーを使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

Hello Amazon Cognito

次のコード例は、Amazon Cognito の使用を開始する方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

C MakeLists.txt CMake ファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_cognito.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
```

```
}
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。 AWS SDK for C++

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
```

```
 return
}
cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
fmt.Println("Let's list the user pools for your account.")
var pools []types.UserPoolDescriptionType
paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
 cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
aws.Int32(10)})
for paginator.HasMorePages() {
 output, err := paginator.NextPage(context.TODO())
 if err != nil {
  log.Printf("Couldn't get user pools. Here's why: %v\n", err)
 } else {
  pools = append(pools, output.UserPools...)
 }
}
if len(pools) == 0 {
 fmt.Println("You don't have any user pools!")
} else {
 for _, pool := range pools {
  fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
 }
}
}
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。AWS SDK for JavaScript

コードの例

- AWS SDKs を使用した Amazon Cognito ID プロバイダーのアクション
  - AWS SDK または CLI AdminCreateUserで を使用する
  - AWS SDK または CLI AdminGetUserで を使用する
  - AWS SDK または CLI AdminInitiateAuthで を使用する
  - AWS SDK または CLI AdminRespondToAuthChallengeで を使用する
  - AWS SDK または CLI AdminSetUserPasswordで を使用する
  - AWS SDK または CLI AssociateSoftwareTokenで を使用する
  - AWS SDK または CLI ConfirmDeviceで を使用する
  - AWS SDK または CLI ConfirmForgotPasswordで を使用する
  - AWS SDK または CLI ConfirmSignUpで を使用する
  - AWS SDK または CLI CreateUserPoolで を使用する
  - AWS SDK または CLI CreateUserPoolClientで を使用する
  - AWS SDK または CLI DeleteUserで を使用する
  - AWS SDK または CLI ForgotPasswordで を使用する
  - AWS SDK または CLI InitiateAuthで を使用する
  - AWS SDK または CLI ListUserPoolsで を使用する
  - AWS SDK または CLI ListUsersで を使用する
  - AWS SDK または CLI ResendConfirmationCodeで を使用する
  - AWS SDK または CLI RespondToAuthChallengeで を使用する
  - AWS SDK または CLI SignUpで を使用する
  - AWS SDK または CLI UpdateUserPoolで を使用する
  - AWS SDK または CLI VerifySoftwareTokenで を使用する
- SDK を使用する Amazon Cognito ID プロバイダーのシナリオ AWS SDKs
  - AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に確認する
  - AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に移行する

- AWS SDK を使用して MFA を必要とする Amazon Cognito ユーザープールでユーザーをサインアップする
- AWS SDK を使用して Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

AWS SDKs を使用した Amazon Cognito ID プロバイダーのアクション

次のコード例は、 AWS SDKs を使用して個々の Amazon Cognito ID プロバイダーアクションを実行する方法を示しています。これらは Amazon Cognito Identity Provider API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。各例には へのリンクが含まれており GitHub、コードの設定と実行の手順を確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、Amazon Cognito Identity Provider API Reference (Amazon Cognito Identity Provider API リファレンス) を参照してください。

例

- AWS SDK または CLI AdminCreateUserで を使用する
- AWS SDK または CLI AdminGetUserで を使用する
- AWS SDK または CLI AdminInitiateAuthで を使用する
- AWS SDK または CLI AdminRespondToAuthChallengeで を使用する
- AWS SDK または CLI AdminSetUserPasswordで を使用する
- AWS SDK または CLI AssociateSoftwareTokenで を使用する
- AWS SDK または CLI ConfirmDeviceで を使用する
- AWS SDK または CLI ConfirmForgotPasswordで を使用する
- AWS SDK または CLI ConfirmSignUpで を使用する
- AWS SDK または CLI CreateUserPoolで を使用する
- AWS SDK または CLI CreateUserPoolClientで を使用する
- AWS SDK または CLI DeleteUserで を使用する
- AWS SDK または CLI ForgotPasswordで を使用する
- AWS SDK または CLI InitiateAuthで を使用する
- AWS SDK または CLI ListUserPoolsで を使用する

- AWS SDK または CLI ListUsersで を使用する
- AWS SDK または CLI ResendConfirmationCodeで を使用する
- AWS SDK または CLI RespondToAuthChallengeで を使用する
- AWS SDK または CLI SignUpで を使用する
- AWS SDK または CLI UpdateUserPoolで を使用する
- AWS SDK または CLI VerifySoftwareTokenで を使用する

AWS SDK または CLI **AdminCreateUser**で を使用する

以下のコード例は、AdminCreateUser の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

CLI

AWS CLI

ユーザーを作成するには

次のadmin-create-user例では、指定された設定の E メールアドレスと電話番号を持つユーザーを作成します。

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

出力:

```
{
    "User": {
        "Username": "diego",
```

```
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- API の詳細については、「 コマンドリファレンスAdminCreateUser」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- API の詳細については、「 API リファレンスAdminCreateUser」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AdminGetUser**で を使用する

以下のコード例は、AdminGetUser の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /// <summary>
   /// Get the specified user from an Amazon Cognito user pool with
administrator access.
   /// </summary>
   /// <param name="userName">The name of the user.</param>
   /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
   /// <returns>Async task.</returns>
   public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
   {
       AdminGetUserRequest userRequest = new AdminGetUserRequest
       {
           Username = userName,
           UserPoolId = poolId,
       };

       var response = await _cognitoService.AdminGetUserAsync(userRequest);

       Console.WriteLine($"User status {response.UserStatus}");
       return response.UserStatus;
   }
```

- API の詳細については、「 API リファレンスAdminGetUser」の「」を参照してください。AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API の詳細については、「 API リファレンスAdminGetUser」の「」を参照してください。AWS SDK for C++

CLI

AWS CLI

ユーザーを取得するには

この例では、ユーザー名 jane@example.com に関する情報を取得します。

コマンド:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --username
 jane@example.com
```

出力:

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number",
          "Value": "+01115551212"
      },
      {
          "Name": "email",
          "Value": "jane@example.com"
      }
  ],
  "UserLastModifiedDate": 1548108509.537
```

```
}
```

- API の詳細については、「 コマンドリファレンスAdminGetUser」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
           String poolId) {
       try {
           AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                   .username(userName)
                   .userPoolId(poolId)
                   .build();

           AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
           System.out.println("User status " + response.userStatusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API の詳細については、「 API リファレンスAdminGetUser」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスAdminGetUser」の「」を参照してください。AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスAdminGetUserの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
    return confirmed
```

- API の詳細については、 AdminGetUser AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AdminInitiateAuth**で を使用する

以下のコード例は、AdminInitiateAuth の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <summary>
/// Initiate an admin auth request.
/// </summary>
/// <param name="clientId">The client ID to use.</param>
/// <param name="userPoolId">The ID of the user pool.</param>
/// <param name="userName">The username to authenticate.</param>
/// <param name="password">The user's password.</param>
```

```
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- API の詳細については、「 API リファレンスAdminInitiateAuth」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API の詳細については、「 API リファレンスAdminInitiateAuth」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

認証を開始するには

この例では、ユーザー名 jane@example.com の ADMIN_NO_SRP_AUTH フローを使用して認証を開始します。

クライアントでは、サーバーベースの認証用のサインイン API (ADMIN_NO_SRP_AUTH) が有効になっている必要があります。

戻り値のセッション情報を使用して、 admin-respond-to-auth-challenge を呼び出します。

コマンド:

```
aws cognito-idp admin-initiate-auth --user-pool-id us-west-2_aaaaaaaaa --client-
id 3n4b5urk1ft4fl3mg5e62d9ado --auth-flow ADMIN_NO_SRP_AUTH --auth-parameters
 USERNAME=jane@example.com,PASSWORD=password
```

出力:

```
{
  "ChallengeName": "NEW_PASSWORD_REQUIRED",
  "Session": "SESSION",
  "ChallengeParameters": {
      "USER_ID_FOR_SRP": "84514837-dcbc-4af1-abff-f3c109334894",
      "requiredAttributes": "[]",
      "userAttributes": "{\"email_verified\":\"true\",\"phone_number_verified\":
\"true\",\"phone_number\":\"+01xxx5550100\",\"email\":\"jane@example.com\"}"
  }
}
```

- API の詳細については、「 コマンドリファレンスAdminInitiateAuth」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);
```

```
            AdminInitiateAuthRequest authRequest =
 AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
 identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " +
 response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- API の詳細については、「 API リファレンスAdminInitiateAuth」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
```

```
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスAdminInitiateAuth」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
```

```
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスAdminInitiateAuthの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.
```

```
        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
 not "
                        "configured for TOTP MFA. This example requires TOTP
 MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
```

```
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API の詳細については、 AdminInitiateAuth AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AdminRespondToAuthChallenge**で を使用する

以下のコード例は、AdminRespondToAuthChallenge の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <summary>
/// Respond to an admin authentication challenge.
```

```
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- API の詳細については、「 API リファレンスAdminRespondToAuthChallenge」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region (overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
```

```
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「 API リファレンスAdminRespondToAuthChallenge」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
```

```
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

 System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

- API の詳細については、「 API リファレンスAdminRespondToAuthChallenge」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスAdminRespondToAuthChallenge」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスAdminRespondToAuthChallengeの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

MFA のチャレンジに応答するには、関連する MFA アプリケーションによって生成されたコードを入手します。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
 of
        a two-factor sign-in. When sign-in is successful, it returns an access
 token
```

```
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        else:
            return auth_result
```

- API の詳細については、 AdminRespondToAuthChallenge AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AdminSetUserPassword**で を使用する

次の例は、AdminSetUserPassword を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- API の詳細については、「 API リファレンスAdminSetUserPassword」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AssociateSoftwareToken**で を使用する

以下のコード例は、AssociateSoftwareToken の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- API の詳細については、「 API リファレンスAssociateSoftwareToken」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
```

```
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「 API リファレンスAssociateSoftwareToken」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
   }
```

- API の詳細については、「 API リファレンスAssociateSoftwareToken」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスAssociateSoftwareToken」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスAssociateSoftwareTokenの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
   def get_mfa_secret(self, session):
       """
       Gets a token that can be used to associate an MFA application with the
user.

       :param session: Session information returned from a previous call to
initiate
                       authentication.
       :return: An MFA token that can be used to set up an MFA application.
       """
       try:
           response =
self.cognito_idp_client.associate_software_token(Session=session)
       except ClientError as err:
           logger.error(
               "Couldn't get MFA secret. Here's why: %s: %s",
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           response.pop("ResponseMetadata", None)
           return response
```

- API の詳細については、 AssociateSoftwareToken AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ConfirmDevice**で を使用する

以下のコード例は、ConfirmDevice の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- API の詳細については、「 API リファレンスConfirmDevice」の「」を参照してください。AWS SDK for .NET

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスConfirmDevice」の「」を参照してください。AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
```

```
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
```

```
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- API の詳細については、 ConfirmDevice AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ConfirmForgotPassword**で を使用する

以下のコード例は、ConfirmForgotPassword の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に移行する

CLI

AWS CLI

忘れたパスワードを確認するには

この例では、ユーザー名 diego@example.com のパスワードを忘れたことを確認します。

コマンド:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- API の詳細については、「 コマンドリファレンスConfirmForgotPassword」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- API の詳細については、「 API リファレンスConfirmForgotPassword」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ConfirmSignUp**で を使用する

以下のコード例は、ConfirmSignUp の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
```

```
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- API の詳細については、「 API リファレンスConfirmSignUp」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「 API リファレンスConfirmSignUp」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

サインアップを確認するには

この例では、ユーザー名 diego@example.com のサインアップを確認します。

コマンド:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- API の詳細については、「 コマンドリファレンスConfirmSignUp」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    public static void confirmSignUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、「 API リファレンスConfirmSignUp」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });
```

```
  return client.send(command);
};
```

- API の詳細については、「 API リファレンスConfirmSignUp」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスConfirmSignUpの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
 registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
```

```
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- API の詳細については、 ConfirmSignUp AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateUserPool**で を使用する

以下のコード例は、CreateUserPool の使用方法を示しています。

CLI

AWS CLI

最小設定のユーザープールを作成するには

この例では、デフォルト値 MyUserPool を使用して という名前のユーザープールを作成します。必要な属性やアプリケーションクライアントはありません。MFA およびアドバンストセキュリティは無効化されています。

コマンド:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

出力:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "family_name",
              "StringAttributeConstraints": {
```

```
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
```

```
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
```

```
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
```

```
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
```

```
      "MfaConfiguration": "OFF",
      "Name": "MyUserPool",
      "LastModifiedDate": 1547833345.777,
      "AdminCreateUserConfig": {
          "UnusedAccountValidityDays": 7,
          "AllowAdminCreateUserOnly": false
      },
      "EmailConfiguration": {},
      "Policies": {
          "PasswordPolicy": {
              "RequireLowercase": true,
              "RequireSymbols": true,
              "RequireNumbers": true,
              "MinimumLength": 8,
              "RequireUppercase": true
          }
      },
      "CreationDate": 1547833345.777,
      "EstimatedNumberOfUsers": 0,
      "Id": "us-west-2_aaaaaaaaa",
      "LambdaConfig": {}
  }
}
```

2 つの必須属性でユーザープールを作成するには

この例では、ユーザープール を作成します MyUserPool。プールは、E メールをユーザー名属性として受け入れるように設定されています。また、Amazon Simple Email Service を使用して、E メールの送信元アドレスを検証済みのアドレスに設定します。

コマンド:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

出力:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
```

```
                "Name": "sub",
                "StringAttributeConstraints": {
                    "MinLength": "1",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": true,
                "AttributeDataType": "String",
                "Mutable": false
            },
            {
                "Name": "name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
```

```
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
```

```
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "zoneinfo",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "locale",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
    "MfaConfiguration": "OFF",
    "Name": "MyUserPool",
    "LastModifiedDate": 1547837788.189,
    "AdminCreateUserConfig": {
        "UnusedAccountValidityDays": 7,
        "AllowAdminCreateUserOnly": false
    },
    "EmailConfiguration": {
        "ReplyToEmailAddress": "jane@example.com",
```

```
          "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
      },
      "Policies": {
          "PasswordPolicy": {
              "RequireLowercase": true,
              "RequireSymbols": true,
              "RequireNumbers": true,
              "MinimumLength": 8,
              "RequireUppercase": true
          }
      },
      "UsernameAttributes": [
          "email"
      ],
      "CreationDate": 1547837788.189,
      "EstimatedNumberOfUsers": 0,
      "Id": "us-west-2_aaaaaaaaa",
      "LambdaConfig": {}
  }
}
```

- API の詳細については、「 コマンドリファレンスCreateUserPool」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }
```

```
    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
 cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API の詳細については、「 API リファレンスCreateUserPool」の「」を参照してください。 AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateUserPoolClient**で を使用する

以下のコード例は、CreateUserPoolClient の使用方法を示しています。

CLI

AWS CLI

ユーザープールクライアントを作成するには

この例では、USER_PASSWORD_AUTH と ADMIN_NO_SRP_AUTH の 2 つの明示的な認証フローを持つ新しいユーザープールクライアントを作成します。

コマンド:

```
aws cognito-idp create-user-pool-client --user-pool-id us-west-2_aaaaaaaaa
  --client-name MyNewClient --no-generate-secret --explicit-auth-flows
 "USER_PASSWORD_AUTH" "ADMIN_NO_SRP_AUTH"
```

出力:

```
{
  "UserPoolClient": {
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "ClientName": "MyNewClient",
      "ClientId": "6p3bs000no6a4ue1idruvd05ad",
      "LastModifiedDate": 1548697449.497,
      "CreationDate": 1548697449.497,
      "RefreshTokenValidity": 30,
      "ExplicitAuthFlows": [
          "USER_PASSWORD_AUTH",
          "ADMIN_NO_SRP_AUTH"
      ],
      "AllowedOAuthFlowsUserPoolClient": false
  }
}
```

- API の詳細については、「 コマンドリファレンスCreateUserPoolClient」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスCreateUserPoolClient」の「」を参照してください。 AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteUser**で を使用する

以下のコード例は、DeleteUser の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に確認する
- Lambda 関数を使用して登録済みのユーザーを自動的に移行する
- Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API の詳細については、「 API リファレンスDeleteUser」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

ユーザーを削除するには

この例では、ユーザーを削除します。

コマンド:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- API の詳細については、「 コマンドリファレンスDeleteUser」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- API の詳細については、「 API リファレンスDeleteUser」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ForgotPassword**で を使用する

以下のコード例は、ForgotPassword の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に移行する

CLI

AWS CLI

パスワードを強制的に変更するには

次のforgot-password例では、パスワードを変更するメッセージを jane@example.com に送信します。

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --username
 jane@example.com
```

出力:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
```

```
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- API の詳細については、「 コマンドリファレンスForgotPassword」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- API の詳細については、「 API リファレンスForgotPassword」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **InitiateAuth**で を使用する

以下のコード例は、InitiateAuth の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に確認する
- Lambda 関数を使用して登録済みのユーザーを自動的に移行する
- MFA を必要とするユーザープールによりユーザーをサインアップする
- Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
```

```
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- API の詳細については、「 API リファレンスInitiateAuth」の「」を参照してください。 AWS SDK for .NET

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
```

```
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- API の詳細については、「 API リファレンスInitiateAuth」の「」を参照してください。 AWS SDK for Go

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスInitiateAuth」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

この例は、トラッキング対象デバイスを使用して認証を開始する方法を示しています。サインインを完了するには、クライアントはセキュアリモートパスワード (SRP) チャレンジに正しく応答する必要があります。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
```

```
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
```

```
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
```

```
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API の詳細については、 InitiateAuth AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListUserPools**で を使用する

以下のコード例は、ListUserPools の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

ユーザープールを一覧表示するには

次の例では、最大 20 のユーザープールを一覧表示します。

コマンド:

```
aws cognito-idp list-user-pools --max-results 20
```

出力:

```
{
  "UserPools": [
      {
         "CreationDate": 1547763720.822,
```

```
            "LastModifiedDate": 1547763720.822,
            "LambdaConfig": {},
            "Id": "us-west-2_aaaaaaaaa",
            "Name": "MyUserPool"
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスListUserPools」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
```

```
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスListUserPools」の「」を参照してください。AWS SDK for Java 2.x

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
```

```
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API の詳細については、 ListUserPools AWS SDK for Rust API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListUsers**で を使用する

以下のコード例は、ListUsers の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- API の詳細については、「 API リファレンスListUsers」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

ユーザーを一覧表示するには

この例では最大 20 のユーザーを一覧表示します。

コマンド:

```
aws cognito-idp list-users --user-pool-id us-west-2_aaaaaaaaa --limit 20
```

出力:

```
{
  "Users": [
      {
          "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
          "Enabled": true,
          "UserStatus": "FORCE_CHANGE_PASSWORD",
          "UserCreateDate": 1548089817.683,
          "UserLastModifiedDate": 1548089817.683,
          "Attributes": [
              {
                  "Name": "sub",
                  "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
              },
              {
                  "Name": "email_verified",
                  "Value": "true"
              },
              {
                  "Name": "email",
                  "Value": "mary@example.com"
              }
          ]
      }
  ]
}
```

- API の詳細については、「 コマンドリファレンスListUsers」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
 created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
```

```
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
cognitoClient, String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
+ " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API の詳細については、「 API リファレンスListUsers」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスListUsers」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllUsers(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスListUsersの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
```

```
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- API の詳細については、 ListUsers AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ResendConfirmationCode**で を使用する

以下のコード例は、ResendConfirmationCode の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);
```

```
        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- API の詳細については、「 API リファレンスResendConfirmationCode」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
```

```
                    << "CognitoIdentityProvider::ResendConfirmationCode was
 successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「 API リファレンスResendConfirmationCode」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

確認コードを再送信するには

次の resend-confirmation-code 例では、ユーザー jane に確認コードを送信します。

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

出力:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

詳細については、「Amazon Cognito デベロッパーガイド」の「ユーザーアカウントのサインアップと確認」を参照してください。

- API の詳細については、「 コマンドリファレンスResendConfirmationCode」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API の詳細については、「 API リファレンスResendConfirmationCode」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスResendConfirmationCode」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスResendConfirmationCodeの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- API の詳細については、 ResendConfirmationCode AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RespondToAuthChallenge**で を使用する

以下のコード例は、RespondToAuthChallenge の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

CLI

AWS CLI

認証チャレンジに応答するには

この例では initiate-auth で開始された認証チャレンジに応答します。NEW_PASSWORD_REQUIRED チャレンジへの応答です。ユーザー jane@example.com のパスワードを設定します。

コマンド:

```
aws cognito-idp respond-to-auth-challenge --client-id 3n4b5urk1ft4fl3mg5e62d9ado
 --challenge-name NEW_PASSWORD_REQUIRED --challenge-responses
 USERNAME=jane@example.com,NEW_PASSWORD="password" --session "SESSION_TOKEN"
```

出力:

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
      "AccessToken": "ACCESS_TOKEN",
      "ExpiresIn": 3600,
      "TokenType": "Bearer",
      "RefreshToken": "REFRESH_TOKEN",
      "IdToken": "ID_TOKEN",
      "NewDeviceMetadata": {
          "DeviceKey": "us-west-2_fec070d2-fa88-424a-8ec8-b26d7198eb23",
          "DeviceGroupKey": "-wt2ha1Zd"
      }
  }
}
```

- API の詳細については、「 コマンドリファレンスRespondToAuthChallenge」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスRespondToAuthChallenge」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

追跡対象デバイスでサインインします。サインインを完了するには、クライアントはセキュアリモートパスワード (SRP) チャレンジに正しく応答する必要があります。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
```

```
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
```

```
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API の詳細については、 RespondToAuthChallenge AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SignUp**で を使用する

以下のコード例は、SignUp の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に確認する
- Lambda 関数を使用して登録済みのユーザーを自動的に移行する
- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
```

```
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API の詳細については、「 API リファレンスSignUp」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

- API の詳細については、「 API リファレンスSignUp」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

ユーザーをサインアップするには

この例では jane@example.com をサインアップします。

コマンド:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

出力:

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- API の詳細については、「 コマンドリファレンスSignUp」の「」を参照してください。AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- API の詳細については、「 API リファレンスSignUp」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
           String password, String email) {
       AttributeType userAttrs = AttributeType.builder()
               .name("email")
               .value(email)
               .build();

       List<AttributeType> userAttrsList = new ArrayList<>();
       userAttrsList.add(userAttrs);
       try {
           SignUpRequest signUpRequest = SignUpRequest.builder()
                   .userAttributes(userAttrsList)
                   .username(userName)
                   .clientId(clientId)
                   .password(password)
                   .build();

           identityProviderClient.signUp(signUpRequest);
           System.out.println("User has been signed up ");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API の詳細については、「 API リファレンスSignUp」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスSignUp」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
```

```
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスSignUpの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
   def sign_up_user(self, user_name, password, user_email):
       """
       Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
       to send an email to the specified email address. The email contains a
code that
       can be used to confirm the user.

       When the user already exists, the user status is checked to determine
whether
       the user has been confirmed.

       :param user_name: The user name that identifies the new user.
       :param password: The password for the new user.
       :param user_email: The email address for the new user.
       :return: True when the user is already confirmed with Amazon Cognito.
                Otherwise, false.
       """
       try:
           kwargs = {
               "ClientId": self.client_id,
               "Username": user_name,
               "Password": password,
               "UserAttributes": [{"Name": "email", "Value": user_email}],
           }
           if self.client_secret is not None:
               kwargs["SecretHash"] = self._secret_hash(user_name)
           response = self.cognito_idp_client.sign_up(**kwargs)
           confirmed = response["UserConfirmed"]
       except ClientError as err:
           if err.response["Error"]["Code"] == "UsernameExistsException":
               response = self.cognito_idp_client.admin_get_user(
                   UserPoolId=self.user_pool_id, Username=user_name
               )
               logger.warning(
                   "User %s exists and is %s.", user_name,
response["UserStatus"]
               )
               confirmed = response["UserStatus"] == "CONFIRMED"
           else:
               logger.error(
                   "Couldn't sign up %s. Here's why: %s: %s",
                   user_name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
    return confirmed
```

- API の詳細については、 SignUp AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UpdateUserPool**で を使用する

以下のコード例は、UpdateUserPool の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- Lambda 関数を使用して登録済みのユーザーを自動的に確認する
- Lambda 関数を使用して登録済みのユーザーを自動的に移行する
- Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

CLI

AWS CLI

ユーザープールを更新するには

この例では、ユーザープールにタグを追加します。

コマンド:

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_aaaaaaaaa --user-pool-
tags Team=Blue,Area=West
```

- API の詳細については、「 コマンドリファレンスUpdateUserPool」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
```

```
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- API の詳細については、「 API リファレンスUpdateUserPool」の「」を参照してください。 AWS SDK for Go

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **VerifySoftwareToken**で を使用する

以下のコード例は、VerifySoftwareToken の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- MFA を必要とするユーザープールによりユーザーをサインアップする

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- API の詳細については、「 API リファレンスVerifySoftwareToken」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「 API リファレンスVerifySoftwareToken」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API の詳細については、「 API リファレンスVerifySoftwareToken」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API の詳細については、「 API リファレンスVerifySoftwareToken」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスVerifySoftwareTokenの「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class CognitoIdentityProviderWrapper:
```

```
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API の詳細については、 VerifySoftwareToken AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用する Amazon Cognito ID プロバイダーのシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して Amazon Cognito ID プロバイダーで一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon Cognito ID プロバイダー内で複数の関数を呼び出すことによって特定のタスクを実行する方法を示しています。各シナリオには GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

例

- AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に確認する
- AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に移行する
- AWS SDK を使用して MFA を必要とする Amazon Cognito ユーザープールでユーザーをサインアップする
- AWS SDK を使用して Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に確認する

次のコード例は、Lambda 関数を使用して登録済みの Amazon Cognito ユーザーを確認する方法を示しています。

- `PreSignUp` トリガーの Lambda 関数を呼び出すようにユーザープールを設定します。
- Amazon Cognito でユーザーをサインアップする
- Lambda 関数は DynamoDB テーブルをスキャンし、登録済みのユーザーを自動的に確認します。
- 新しいユーザーとしてサインインし、リソースをクリーンアップします。

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
```

```
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(clientId string, usersTable string)
 (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
```

```
    if errors.As(err, &invalidPassword) {
     password = runner.questioner.AskPassword("Enter another password:", 8)
    } else {
     panic(err)
    }
   } else {
    signedUp = true
   }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(clientId string, userName string, password
 string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))
```

```
 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 関数を使用して PreSignUp トリガーを処理します。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
```

```
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
```

```
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

一般的なタスクを実行する構造体を作成します。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}
```

```
// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
```

```
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
```

```
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito アクションをラップする構造体を作成します。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
```

```
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
```

```
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
```

```
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
```

```
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
```

```
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB アクションをラップする構造体を作成します。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}
```

```
// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
```

```
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Logs CloudWatch アクションをラップする構造体を作成します。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
```

```
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

AWS CloudFormation アクションをラップする構造体を作成します。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

リソースをクリーンアップします。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
```

```
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Lambda 関数で既知の Amazon Cognito ユーザーを自動的に移行する

次のコード例は、Lambda 関数を使用して登録済みの Amazon Cognito ユーザーを自動的に移行する方法を示しています。

- `MigrateUser` トリガーの Lambda 関数を呼び出すようにユーザープールを設定します。
- ユーザープールにないユーザー名と E メールで Amazon Cognito にサインインします。
- Lambda 関数は DynamoDB テーブルをスキャンし、登録済みのユーザーをユーザープールに自動的に移行します。
- パスワードを忘れた場合のフローを実行して、移行したユーザーのパスワードをリセットします。
- 新しいユーザーとしてサインインし、リソースをクリーンアップします。

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
import (
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
```

```
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(usersTable string, clientId string) (bool,
 actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")
```

```
 runner.helper.AddKnownUser(usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(clientId string, user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
```

```
 log.Println("To complete this example and successfully migrate a user to
Cognito, you must enter an email\n" +
  "you own that can receive a confirmation code.")
 return
}
codeDelivery, err := runner.cognitoActor.ForgotPassword(clientId, user.UserName)
if err != nil {
 panic(err)
}
log.Printf("\nA confirmation code has been sent to %v.",
*codeDelivery.Destination)
code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
 "(the password will not display as you type):", 8)
for !confirmed {
 log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
 err = runner.cognitoActor.ConfirmForgotPassword(clientId, code, user.UserName,
password)
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   password = runner.questioner.AskPassword("\nEnter another password:", 8)
  } else {
   panic(err)
  }
 } else {
  confirmed = true
 }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(clientId, user.UserName, password)
if err != nil {
 panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)

log.Println(strings.Repeat("-", 88))
```

```
}

// Run runs the scenario.
func (runner *MigrateUser) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 関数を使用して MigrateUser トリガーを処理します。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
```

```
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if output.Items == nil || len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

一般的なタスクを実行する構造体を作成します。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
```

```
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}
```

```
// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito アクションをラップする構造体を作成します。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
```

```
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
```

```
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
```

```
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
```

```
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB アクションをラップする構造体を作成します。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
```

```
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
```

```
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
```

```
 }
 return err
}
```

Logs CloudWatch アクションをラップする構造体を作成します。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
```

```
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

AWS CloudFormation アクションをラップする構造体を作成します。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

リソースをクリーンアップします。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
```

```
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して MFA を必要とする Amazon Cognito ユーザープールでユーザーをサインアップする

次のコード例は、以下を実行する方法を示しています。

- ユーザー名、パスワード、E メールアドレスでサインアップしてユーザーを確認します。
- MFA アプリケーションをユーザーに関連付けて、多要素認証を設定します。
- パスワードと MFA コードを使用してサインインします。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
```

```
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();
```

```
        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
 pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;
```

```
namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }



    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
```

```
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
```

```
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
```

```
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
```

```
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
```

```
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
```

```
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
```

```
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice

- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();
```

```
   std::cout
           << "This scenario will add a user to an Amazon Cognito user pool."
           << std::endl;
   const Aws::String userName = askQuestion("Enter a new username: ");
   const Aws::String password = askQuestion("Enter a new password: ");
   const Aws::String email = askQuestion("Enter a valid email for the user: ");

   std::cout << "Signing up " << userName << std::endl;

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
   bool userExists = false;
   do {
       // 1. Add a user with a username, password, and email address.
       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
```

```
        }
    } while (userExists);

    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
```

```
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
session, client)) {
        return false;
    }

    printAsterisksLine();
```

```
    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);
```

```
    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
```

```
                "Re-enter the 6 digit code displayed in the authenticator app:
");

       // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }

       std::cout << "You have successfully added a user to Amazon Cognito."
                 << std::endl;
   }

   if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
       // 10. Delete the user that you just added. (DeleteUser)
```

```
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                         outcome.GetResult().GetUserStatus()) << std::endl;
```

```
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、『AWS SDK for C++ API リファレンス』の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """
```

```
                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);
```

```
        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
```

```
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId)
{
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
```

```
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " +
response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
```

```
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
```

```
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

最良のエクスペリエンスを得るには、 GitHub リポジトリのクローンを作成し、この例を実行します。次のコードは、サンプルアプリケーション全体のサンプルを表しています。

```
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
```

```
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. A confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";
```

```
const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
```

```
    );
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};
```

```
const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
```

```
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
```

```
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};

export { adminRespondToAuthChallengeHandler };
```

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
```

```
    console.log(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge

- SignUp
- VerifySoftwareToken

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
 new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
```

```
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
```

```
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

シナリオで使用される Amazon Cognito 関数をラップするクラスを作成します。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
```

```
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

```
    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
```

```
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
```

```
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

```
   def confirm_mfa_device(
       self,
       user_name,
       device_key,
       device_group_key,
       device_password,
       access_token,
       aws_srp,
   ):
       """
       Confirms an MFA device to be tracked by Amazon Cognito. When a device is
       tracked, its key and password can be used to sign in without requiring a
new
       MFA code from the MFA application.

       :param user_name: The user that is associated with the device.
       :param device_key: The key of the device, returned by Amazon Cognito.
       :param device_group_key: The group key of the device, returned by Amazon
Cognito.
       :param device_password: The password that is associated with the device.
       :param access_token: The user's access token.
       :param aws_srp: A class that helps with Secure Remote Password (SRP)
                       calculations. The scenario associated with this example
uses
                       the warrant package.
       :return: True when the user must confirm the device. Otherwise, False.
When
                False, the device is automatically confirmed and tracked.
       """
       srp_helper = aws_srp.AWSSRP(
           username=user_name,
           password=device_password,
           pool_id="_",
           client_id=self.client_id,
           client_secret=None,
           client=self.cognito_idp_client,
       )
       device_and_pw = f"{device_group_key}{device_key}:{device_password}"
       device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
       salt = aws_srp.pad_hex(aws_srp.get_random(16))
       x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
```

```
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
code.

        Signing in with a tracked device requires that the client respond to the
SRP
```

```
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
```

```
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

シナリオを実行するクラスを作成します。この例では Amazon Cognito で追跡する MFA デバイスを登録し、追跡対象デバイスのパスワードと情報を使用してサインインする方法も示します。これにより、新しい MFA コードを入力する必要がなくなります。

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
 q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
 q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
 confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
```

```
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
```

```
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
```

```
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む

次のコード例は、Amazon Cognito ユーザー認証後に Lambda 関数を使用してカスタムアクティビティデータを書き込む方法を示しています。

- 管理者関数を使用して、ユーザーをユーザープールに追加します。
- `PostAuthentication` トリガーの Lambda 関数を呼び出すようにユーザープールを設定します。
- 新しいユーザーを Amazon Cognito にサインインします。
- Lambda 関数は、カスタム情報を CloudWatch Logs と DynamoDB テーブルに書き込みます。
- DynamoDB テーブルからカスタムデータを取得して表示し、リソースをクリーンアップします。

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(userPoolId string, tableName string)
 (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
```

```
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(userPoolId string,
 activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
```

```
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(clientId string, userName string, password
 string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(tableName string, userName
 string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
```

```
func (runner *ActivityLog) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(stackOutputs["TableName"], userName)

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 関数を使用して PostAuthentication トリガーを処理します。

```
const TABLE_NAME = "TABLE_NAME"
```

```
// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
```

```
    Time:       time.Now().Format(time.UnixDate),
   },
  }
  // Write to CloudWatch Logs.
  fmt.Printf("%#v", user)

  // Also write to an external system. This examples uses DynamoDB to demonstrate.
  userMap, err := attributevalue.MarshalMap(user)
  if err != nil {
   log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
  } else if len(userMap) == 0 {
   log.Printf("User info marshaled to an empty map.")
  } else {
   _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
    Item:      userMap,
    TableName: aws.String(tableName),
   })
   if err != nil {
    log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
   } else {
    log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
   }
  }

  return event, nil
}

func main() {
  sdkConfig, err := config.LoadDefaultConfig(context.TODO())
  if err != nil {
   log.Panicln(err)
  }
  h := handler{
   dynamoClient: dynamodb.NewFromConfig(sdkConfig),
  }
  lambda.Start(h.HandleRequest)
}
```

一般的なタスクを実行する構造体を作成します。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
```

```
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito アクションをラップする構造体を作成します。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
```

```
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
```

```
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
```

```
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
```

```
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
```

```
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB アクションをラップする構造体を作成します。

```


// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
```

```
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
```

```
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Logs CloudWatch アクションをラップする構造体を作成します。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

AWS CloudFormation アクションをラップする構造体を作成します。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

リソースをクリーンアップします。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
```

```
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon Cognito Sync のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon Cognito Sync を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SDK を使用した Amazon Cognito Sync のアクション AWS SDKs
  - AWS SDK または CLI ListIdentityPoolUsageで を使用する

SDK を使用した Amazon Cognito Sync のアクション AWS SDKs

次のコード例は、 AWS SDKs で個々の Amazon Cognito Sync アクションを実行する方法を示しています。これらは Amazon Cognito Sync API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。各例には GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、Amazon Cognito Sync API Reference (Amazon Cognito Sync API リファレンス) を参照してください。

例

- AWS SDK または CLI ListIdentityPoolUsageで を使用する

AWS SDK または CLI **`ListIdentityPoolUsage`**で を使用する

次の例は、`ListIdentityPoolUsage` を使用する方法を説明しています。

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API の詳細については、 ListIdentityPoolUsage AWS SDK for Rust API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

マルチテナントアプリケーションのベストプラクティス

Amazon Cognito ユーザープールは、Amazon Cognito クォータ内にとどまる必要がある大量のリクエストを生成するマルチテナントアプリケーションで動作します。カスタマーベースが大きくなったときにこの容量をスケールアップするには、追加のクォータ容量 を購入できます。

> **Note**
>
> Amazon Cognito クォータは、 AWS アカウント および ごとに適用されます AWS リージョン。これらのクォータは、アプリケーション内のすべてのテナント間で共有されます。Amazon Cognito サービスクォータを確認し、クォータがアプリケーション内の予想されるボリュームと予想されるテナント数を満たしていることを確認します。

このセクションでは、同じ リージョンと 内の Amazon Cognito リソース間でテナントを分離するために実装できる方法について説明します AWS アカウント。また、テナントを複数の AWS アカウント またはリージョンに分割し、それぞれに独自のクォータを与えることもできます。マルチリージョンマルチテナンシーのその他の利点には、可能な限り高いレベルの分離、グローバルに分散されたユーザーのネットワーク転送時間の短縮、組織内の既存の分散モデルの遵守などがあります。

単一リージョンのマルチテナンシーは、顧客や管理者にとっても利点があります。

次のリストでは、共有リソースを使用したマルチテナンシーの利点の一部について説明します。

マルチテナンシーの利点

共通ユーザーディレクトリ

マルチテナンシーは、顧客が複数のアプリケーションにアカウントを持つモデルをサポートします。サードパーティープロバイダーの ID を 1 つの一貫したユーザープールプロファイルにリンクできます。ユーザープロファイルがテナントに固有の場合、単一のユーザープールを持つマルチテナンシー戦略には、ユーザー管理へのエントリポイントが 1 つあります。

一般的なセキュリティ

共有ユーザープールでは、セキュリティの単一の標準を作成し、同じ高度なセキュリティ 、多要素認証 (MFA)、およびAWS WAF標準をすべてのテナントに適用できます。 AWS WAF ウェブ ACL は、関連付けるリソース AWS リージョン と同じ に存在する必要があるため、マルチテナンシーは複雑なリソースへの共有アクセスを提供します。マルチリージョンの Amazon Cognito

アプリケーションで一貫したセキュリティ設定を維持する場合は、リソース間で設定を複製する運用標準を適用する必要があります。

一般的なカスタマイズ

を使用して、ユーザープールと ID プールをカスタマイズできます AWS Lambda。ユーザープールでの Lambda トリガーの設定と、アイデンティティプールでの Amazon Cognito イベントの設定は複雑になる可能性があります。Lambda 関数は、ユーザープールまたは ID プール AWS リージョン と同じ に存在する必要があります。共有 Lambda 関数は、カスタム認証フロー、ユーザー移行、トークン生成、およびリージョン内の他の関数の標準を適用できます。

一般的なメッセージング

Amazon Simple Notification Service (Amazon SNS) では、ユーザーに SMS メッセージを送信する前に、リージョンで追加の設定が必要です。リージョンに含まれる Amazon Simple Email Service (Amazon SES) で検証された ID とドメインを使用して E メールメッセージを送信できます。

マルチテナンシーを使用すると、この設定とメンテナンスオーバーヘッドをすべてのテナント間で共有できます。Amazon SNS と Amazon SES はすべての で利用できるわけではないため AWS リージョン、リソースをリージョン間で分割するには追加の考慮事項が必要です。

カスタムメッセージングプロバイダー を使用すると、メッセージ配信を管理するための単一の Lambda 関数の一般的なカスタマイズが得られます。

ホストされた UI は、既に認証されたユーザーを認識するように、ブラウザにセッション Cookie を設定します。ユーザープールでローカルユーザーを認証すると、そのセッション Cookie は、同じユーザープール内のすべてのアプリケーションクライアントに対してユーザーを認証します。ローカルユーザーは、外部 IdP を介したフェデレーションなしに、ユーザープールディレクトリにのみ存在します。セッション Cookieは 1 時間有効です。セッション Cookie の有効期間を変更することはできません。

ホストされた UI セッション Cookie を使用してアプリケーションクライアント間でサインインを防ぐには、2 つの方法があります。

- ユーザーをテナントごとのユーザープールに分割します。
- ホストされた UI サインインを Amazon Cognito ユーザープール API サインインに置き換えます。

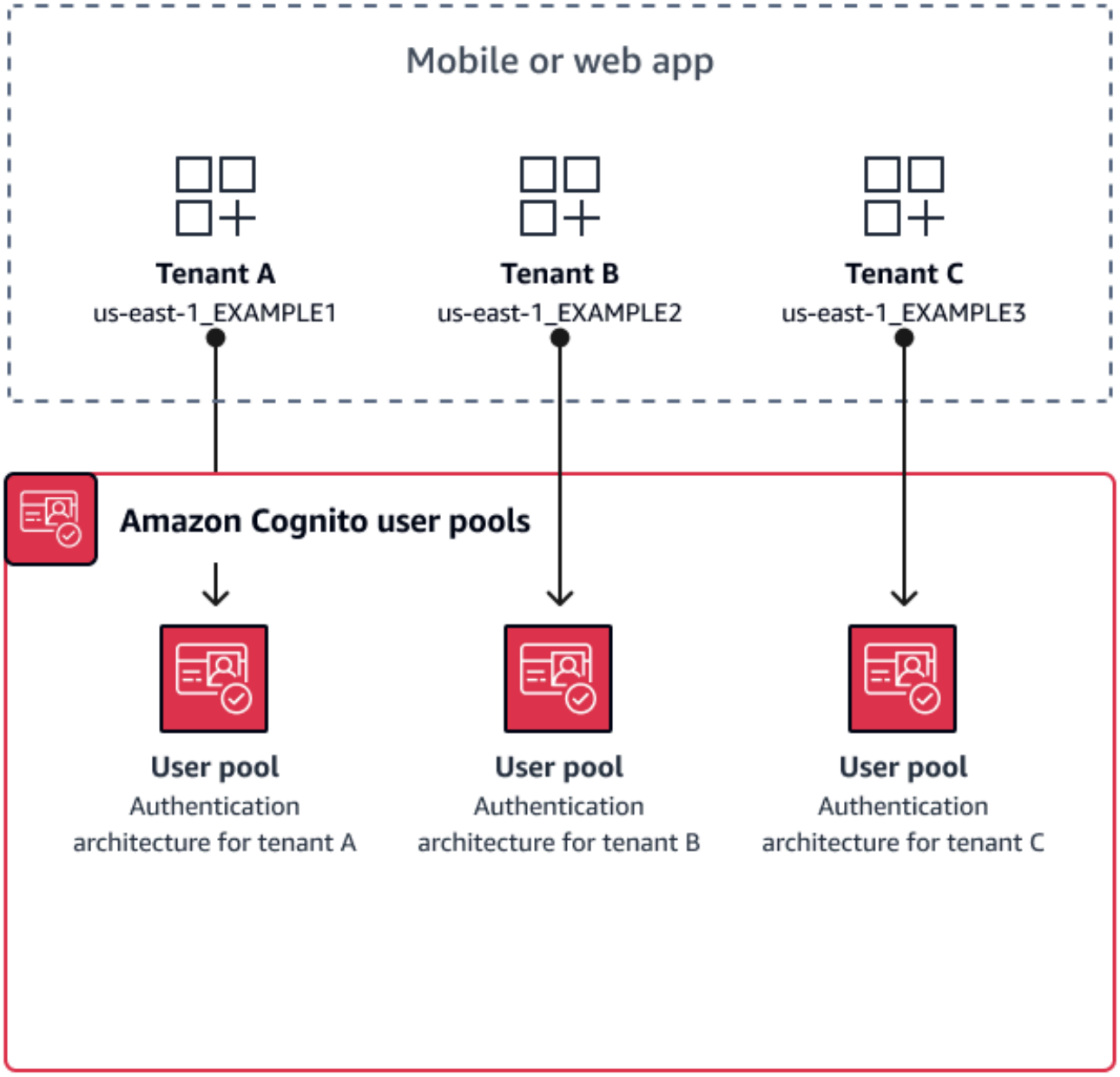
トピック

ユーザープールのマルチテナンシーのベストプラクティス

アプリ内のテナントごとにユーザープールを作成します。このアプローチは各テナントを最大限に分離します。テナントごとに異なる設定を実装することができます。ユーザープールによるテナント分離により、 user-to-tenant マッピングに柔軟性がもたらされます。同じユーザーに複数のプロファイルを作成することができます。ただし、ユーザーはそれぞれ、アクセスできるテナントごとに個別にサインアップする必要があります。

このアプローチを使用すると、テナントごとにホストされた UI を個別に設定し、アプリケーションのテナント固有のインスタンスにユーザーをリダイレクトできます。このアプローチを使用して、Amazon API Gatewayなどのバックエンドサービスと統合することもできます。

次の図は、専用ユーザープールを持つ各テナントを示しています。



ユーザープールのマルチテナンシーを実装するタイミング

分離とカスタマイズが主な懸念事項である場合。ユーザーとテナントの関係は、複数のユーザープールを持つアーキテクチャでは複雑な場合があります。2 つの教育テナントがある例を考えてみましょう。同じユーザーが、あるアプリではアクセスが制限された学生であり、別のアプリでは高レベルのアクセス許可を持つ教師である可能性があります。あるアプリでは MFA を要求し、別のアプリでは要求しないか、別のパスワードポリシーを持つ場合があります。ローカルユーザーはホストされた

UI を使用してユーザープール内の複数のアプリケーションクライアントにサインインできるため、複数のテナントにホストされた UI でサインインさせたい場合にも、ユーザープールのマルチテナンシーが最適です。

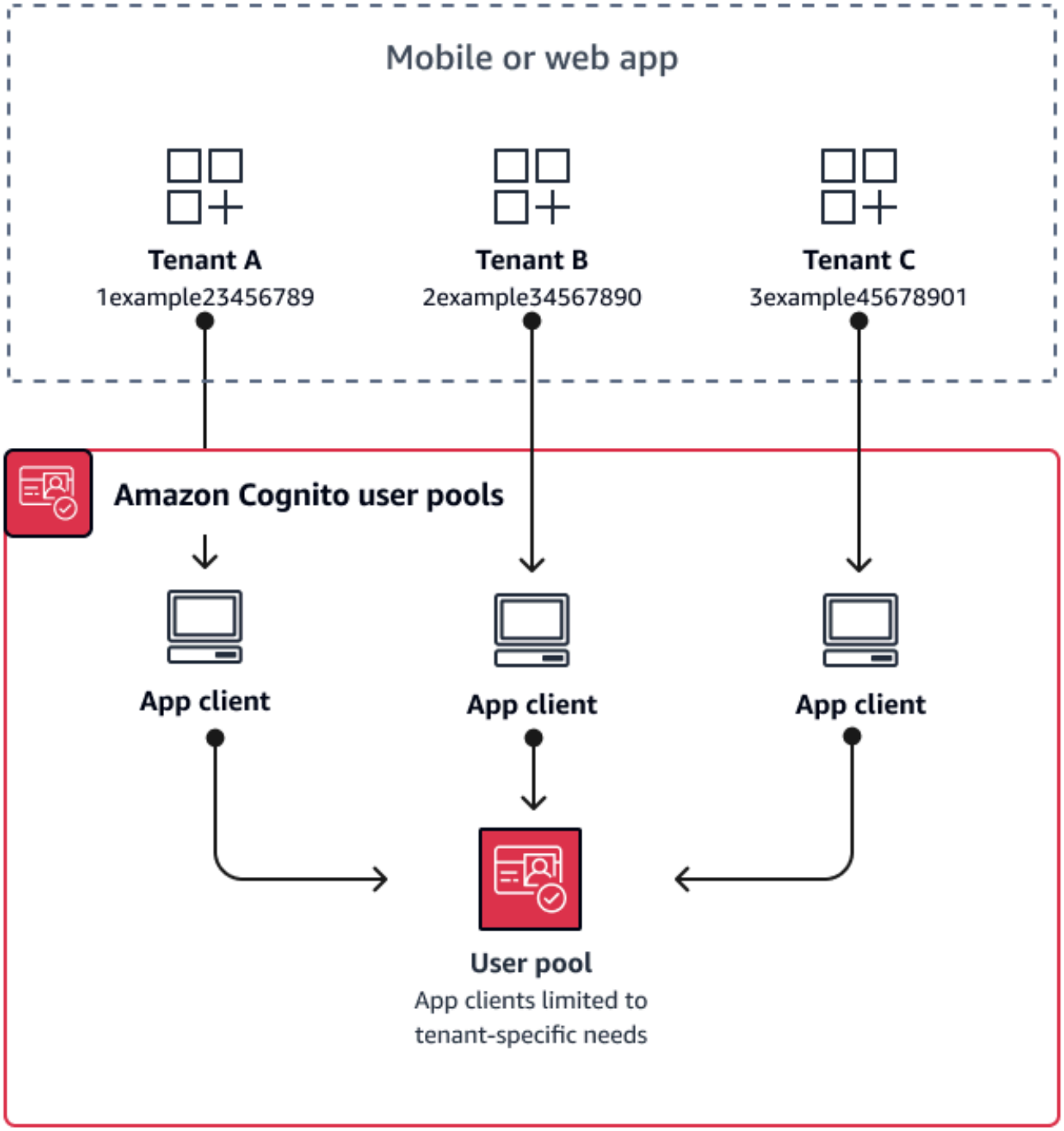
労力レベル

このアプローチを使用するには、高い開発労力と運用労力がかかります。アプリケーションファミリーの一貫性と予測可能な結果を確保するには、Amazon Cognito リソースをオートメーションツールと統合し、認証アーキテクチャがより複雑になるにつれてベースラインを維持する必要があります。アプリケーション用に単一の開始場所を作成する場合は、ユーザーを正しいリソースにルーティングする最初の決定をキャプチャするために、ユーザーインターフェイス (UI) 要素を構築する必要があります。

アプリケーションクライアントマルチテナンシーのベストプラクティス

アプリ内のテナントごとにアプリクライアントを作成します。アプリケーションクライアントマルチテナンシーを使用すると、テナントにリンクされたアプリケーションクライアントに任意のユーザーを割り当て、単一のユーザープロファイルを保持できます。ユーザープール内の ID プロバイダー (IdPs) の一部またはすべてをアプリケーションクライアントに割り当てることができるため、テナントアプリケーションクライアントはテナント固有の IdP によるサインインを許可できます。ユーザーが複数のテナントに存在する場合、プロファイルを複数の にリンク IdPs して一貫したユーザーエクスペリエンスを実現できます。

次の図は、共有ユーザープール内の専用アプリケーションクライアントを持つ各テナントを示しています。



アプリケーションクライアントマルチテナンシーを実装するタイミング

Lambda トリガー、パスワードポリシー、E メールおよび SMS メッセージの内容と配信方法など、ユーザープールレベルで設定のユニバーサル設定を選択できる場合。共有ユーザープールのユーザー

は任意のアプリケーションクライアントにサインインできるため、アプリケーションクライアントマルチテナンシーは app-client-specific IdPs または Amazon Cognito ユーザープール API でのサインインに最適です。アプリケーションクライアントマルチテナンシーは、ユーザーが複数のアプリケーション間で移行できるようにする環境にも適しています one-to-many。

労力レベル

アプリケーションクライアントマルチテナンシーには中程度の労力が必要です。アプリケーションクライアントマルチテナンシーの主な課題は、テナントがホストされた UI Cookie を提示し、アプリケーションを切り替える機能です。アプリケーションクライアントマルチテナンシーアーキテクチャでは、分離が必要な場所でホストされた UI サインインを避けます。アプリクライアントロジックが組み込まれたモバイルアプリまたはウェブアプリへのリンクを配信することも、ユーザーのテナンシーを決定する初期 UI 要素を構築することもできます。複数のユーザープールと ID プール間で設定を標準化して維持する必要がないため、労力のレベルは低くなります。

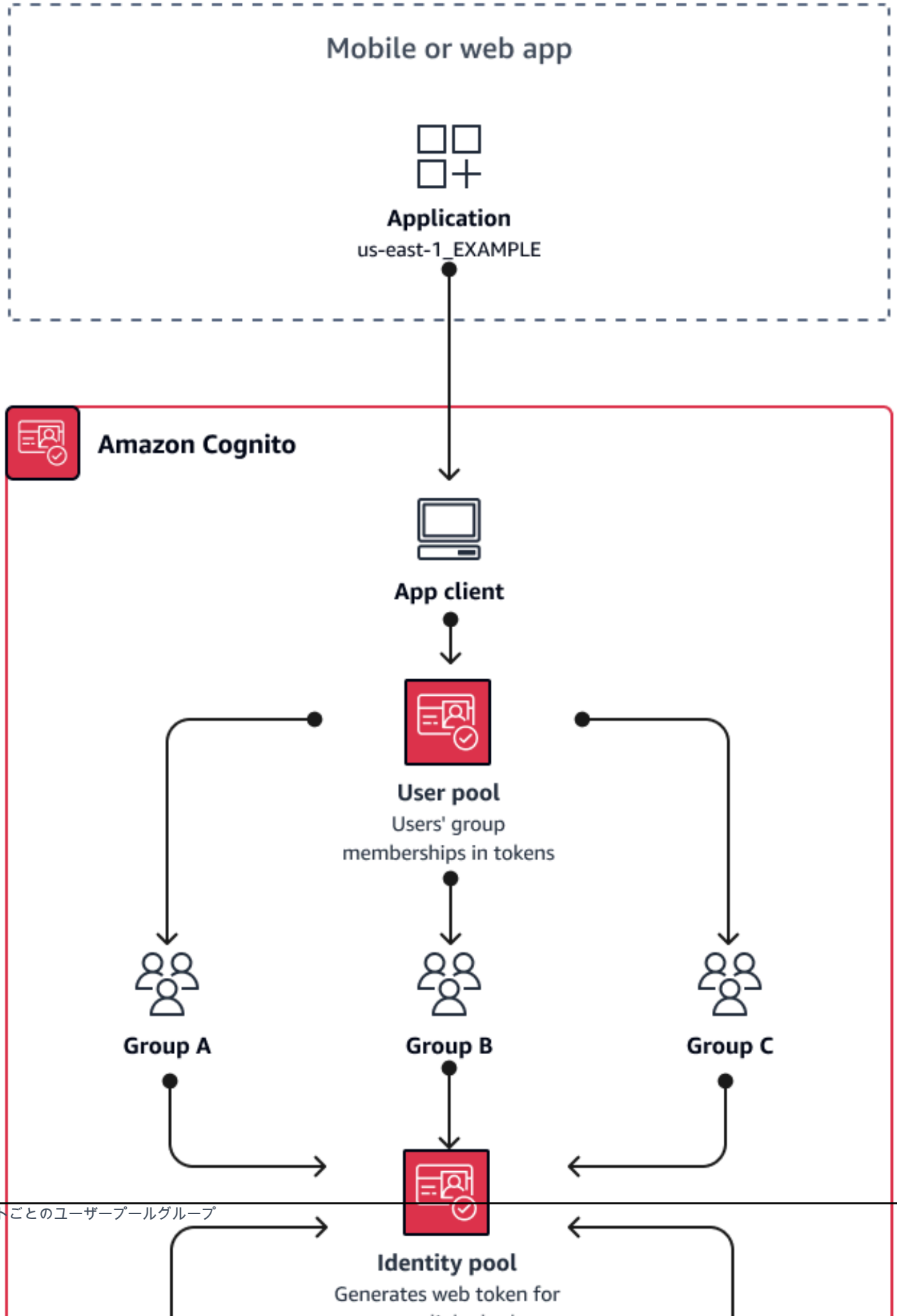
ユーザープールグループのマルチテナンシーのベストプラクティス

グループベースのマルチテナンシーは、アーキテクチャで ID プールを持つ Amazon Cognito ユーザープールが必要な場合に最適です。

ユーザープール ID とアクセストークンには `cognito:groups`クレームが含まれています。さらに、ID トークンには `cognito:roles`および `cognito:preferred_role`クレームが含まれます。アプリでの認証の主な結果が ID プールからの一時的な AWS 認証情報である場合、ユーザーのグループメンバーシップは、受け取る IAM ロールとアクセス許可を決定できます。

例として、それぞれがアプリケーションアセットを独自の Amazon S3 バケットに保存している 3 つのテナントを考えてみましょう。各テナントのユーザーを関連付けられたグループに割り当て、グループの優先ロールを設定し、そのロールにバケットへの読み取りアクセスを許可します。

次の図は、アプリケーションクライアントとユーザープールを共有するテナントと、IAM ロールの適格性を決定するユーザープール内の専用グループを示しています。


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for

グループマルチテナンシーを実装するタイミング

AWS リソースへのアクセスが主な懸念事項である場合。Amazon Cognito ユーザープールのユーザープールのグループは、ロールベースのアクセスコントロール (RBAC) のメカニズムです。ユーザープールに多数のグループを設定し、グループの優先順位で複雑な RBAC 決定を行うことができます。ID プールは、優先度が最も高いロール、グループクレーム内の任意のロール、またはユーザーのトークン内の他のクレームの認証情報を割り当てることができます。

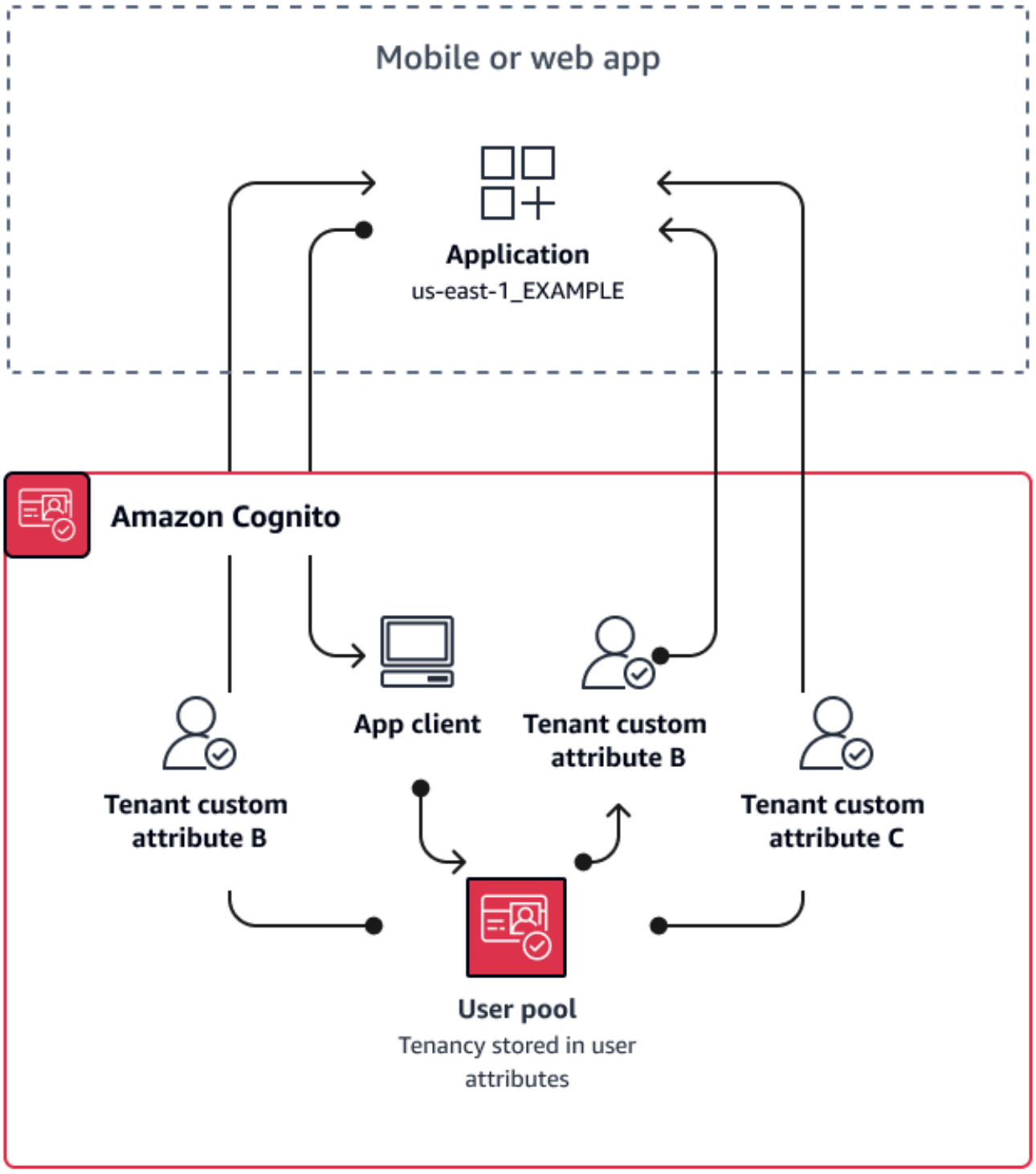
労力レベル

グループメンバーシップだけでマルチテナンシーを維持する労力のレベルは低くなります。ただし、IAM ロール選択の組み込み容量を超えてユーザープールグループのロールを拡張するには、ユーザーのトークンのグループメンバーシップを処理するアプリケーションロジックを構築し、クライアントで何をするかを決定する必要があります。Amazon Verified Permissions をアプリケーションと統合して、クライアント側の承認を決定できます。グループ識別子は、現在 Verified Permissions IsAuthorizedWithToken API オペレーションでは処理されませんが、グループメンバーのクレームなど、トークンの内容を解析するカスタムコードを開発できます。

カスタム属性マルチテナンシーのベストプラクティス

Amazon Cognito は、選択した名前のカスタム属性をサポートします。カスタム属性が役立つシナリオの 1 つは、共有ユーザープール内のユーザーのテナンシーを区別する場合です。などの属性に値をユーザーに割り当てるとcustom:tenantID、アプリケーションはそれに応じてテナント固有のリソースへのアクセスを割り当てることができます。テナント ID を定義するカスタム属性は、アプリクライアントに対してイミュータブルまたは読み取り専用にする必要があります。

次の図は、アプリケーションクライアントとユーザープールを共有するテナントと、それらが属するテナントを示すユーザープール内のカスタム属性を示しています。



カスタム属性がテナンシーを決定する場合、単一のアプリケーションまたはサインイン URL を配布できます。ユーザーがサインインすると、アプリはロードするアセット、適用するブランド、表示

する機能を決定するcustom:tenantIDクレームを処理できます。ユーザー属性からの高度なアクセスコントロールの決定については、Amazon Verified Permissions で ID プロバイダーとしてユーザープールを設定し、ID またはアクセストークンの内容からアクセス決定を生成します。

カスタム属性マルチテナンシーを実装するタイミング

テナンシーが表面レベルの場合。テナント属性は、ブランド化とレイアウトの結果に貢献できます。テナント間で大幅な分離を実現する場合、カスタム属性は最適な選択肢ではありません。MFA やホストされた UI ブランドなど、ユーザープールレベルまたはアプリクライアントレベルで設定する必要があるテナント間の違いは、カスタム属性が提供しない方法でテナント間の区別を作成する必要があります。ID プールを使用すると、ID トークンのカスタム属性クレームからユーザーから IAM ロールを選択することもできます。

労力レベル

カスタム属性のマルチテナンシーは、アプリケーションに対するテナントベースの承認決定の義務を転送するため、労力のレベルが高い傾向があります。OIDC クレームを解析するクライアント設定、または Amazon Verified Permissions を既に熟知している場合、このアプローチには最低レベルの労力が必要になる場合があります。

マルチテナンシーのセキュリティに関する推奨事項

アプリケーションの安全性を高めるために、以下を推奨します。

- Amazon Verified Permissions を使用してアプリのテナンシーを検証します。アプリケーションでユーザーのリクエストを許可する前に、ユーザープール、アプリケーションクライアント、グループ、またはカスタム属性の使用権限を調べるポリシーを構築します。Amazon Cognito ユーザープールを念頭に置いて Verified Permissions ID ソース AWS を作成しました。Verified Permissions には、マルチテナンシー管理に関する追加のガイダンスがあります。
- ドメインの一致に基づいたテナントへのユーザーアクセスを承認するために、検証済みの E メールアドレスのみを使用します。E メールアドレスと電話番号は、アプリが検証するか、外部 IdP が検証の証明を提供しない限り、信頼しないでください。これらの許可を設定する方法の詳細については、「属性の許可と範囲」を参照してください。
- テナントを識別するユーザープロファイル属性には、イミュータブルな、または読み取り専用のカスタム属性を使用します。変更不可能な属性の値は、ユーザーを作成するとき、またはユーザーがユーザープールにサインアップするときにのみ設定できます。また、アプリクライアントに属性への読み取り専用アクセスを許可します。

- テナントの外部 IdP とアプリケーションクライアント間の 1:1 マッピングを使用して、不正なクロステナントアクセスを防止します。外部 IdP によって認証され、有効な Amazon Cognito セッション Cookie を持つユーザーは、同じ IdP を信頼する他のテナントアプリケーションにアクセスできます。
- アプリケーションにテナントマッチングおよび認可ロジックを実装する場合は、テナントへのユーザーアクセスを認可する基準を変更できないようにユーザーを制限してください。また、フェデレーションのために外部 IdP が使用されている場合は、テナント ID プロバイダー管理者がユーザーアクセスを変更できないように制限してください。

一般的な Amazon Cognito シナリオ

このトピックでは、Amazon Cognito を使用するための 6 つの一般的なシナリオについて説明します。

Amazon Cognito の主な 2 つのコンポーネントは、ユーザープールと ID プールです。ユーザープールは、ウェブおよびモバイルユーザーにサインアップとサインインオプションを提供するユーザーディレクトリです。ID プールは、ユーザーに他の へのアクセスを許可するための一時的な AWS 認証情報を提供します AWS のサービス。

ユーザープールは、Amazon Cognito のユーザーディレクトリです。アプリユーザーは、ユーザープールから直接サインインすることも、サードパーティー ID プロバイダー (IdP) を介してフェデレーションすることもできます。ユーザープールは、Facebook、Google、Amazon、Apple を介したソーシャルサインイン、および OpenID Connect (OIDC) と SAML から返されるトークンの処理のオーバーヘッドを管理します IdPs。ユーザーが直接またはサードパーティーを通じてサインインするかどうかにかかわらず、ユーザープールのすべてのメンバーには、SDK を通じてアクセスできるディレクトリプロファイルがあります。

ID プールを使用すると、ユーザーは Amazon S3 や DynamoDB などの のサービスにアクセス AWS するための一時的な AWS 認証情報を取得できます。ID プールは、匿名ゲストユーザーと、サードパーティー を介したフェデレーションをサポートします IdPs。

トピック

- ユーザープールを使用して認証する
- ユーザープールを使用してサーバー側のリソースにアクセスする
- ユーザープールと共に API Gateway と Lambda を使用してリソースにアクセスする
- ユーザープールと ID プールを使用して AWS サービスにアクセスする
- サードパーティーで認証を行い、ID プールを使用して AWS サービスにアクセスする
- Amazon Cognito で AWS AppSync リソースにアクセスする

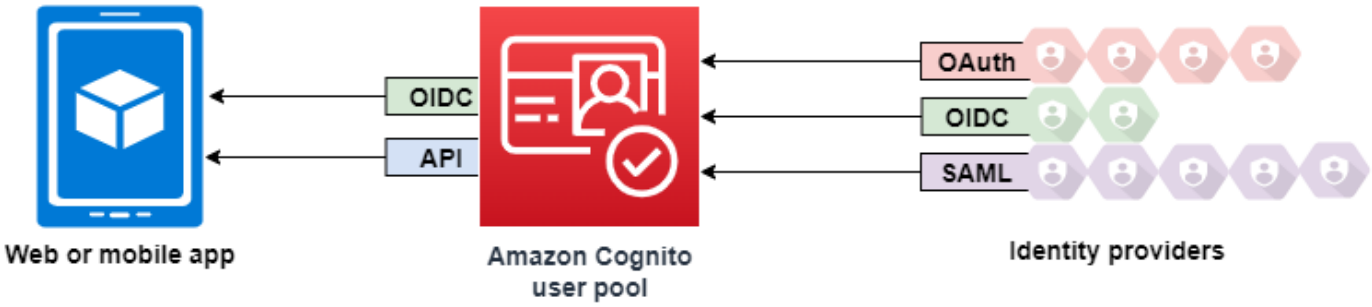
ユーザープールを使用して認証する

ユーザーがユーザープールを使用して認証できるようにすることが可能です。アプリユーザーは、ユーザープールから直接サインインすることも、サードパーティー ID プロバイダー (IdP) を介してフェデレーションすることもできます。ユーザープールは、Facebook、Google、Amazon、Apple

を介したソーシャルサインイン、および OpenID Connect (OIDC) と SAML から返されるトークンの処理のオーバーヘッドを管理します IdPs。

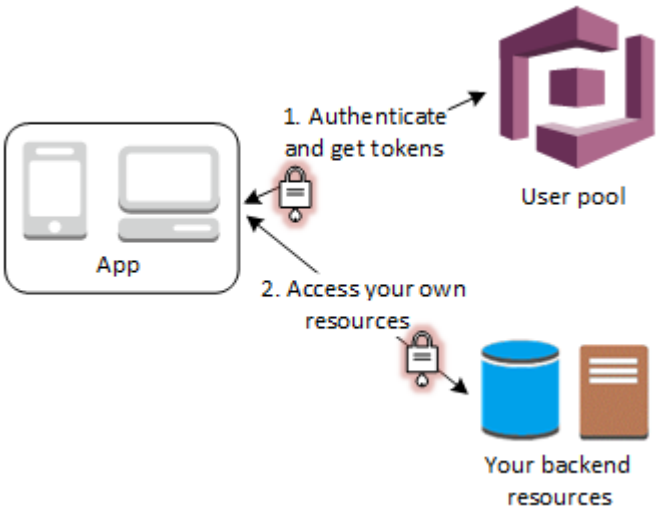
認証が正常に行われると、ウェブまたはモバイルアプリが Amazon Cognito からユーザープールトークンを受け取ります。これらのトークンを使用して、アプリケーションが他の AWS のサービスにアクセスできるようにする AWS 認証情報を取得できます。または、サーバー側のリソースまたは Amazon API Gateway へのアクセスを制御するためにトークンを使用することも可能です。

詳細については、「ユーザープール認証フロー」および「ユーザープールでのトークンの使用」を参照してください。



ユーザープールを使用してサーバー側のリソースにアクセスする

ユーザープールへのサインインが正常に行われると、ウェブまたはモバイルアプリが Amazon Cognito からユーザープールトークンを受け取ります。サーバー側のリソースへのアクセスを制御するには、これらのトークンを使用します。また、ユーザープールグループを作成して許可を管理したり、異なるタイプのユーザーを表したりすることもできます。グループを使用して、リソースをアクセス制御する方法の詳細については、「ユーザープールにグループを追加する」を参照してください。



ユーザープールのドメインを設定した後、Amazon Cognito が、アプリにサインアップおよびサインインページを追加できるようにするホストされたウェブの UI をプロビジョニングします。この OAuth 2.0 認証基盤を使用することで、独自のリソースサーバーを作成でき、ユーザーは保護されたリソースにアクセスできるようになります。詳細については、「スコープ、M2M 、およびリソースサーバーによる API 認証」を参照してください。

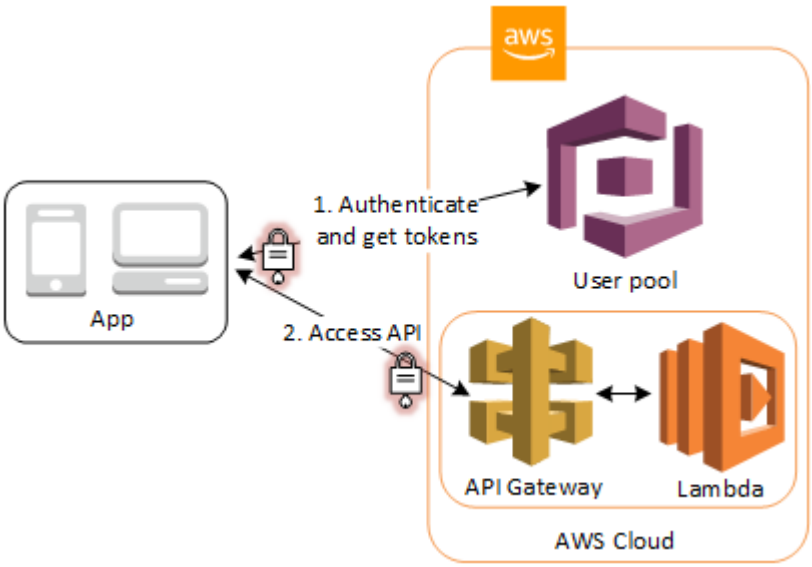
ユーザープール認証の詳細については、「ユーザープール認証フロー」および「ユーザープールでのトークンの使用」を参照してください。

ユーザープールと共に API Gateway と Lambda を使用してリソースにアクセスする

ユーザーが API Gateway 経由で API にアクセスできるようにすることが可能です。API Gateway は、正常に行われたユーザープール認証からのトークンを検証し、これらのトークンを Lambda 関数などのリソース、または独自の API へのアクセス権をユーザーに付与するために使用します。

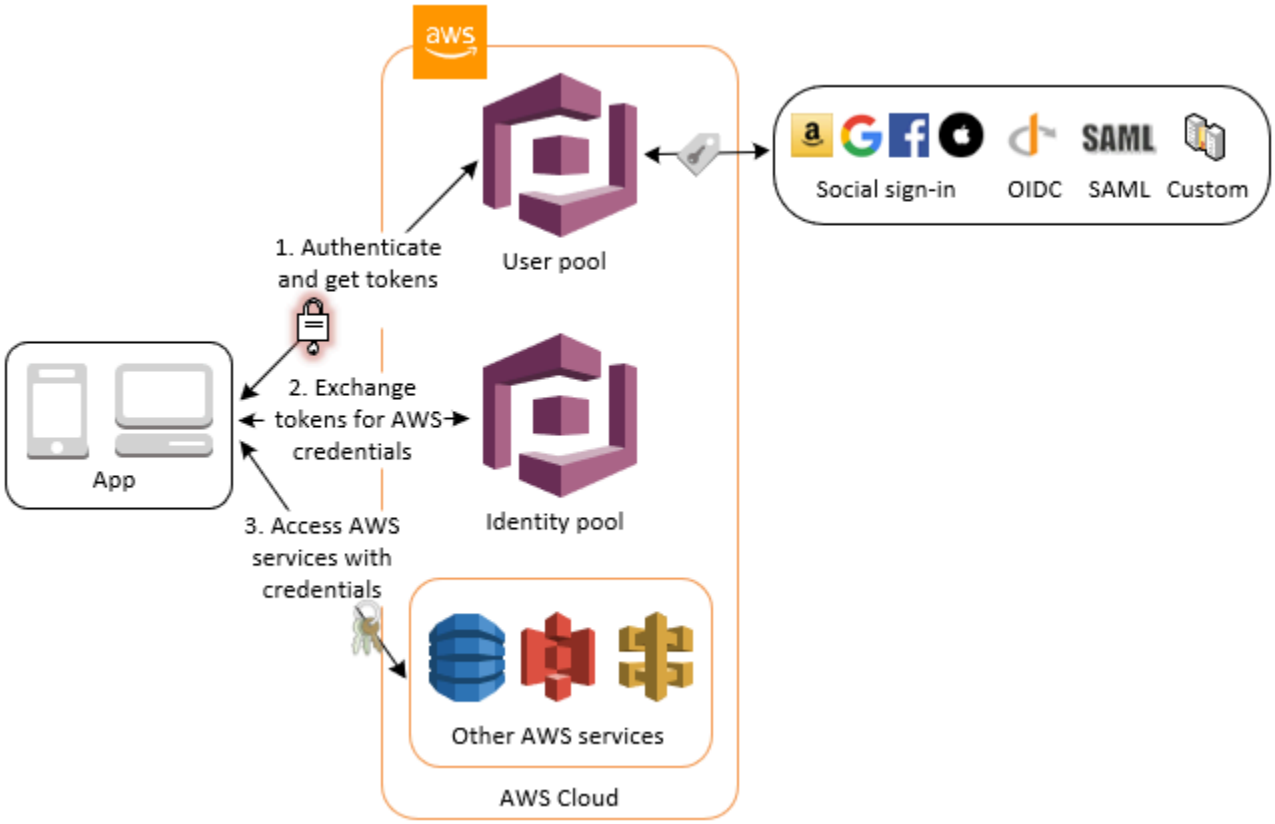
ユーザープール内のグループは、グループメンバーシップを IAM ロールにマップすることによって、API Gateway でアクセス許可を制御するために使用できます。ユーザーがメンバーとなっているグループは、アプリユーザーのサインイン時にユーザープールより付与される ID トークンに含まれます。ユーザープールグループの詳細については、「ユーザープールにグループを追加する」を参照してください。

Amazon Cognito オーソライザーの Lambda 関数による検証のために、API Gateway へのリクエストでユーザープールトークンを送信することができます。API Gateway の詳細については、「Amazon Cognito ユーザープールをオーソライザーとして使用して REST API へのアクセスを制御する」を参照してください。



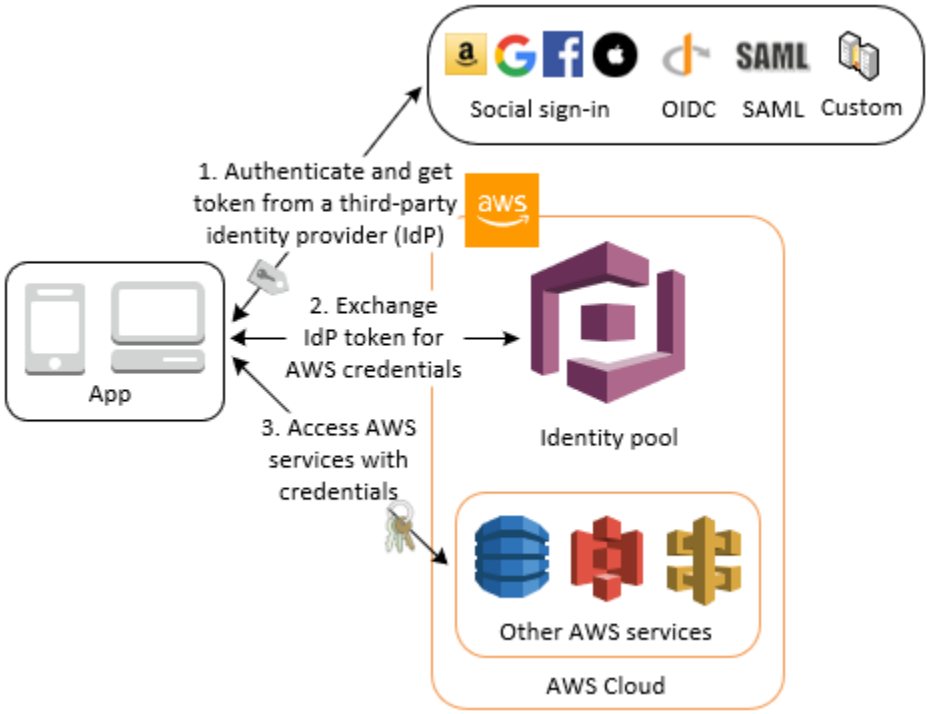
ユーザープールと ID プールを使用して AWS サービスにアクセスする

ユーザープールの認証が正常に行われると、アプリケーションが Amazon Cognito からユーザープールトークンを受け取ります。ID プールを使用して、他の AWS のサービスへの一時的なアクセスと交換できます。詳細については、「サインイン後の ID プール AWS のサービス を使用した へのアクセス」および「Amazon Cognito ID プールの開始方法」を参照してください。



サードパーティーで認証を行い、ID プールを使用して AWS サービスにアクセスする

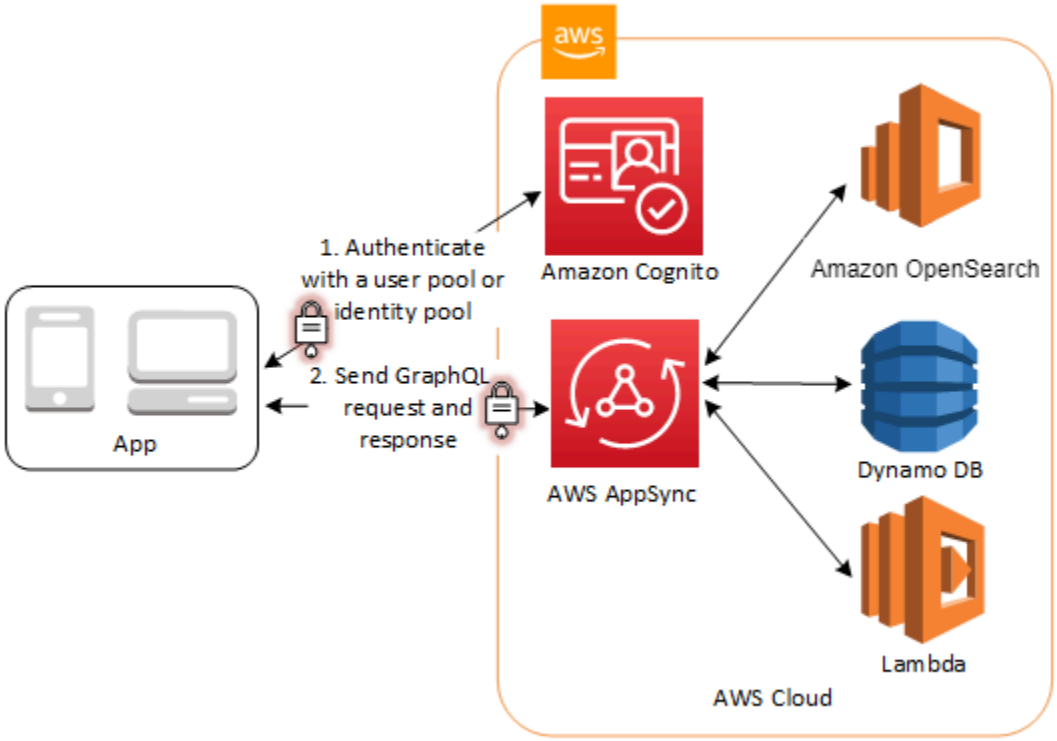
ID プールを介して AWS のサービスへのアクセスをユーザーに許可できます。ID プールには、サードパーティー ID プロバイダーによって認証されたユーザーからの IdP トークンが必要です (匿名ゲストの場合は何も必要ありません)。代わりに、アイデンティティプールは、他の AWS サービスにアクセスするために使用できる一時的な AWS 認証情報を付与します。詳細については、「Amazon Cognito ID プールの開始方法」を参照してください。



Amazon Cognito で AWS AppSync リソースにアクセスする

Amazon Cognito ユーザープール認証が成功したトークンを使用して、 AWS AppSync リソースへのアクセス権をユーザーに付与できます。詳細については、「AWS AppSync デベロッパーガイド」の「AMAZON_COGNITO_USER_POOLS 認証」を参照してください。

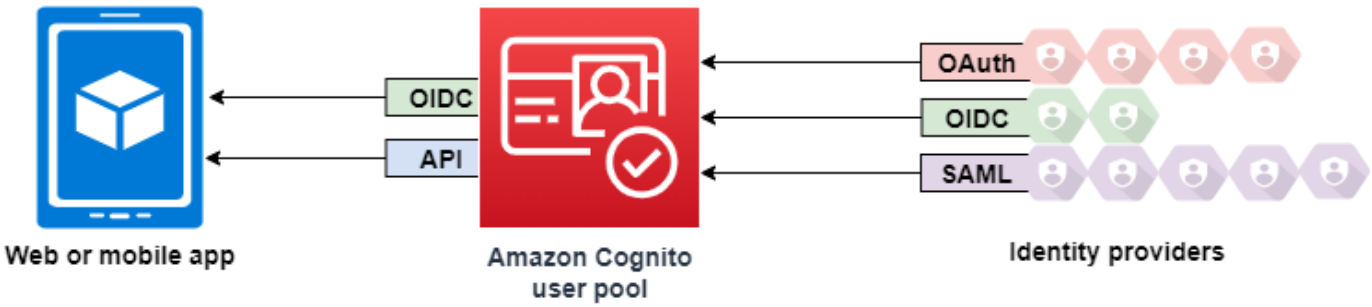
ID プールから受信した IAM 認証情報を使用して、 AWS AppSync GraphQL API へのリクエストに署名することもできます。「AWS_IAM 認証」を参照してください。


aws
1. Authenticate with a user pool or identity pool
2. Send GraphQL request and response
App
Amazon Cognito
AWS AppSync
Amazon OpenSearch
Dynamo DB
Lambda
AWS Cloud

Amazon Cognito user pools

Amazon Cognito ユーザープールは、ウェブおよびモバイルアプリケーションの認証と承認のためのユーザーディレクトリです。アプリケーションの観点から見ると、Amazon Cognito ユーザープールは OpenID Connect (OIDC) ID プロバイダー (OIDC) です。ユーザープールには、セキュリティ、ID フェデレーション、アプリ統合、ユーザーエクスペリエンスのカスタマイズなどの機能が何層も追加されます。

例えば、ユーザーのセッションが信頼できるソースからのものであることを確認できます。Amazon Cognito ディレクトリを外部 ID プロバイダーと組み合わせることができます。任意の AWS SDK を使用すると、アプリケーションに最適な API 認証モデルを選択できます。また、Amazon Cognito のデフォルトの動作を変更またはオーバーホールする AWS Lambda 関数を追加することもできます。



トピック

- 機能
- ユーザープールでの認証
- Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用
- ユーザープール設定の更新
- Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用
- スコープ、M2M 、およびリソースサーバーによる API 認証
- サードパーティー経由のユーザープールへのサインインの追加
- Lambda トリガーを使用したユーザープールワークフローのカスタマイズ
- Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用
- ユーザープール内のユーザーを管理する

- Amazon Cognito ユーザープールの E メール設定
- Amazon Cognito ユーザープール用の SMS メッセージ設定
- ユーザープールでのトークンの使用
- ユーザープール認証に成功した後のリソースへのアクセス
- Amazon Cognito ユーザープールのセキュリティ機能を使用する

機能

Amazon Cognito ユーザープには次の機能があります。

サインアップ

Amazon Cognito ユーザープールには、ユーザープールにユーザープロファイルを追加するためのユーザー主導型、管理者主導型、およびプログラムによる方法があります。Amazon Cognito ユーザープールは、以下のサインアップモデルをサポートしています。アプリでは、これらのモデルを任意に組み合わせて、アプリで使用できます。

> **Important**
>
> ユーザープールでユーザーサインアップを有効にすると、インターネット上のすべてのユーザーがアカウントにサインアップしてアプリケーションにサインインできるようになります。アプリケーションをパブリックサインアップで開く場合を除き、ユーザープールで自己登録を有効にしないでください。この設定を変更するには、ユーザープールコンソールのサインアップエクスペリエンスタブでセルフサービスサインアップを更新するか、CreateUserPool または UpdateUserPool API リクエスト AllowAdminCreateUserOnlyの の値を更新します。
> ユーザープールに設定できるセキュリティ機能については、「Amazon Cognito ユーザープールのセキュリティ機能を使用する」を参照してください。

1. ユーザーはアプリに情報を入力し、ユーザープールにネイティブなユーザープロファイルを作成できます。API サインアップオペレーションを呼び出して、ユーザープールにユーザーを登録できます。これらのサインアップオペレーションは、誰でも開くことも、クライアントシークレットまたは AWS 認証情報を使用して承認することもできます。
2. ユーザーを Amazon Cognito への情報の受け渡しを許可できるサードパーティーの IdP にリダイレクトできます。Amazon Cognito は、OIDC ID トークン、OAuth 2.0 `userInfo` データ、

および SAML 2.0 アサーションを処理して、ユーザープールのユーザープロファイルに入れます。Amazon Cognito に受信させる属性は、属性マッピングルールに基づいて制御します。

3. パブリックサインアップやフェデレーションサインアップをスキップして、独自のデータソースとスキーマに基づいてユーザーを作成できます。Amazon Cognito コンソールまたは API でユーザーを直接追加できます。CSV ファイルからのユーザーをインポートします。既存のディレクトリで新しいユーザーを検索し、既存のデータからユーザープロファイルを入力する just-in-time AWS Lambda 関数を実行します。

ユーザーがサインアップしたら、Amazon Cognito がアクセストークンと ID トークンにリストするグループにユーザーを追加できます。ID トークンをアイデンティティプールに渡すときに、ユーザープールグループを IAM ロールにリンクすることもできます。

関連トピック

- ユーザープール内のユーザーを管理する
- Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用
- SDK を使用した Amazon Cognito ID プロバイダーのコード例 AWS SDKs

サインイン

Amazon Cognito は、スタンドアロンのユーザーディレクトリおよび ID プロバイダー (IdP) として使用し、アプリケーションの ID プロバイダー (IdP) として使用できます。ユーザーは Amazon Cognito でホストされている UI を使用してサインインすることも、Amazon Cognito ユーザープール API を介して独自の UI を使用してサインインすることもできます。フロントエンドのカスタム UI の背後にあるアプリケーション層は、正当なリクエストを確認するためのいくつかの方法のいずれかを使用してバックエンドのリクエストを承認できます。

外部ディレクトリ (オプションで Amazon Cognito に組み込まれたユーザーディレクトリと組み合わせる) を使用してユーザーをサインインさせるには、次の統合を追加できます。

1. OAuth 2.0 ソーシャルサインインでサインインし、コンシューマーユーザーデータをインポートします。Amazon Cognito は、OAuth 2.0 による Google、Facebook、Amazon、および Apple へのサインインをサポートしています。
2. サインインして、SAML と OIDC サインインでエンタープライズユーザーデータをインポートします。Amazon Cognito は、任意の SAMID Connect (OIDC) ID プロバイダー (IdP) からのクレームを受け入れるように設定することもできます。

3. 外部ユーザープロファイルをネイティブユーザープロファイルにリンクします。リンクされたユーザーは、サードパーティーのユーザー ID を使用してサインインし、組み込みディレクトリのユーザーに割り当てたアクセス権を受け取ることができます。

関連トピック

- サードパーティー経由のユーザープールへのサインインの追加
- フェデレーションユーザーを既存のユーザープロファイルにリンクする

M achine-to-machine 認証

一部のセッションは human-to-machine インタラクションではありません。自動プロセスによって API へのリクエストを承認できるサービスアカウントが必要になる場合があります。OAuth 2.0 スコープで machine-to-machine 認証用のアクセストークンを生成するには、クライアント認証情報付与を生成するアプリケーションクライアントを追加できます。

関連トピック

- スコープ、M2M 、およびリソースサーバーによる API 認証

ホストされた UI

ユーザーインターフェイスを構築しない場合は、カスタマイズされた Amazon Cognito がホストする UI をユーザーに提供できます。ホストされている UI は、多要素認証 (MFA) およびパスワードリセットするためのウェブページです。ホストされた UI を既存のドメインに追加するか、 AWS サブドメインでプレフィックス識別子を使用できます。

関連トピック

- Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用
- ユーザープールのドメインを設定する

セキュリティ

ローカルユーザーは、SMS メッセージからのコードを使用して追加の認証要素を提供したり、多要素認証 (MFA) コードを生成するアプリを提供したりできます。アプリで MFA を設定して処理する

メカニズムを構築することも、ホスト UI に管理させることもできます。Amazon Cognito ユーザープールは、ユーザーが信頼できるデバイスからサインインするときに MFA をバイパスできます。

最初にユーザーに MFA を要求しない場合は、条件付きで要求できます。高度なセキュリティ機能により、Amazon Cognito は潜在的な悪意のあるアクティビティを検出し、ユーザーに MFA の設定を要求したり、ログインをブロックしたりすることができます。

ユーザープールへのネットワークトラフィックが悪意のあるものである可能性がある場合は、それをモニタリングし、 AWS WAF ウェブ ACLs でアクションを実行できます。

関連トピック

- ユーザープールに MFA を追加します
- ユーザープールにアドバンストセキュリティを追加する
- AWS WAF ウェブ ACL とユーザープールの関連付け

カスタマーユーザーエクスペリエンス

ユーザーのサインアップ、サインイン、またはプロファイル更新のほとんどの段階で、Amazon Cognito がリクエストを処理する方法をカスタマイズできます。Lambda トリガーを使用すると、ID トークンを変更したり、カスタム条件に基づいてサインアップリクエストを拒否したりできます。自分のカスタム認証フローを作成できます。

カスタム CSS とロゴをアップロードして、ホストされた UI をユーザーが使い慣れた外観にすることができます。

関連トピック

- Lambda トリガーを使用したユーザープールワークフローのカスタマイズ
- カスタム認証チャレンジの Lambda トリガー
- 組み込みのサインインおよびサインアップウェブページのカスタマイズ

モニタリングと分析

Amazon Cognito ユーザープールは、ホストされている UI へのリクエストを含む API リクエストを AWS CloudTrail に記録します。Amazon CloudWatch Logs でパフォーマンスメトリクスを確認し、Lambda トリガー CloudWatch を使用してカスタムログを にプッシュし、Service Quotas コンソールで API リクエストボリュームをモニタリングできます。

API リクエストのデバイスおよびセッションデータを Amazon Pinpoint キャンペーンに記録することもできます。Amazon Pinpoint では、ユーザーアクティビティの分析に基づいて、アプリからプッシュ通知を送信できます。

関連トピック

- を使用した Amazon Cognito API コールのログ記録 AWS CloudTrail
- および Service Quotas でのクォータ CloudWatch と使用状況の追跡
- Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用

Amazon Cognito アイデンティティプール統合

Amazon Cognito のもう半分はアイデンティティプールです。ID プールは AWS のサービス、ユーザーからの Amazon DynamoDB や Amazon S3 などの への API リクエストを承認およびモニタリングする認証情報を提供します。ユーザープール内のユーザーをどのように分類したかに基づいてデータを保護する ID ベースのアクセスポリシーを構築できます。アイデンティティプールは、ユーザープールの認証とは無関係に、さまざまな ID プロバイダーからのトークンや SAML 2.0 アサーションを受け入れることもできます。

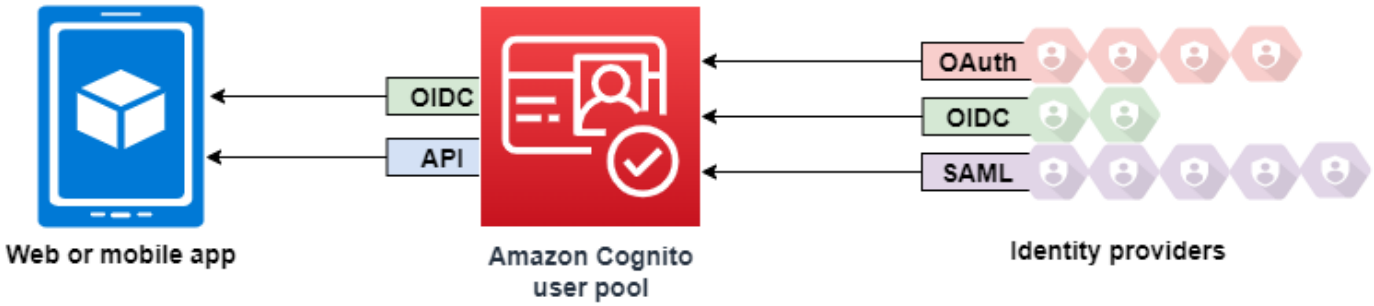
関連トピック

- サインイン後の ID プール AWS のサービス を使用した へのアクセス
- Amazon Cognito アイデンティティプール

ユーザープールでの認証

アプリユーザーは、ユーザープールから直接サインインすることも、サードパーティー ID プロバイダー (IdP) を介してフェデレーションすることもできます。ユーザープールは、Facebook、Google、Amazon、Apple を介したソーシャルサインイン、および OpenID Connect (OIDC) と SAML から返されるトークンの処理のオーバーヘッドを管理します IdPs。

認証が正常に行われると、Amazon Cognito がアプリにユーザープールトークンを返します。このトークンを使用して、独自のサーバー側のリソース、または Amazon API Gateway へのアクセス権をユーザーに付与できます。または、他の AWS のサービスにアクセスするための AWS 認証情報と交換することもできます。



ウェブまたはモバイルアプリのためのユーザープールトークンの処理と管理は、Amazon Cognito SDK を通じてクライアント側で提供されます。同様に、Mobile SDK for iOS と Mobile SDK for Android は、有効な (期限が切れていない) 更新トークンが存在し、ID トークンとアクセストークンの最小残存有効期間が 5 分になっている場合に ID トークンとアクセストークンを自動的に更新します。SDK および 、Android、iOS のサンプルコードについては、Amazon Cognito ユーザープール SDKs」を参照してください。 JavaScript

アプリユーザーが正常にログインすると、Amazon Cognito がセッションを作成し、認証されたユーザーの ID トークン、アクセストークン、および更新トークンを返します。

JavaScript

```
// Amazon Cognito creates a session which includes the id, access, and refresh
 tokens of an authenticated user.

var authenticationData = {
        Username : 'username',
        Password : 'password',
    };
    var authenticationDetails = new
 AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
    var poolData = { UserPoolId : 'us-east-1_ExaMPle',
        ClientId : '1example23456789'
    };
    var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
    var userData = {
        Username : 'username',
        Pool : userPool
    };
    var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
    cognitoUser.authenticateUser(authenticationDetails, {
        onSuccess: function (result) {
```

```
            var accessToken = result.getAccessToken().getJwtToken();

            /* Use the idToken for Logins Map when Federating User Pools with
 identity pools or when passing through an Authorization Header to an API Gateway
 Authorizer */
            var idToken = result.idToken.jwtToken;
        },

        onFailure: function(err) {
            alert(err);
        },

});
```

Android

```
// Session is an object of the type CognitoUserSession, and includes the id, access,
 and refresh tokens for a user.

String idToken = session.getIdToken().getJWTToken();
String accessToken = session.getAccessToken().getJWT();
```

iOS - swift

```
// AWSCognitoIdentityUserSession includes id, access, and refresh tokens for a user.

- (AWSTask<AWSCognitoIdentityUserSession *> *)getSession;
```

iOS - objective-C

```
// AWSCognitoIdentityUserSession includes the id, access, and refresh tokens for a
 user.

[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
 continueWithSuccessBlock:^id _Nullable(AWSTask<AWSCognitoIdentityUserSession *> *
 _Nonnull task) {
    // success, task.result has user session
    return nil;
}];
```

トピック

ユーザープール認証フロー

Amazon Cognito には、ユーザーを認証する方法がいくつか用意されています。ドメインの有無にかかわらず、すべてのユーザープールはユーザープール API でユーザーを認証できます。ユーザープールにドメインを追加すると、ユーザープール エンドポイントを使用できます。ユーザープール API は、API リクエストのさまざまな認証モデルとリクエストフローをサポートします。

ユーザーのアイデンティティを検証するため、Amazon Cognito では、パスワードの他にも新しいタイプのチャレンジが導入されています。Amazon Cognito 認証では、通常、次の順序で 2 つの API オペレーションを実装する必要があります。

Public authentication

1. InitiateAuth
2. RespondToAuthChallenge

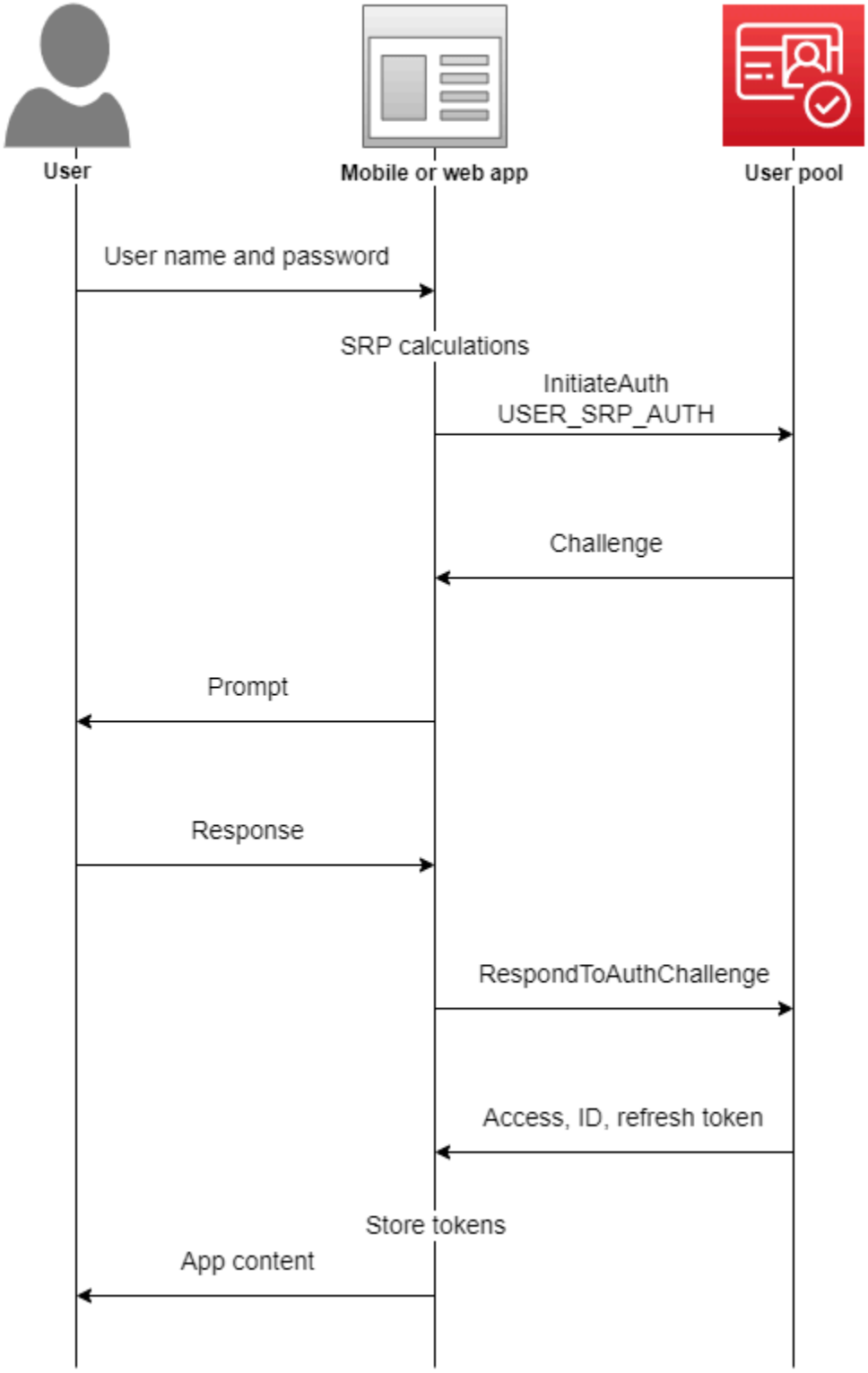
   `InitiateAuth` および `RespondToAuthChallenge` は、クライアント側のパブリックアプリケーションクライアントで使用する認証されていない API です。

Server-side authentication

1. AdminInitiateAuth
2. AdminRespondToAuthChallenge

   `AdminInitiateAuth` および `AdminRespondToAuthChallenge` は IAM 認証情報を必要とし、サーバー側の機密アプリクライアントに適しています。

認証が失敗するか、Amazon Cognito がユーザーにトークンを発行するまで、ユーザーは連続してチャレンジに回答して認証を進めます。カスタム認証フローをサポートするために、さまざまなチャレンジを含むプロセスで Amazon Cognito でこれらの手順を繰り返すことができます。


User
Mobile or web app
User pool
User name and password
SRP calculations
InitiateAuth
USER_SRP_AUTH
Challenge
Prompt
Response
RespondToAuthChallenge
Access, ID, refresh token
Store tokens
App content

通常、アプリケーションはプロンプトを表示してユーザーから情報を収集し、その情報を API リクエストで Amazon Cognito に送信します。ユーザープールでの `InitiateAuth` フローを考えてみます。このユーザープールには、多要素認証 (MFA) を使用してユーザーを設定済みです。

1. アプリケーションは、ユーザーにユーザー名とパスワードの入力を求めます。
2. ユーザー名とパスワードをパラメータとして `InitiateAuth` に含めます。
3. Amazon Cognito から `SMS_MFA` チャレンジとセッション識別子が返されます。
4. アプリケーションは、ユーザーにスマートフォンの MFA コードの入力を求めます。
5. そのコードとセッション ID を `RespondToAuthChallenge` リクエストに含めます。

ユーザープールの機能によっては、アプリケーションが Amazon Cognito からトークンを取得する前に、`InitiateAuth` へのいくつかのチャレンジに対応することが必要になります。Amazon Cognito は、各リクエストへの応答にセッション文字列を含めます。API リクエストを 1 つの認証フローにまとめるには、前のリクエストに対する応答のセッション文字列を、後続の各リクエストに含めます。デフォルトでは、セッション文字列が期限切れになる前に、ユーザーに各チャレンジを完了するまでに 3 分間が与えられます。この期間を調整するには、アプリケーションクライアントの認証フローセッション持続期間を変更します。次の手順では、アプリクライアントの構成で、この設定を変更する方法について説明します。

> **Note**
>
> 認証フローセッション期間の設定は、Amazon Cognito ユーザープール API による認証に適用されます。Amazon Cognito がホストする UI では、多要素認証の場合はセッション時間を 3 分、パスワードリセットコードの場合は 8 分に設定しています。

Amazon Cognito console

アプリクライアントの認証フローセッション持続期間を設定するには (AWS Management Console)

1. ユーザープールの [App integration] (アプリの統合) タブで、[App clients and analytics] (アプリクライアントと分析) コンテナからアプリクライアントの名前を選択します。
2. [アプリケーションクライアントに関する情報] コンテナで [編集] を選択します。

3. [Authentication flow session duration] (認証フローセッション持続期間) の値を、SMS MFA コードに必要な有効持続期間 (分単位) に変更します。これにより、ユーザーがアプリクライアントで認証チャレンジを完了するまでの時間も変更されます。
4. [変更の保存] をクリックします。

Amazon Cognito API

アプリクライアントの認証フローセッション持続期間を設定するには (Amazon Cognito API)

1. `DescribeUserPoolClient` リクエストから既存のユーザープール設定を使用して `UpdateUserPoolClient` リクエストを準備します。`UpdateUserPoolClient` リクエストには、既存のアプリクライアントのプロパティをすべて含める必要があります。
2. `AuthSessionValidity` の値を、SMS MFA コードに必要な有効持続期間 (分単位) に変更します。これにより、ユーザーがアプリクライアントで認証チャレンジを完了するまでの時間も変更されます。

アプリクライアントの詳細については、「ユーザープールアプリクライアント」を参照してください。

AWS Lambda トリガーを使用して、ユーザーの認証方法をカスタマイズできます。トリガーは独自のチャレンジを認証フローの一部として発行し検証を行います。

また、安全なバックエンドサーバーでは、管理者の認証フローを使用することもできます。ユーザー移行の認証フローを使用すると、ユーザーがパスワードをリセットしなくてもユーザー移行できます。

サインインの試行の失敗に対する、Amazon Cognito ロックアウト動作

認証されていないか、または IAM 認証でのサインインに 5 回失敗すると、Amazon Cognito はユーザーを 1 秒間ロックアウトします。ロックアウトの期間は、試行が 1 回失敗するたびに 2 倍になり、最大で約 15 分になります。ロックアウト期間中に試行すると `Password attempts exceeded` 例外が生成され、その後のロックアウト期間の長さには影響しません。サインイン試行の累積失敗回数 n (`Password attempts exceeded` 例外を含まない) に対して、Amazon Cognito はユーザーを 2^(n-5) 秒間ロックアウトします。ロックアウトを n=0 初期状態にリセットするには、ユーザーは、ロックアウト期間後にサインインに成功するか、連続 15 分間、サインイン試行を開始してはなりません。この動作は変更される可能性があります。この動作は、パスワードベースの認証も実行しない限り、カスタムチャレンジに適用されません。

トピック

クライアント側の認証フロー

AWS Amplify または AWS SDK で作成したユーザークライアント側のアプリケーションの場合、以下の手順で動作します。

1. ユーザーがアプリにユーザー名とパスワードを入力します。
2. アプリケーションが、ユーザーのユーザー名と SRP (Secure Remote Password) 詳細を使用して、`InitiateAuth` オペレーションを呼び出します。

   この API オペレーションは、認証のパラメータを返します。

   > **Note**
   >
   > アプリは、 AWS SDK に組み込まれている Amazon Cognito SRP 機能を使用して SRP の詳細を生成します。

3. アプリが `RespondToAuthChallenge` 操作を呼び出します。呼び出しが成功すると、Amazon Cognito からユーザーのトークンが返され、認証フローが完了します。

   Amazon Cognito に必要なチャレンジが別に存在する場合は、`RespondToAuthChallenge` を呼び出してもトークンは返されません。代わりに、呼び出しによってセッションが返されます。

4. `RespondToAuthChallenge` からセッションが返された場合、アプリは `RespondToAuthChallenge` を再度呼び出します。今回の呼び出しには、セッションとチャレンジのレスポンス (MFA コードなど) が使用されます。

サーバー側の認証フロー

ユーザーアプリがなく、その代わりに Java、Ruby、Node.js のセキュアバックエンド、またはサーバー側のアプリを使用する場合は、Amazon Cognito ユーザープールの認証済みのサーバー側 API を使用できます。

サーバー側のアプリでのユーザープール認証は、クライアント側のアプリでの認証に似ています。次の点が異なります。

- サーバー側のアプリは、`InitiateAuth` の代わりに `AdminInitiateAuth` API 操作を呼び出します。このオペレーションには、 `cognito-idp:AdminInitiateAuth`および を含むアクセス許可を持つ AWS 認証情報が必要です`cognito-idp:AdminRespondToAuthChallenge`。オペレーションからは、必要な認証パラメータが返されます。
- サーバー側のアプリに認証パラメータが設定された後に、`AdminRespondToAuthChallenge` API オペレーションが (`RespondToAuthChallenge` の代わりに) 呼び出されます。`AdminRespondToAuthChallenge` API オペレーションは、 AWS 認証情報を指定した場合にのみ成功します。

AWS 認証情報を使用した Amazon Cognito API リクエストの署名の詳細については、「 AWS 全般のリファレンス」の「署名バージョン 4 の署名プロセス」を参照してください。

`AdminInitiateAuth` および `AdminRespondToAuthChallenge` API オペレーションは、次のいずれかの方法で明示的に有効にしない限り、管理者サインイン用の username-and-password ユーザー認証情報を受け入れることはできません。

- `CreateUserPoolClient` または `UpdateUserPoolClient` を呼び出すときは、`ExplicitAuthFlow` パラメータに `ALLOW_ADMIN_USER_PASSWORD_AUTH` (以前は `ADMIN_NO_SRP_AUTH` と呼ばれています) を含めます。
- アプリクライアントの認証フローのリストに `ALLOW_ADMIN_USER_PASSWORD_AUTH` を追加します。ユーザープールの [App integration] (アプリ統合) タブの [App clients and analytics] (アプリクライアントと分析) でアプリクライアントを設定します。詳細については、「ユーザープールアプリクライアント」を参照してください。

カスタム認証フロー

Amazon Cognito ユーザープールでは、カスタム認証フローを使用することもできます。これは、AWS Lambda トリガーを使用してチャレンジ/レスポンスベースの認証モデルを作成するのに役立ちます。

> **Note**
>
> カスタム認証フローでは、侵害された認証情報やアダプティブ認証にアドバンストセキュリティ機能を使用することはできません。詳細については、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

カスタム認証フローを使用すると、チャレンジとレスポンスサイクルをカスタマイズすることが可能になり、さまざまな要件に対応できます。このフローは、使用する認証のタイプを示す `InitiateAuth` API 操作の呼び出しから始まり、初期の認証パラメータを提供します。Amazon Cognito は、以下のいずれかの情報を使用して `InitiateAuth` コールに応答します。

- ユーザーへのチャレンジ、ならびにセッションおよびパラメータ。
- ユーザーが認証に失敗した場合はエラー。
- `InitiateAuth` コールで渡されたパラメータがユーザーのサインインに十分な場合は、ID、アクセス権限、更新トークン。(通常、ユーザーまたはアプリケーションは最初にチャレンジに応答する必要がありますが、カスタムコードはこれを判定する必要があります。)

Amazon Cognito がチャレンジで `InitiateAuth` コールに応答する場合、アプリは追加の入力を収集して、`RespondToAuthChallenge` 操作を呼び出します。このコールは、チャレンジ応答を提供し、セッションを返します。Amazon Cognito は、`RespondToAuthChallenge` コールに対しても `InitiateAuth` コール同様の対応をします。ユーザーがサインインしている場合、Amazon Cognito はトークンを提供します。ユーザーがサインインしていない場合、Amazon Cognito は別のチャレンジまたはエラーを提供します。Amazon Cognito が別のチャレンジを返した場合、ユーザーが正常にサインインするかエラーが返されるまで、シーケンスが繰り返され、アプリは `RespondToAuthChallenge` を呼び出します。`InitiateAuth` API 操作と `RespondToAuthChallenge` API 操作に関する詳細については、API ドキュメントに記載されています。

組み込みの認証フローとチャレンジ

Amazon Cognito には、SRP (Secure Remote Password) プロトコルを通じてユーザー名とパスワードを検証する標準的な認証フロー用として、`AuthFlow` 値と `ChallengeName` 値が組み込まれています。 AWS SDKsには、Amazon Cognito によるこれらのフローのサポートが組み込まれています。

フローは USER_SRP_AUTH を AuthFlow として InitiateAuth に送信することから始まります。また、AuthParameters の USERNAME および SRP_A の値を送信します。InitiateAuth コールが成功した時点で、レスポンスのチャレンジパラメータには PASSWORD_VERIFIER が ChallengeName および SRP_B として含められています。次に、アプリは RespondToAuthChallenge を呼び出します。この呼び出しには PASSWORD_VERIFIER ChallengeName と、ChallengeResponses の必要なパラメータが使用されます。RespondToAuthChallenge の呼び出しが成功し、ユーザーのサインインが受け入れられると、Amazon Cognito によりトークンが発行されます。ユーザーの多要素認証 (MFA) を有効にした場合は、Amazon Cognito は、SMS_MFA の ChallengeName を返します。アプリは、RespondToAuthChallenge への別の呼び出しを通じて、必要なコードを提供できます。

カスタム認証フローとチャレンジ

アプリはカスタム認証フローを開始するために、InitiateAuth を CUSTOM_AUTH として Authflow を呼び出すことができます。カスタム認証フローで、3 つの Lambda トリガーがチャレンジとレスポンスの検証を制御します。

- DefineAuthChallenge Lambda トリガーは、以前のチャレンジとレスポンスのセッション配列を入力として使用します。そして、次のチャレンジ名と、ユーザーが認証済みで、トークンを付与できるかどうかを示すブール値を生成します。この Lambda トリガーは、チャレンジを通じてユーザーのパスを制御するステートマシンです。
- CreateAuthChallenge Lambda トリガーは、チャレンジ名を入力として使用し、チャレンジとパラメータを生成してレスポンスを評価します。DefineAuthChallenge が次のチャレンジとして CUSTOM_CHALLENGE を返すと、認証フローは、CreateAuthChallenge を呼び出します。CreateAuthChallengeLambda トリガーは、チャレンジメタデータパラメータで次のタイプのチャレンジを渡します。
- VerifyAuthChallengeResponse Lambda 関数は、レスポンスを評価し、レスポンスが有効であるかどうかを示すブール値を返します。

カスタム認証フローでは、SRP パスワードの検証や SMS を介した MFA などの、内部定義されたチャレンジを組み合わせて使用することもできます。CAPTCHA や秘密の質問などのカスタムチャレンジを使用できます。

カスタム認証フローにおける SRP パスワードの検証の使用

カスタム認証フローに SRP を含める場合には、最初に SRP を処理する必要があります。

- カスタムフローで SRP パスワードの検証を開始する場合、アプリは InitiateAuth を CUSTOM_AUTH として Authflow を呼び出します。AuthParameters マップで、アプリケーションからのリクエストは、SRP_A: (SRP A 値) および CHALLENGE_NAME: SRP_A を含んでいます。
- CUSTOM_AUTH フローは、challengeName: SRP_A および challengeResult: true の初期セッションにより、DefineAuthChallenge の Lambda トリガーを呼び出します。Lambda 関数は、challengeName: PASSWORD_VERIFIER、issueTokens: false、および failAuthentication: false により、応答します。
- 次にアプリケーションは、(challengeName: PASSWORD_VERIFIER、そして challengeResponses マップ内の SRP に必要な他のパラメータを使用して) RespondToAuthChallenge を呼び出す必要があります。
- Amazon Cognito がパスワードを検証すると、challengeName: PASSWORD_VERIFIER と challengeResult: true の 2 番目のセッションで、RespondToAuthChallenge により Lambda トリガーの DefineAuthChallenge が呼び出されます。その時点で DefineAuthChallenge Lambda トリガーは、challengeName: CUSTOM_CHALLENGE を使用して応答することでカスタムチャレンジを開始します。
- MFA がユーザーに対して有効になっている場合、Amazon Cognito がパスワードを確認した後、ユーザーは MFA の設定またはサインインを要求されます。

> **Note**
>
> Amazon Cognito がホストするサインインウェブページは、カスタム認証チャレンジの Lambda トリガー をアクティブ化できません。

サンプルコードを含めた Lambda トリガーの詳細については、「Lambda トリガーを使用したユーザープールワークフローのカスタマイズ」を参照してください。

管理認証フロー

認証のベストプラクティスは、カスタム認証フローで説明されている API オペレーションを (SRP でのパスワード検証とともに) 使用することです。 AWS SDKs はこのアプローチを使用し、このアプローチは SRP の使用に役立ちます。ただし、SRP に関する計算を回避したい場合は、セキュアなバックエンドサーバーでは、代替の管理者 API オペレーションセットを利用できます。これらバックエンド管理の実装では、InitiateAuth の代わりに AdminInitiateAuth を使用します。同時に、RespondToAuthChallenge の代わりには AdminRespondToAuthChallenge を使用し

ます。パスワードはプレーンテキストとして送信できるため、これらのオペレーションを使用するときに SRP 計算を実行する必要はありません。以下がその例です。

```
AdminInitiateAuth Request {
    "AuthFlow":"ADMIN_USER_PASSWORD_AUTH",
    "AuthParameters":{
        "USERNAME":"<username>",
        "PASSWORD":"<password>"
        },
    "ClientId":"<clientId>",
    "UserPoolId":"<userPoolId>"
}
```

これらの管理認証オペレーションには、開発者の認証情報が必要であり、 AWS Signature Version 4 (SigV4) の署名プロセスが使用されます。これらのオペレーションは、Lambda 関数にとって便利な、標準の AWS SDK (Node.js を含む) 内での使用が可能です。これらのオペレーションを使用し、同時にパスワードをプレーンテキストで受け入れるには、コンソールでそれらをアプリケーション向けに有効化する必要があります。または、ADMIN_USER_PASSWORD_AUTH または ExplicitAuthFlow への呼び出しで CreateUserPoolClient をUpdateUserPoolClient パラメータに渡すことができます。InitiateAuth および RespondToAuthChallenge オペレーションでは、ADMIN_USER_PASSWORD_AUTH AuthFlow は受け入れられません。

AdminInitiateAuth からのレスポンス ChallengeParameters では、USER_ID_FOR_SRP 属性 (存在する場合) に、ユーザーのエイリアス (E メールアドレスや電話番号など) ではなく、実際のユーザー名が含まれています。AdminRespondToAuthChallenge への呼び出しの ChallengeResponses では、このユーザー名を USERNAME パラメータで渡す必要があります。

> **Note**
>
> バックエンド管理者実装は、管理者認証フローを使用しているため、フローはデバイス追跡をサポートしていません。デバイス追跡が有効にしている場合、管理者認証は成功しますが、アクセストークンの更新に関する呼び出しはいずれも失敗します。

ユーザー移行の認証フロー

ユーザー移行用 Lambda トリガーは、レガシーのユーザー管理システムからユーザープールにユーザーを移行する際に役立ちます。USER_PASSWORD_AUTH 認証フローを選択している場合には、移

行中のユーザーがパスワードをリセットする必要はありません。このフローは、認証中に暗号化された SSL 接続を介してユーザーのパスワードをサービスに送信します。

すべてのユーザーの移行が完了したら、フローをよりセキュアな SRP フローに切り替えます。SRP フローは、ネットワーク経由でパスワードを送信しません。

Lambda トリガーの詳細については、「Lambda トリガーを使用したユーザープールワークフローのカスタマイズ」を参照してください。

Lambda トリガーを使用したユーザーの移行の詳細については、「ユーザー移行の Lambda トリガーを使用したユーザープールへのユーザーのインポート」を参照してください。

ユーザープールアプリクライアント

ユーザープールアプリケーションクライアントは、Amazon Cognito で認証される 1 つのモバイルアプリケーションまたはウェブアプリケーションを操作するユーザープール内の設定です。アプリケーションクライアントは、認証済みおよび未認証の API オペレーションを呼び出したり、ユーザーの属性の一部またはすべてを読み取ったり変更したりできます。アプリでは、登録、サインイン、パスワードを忘れた場合の処理を行う際に、アプリクライアントに対して自分自身を識別する必要があります。これらの API リクエストには、アプリクライアント ID による自己識別と、オプションのクライアントシークレットによる認証が含まれていなければなりません。認証されたクライアントアプリのみがこれらの認証されていない API を呼び出すことができるように、アプリクライアント ID またはシークレットをセキュア化する必要があります。さらに、認証された API リクエストに AWS 認証情報で署名するようにアプリを設定する場合は、ユーザーの検査から認証情報を保護する必要があります。

ユーザープールには複数のアプリを作成できます。アプリクライアントは、アプリのコードプラットフォームにリンクされている場合もあれば、ユーザープール内の別のテナントにリンクされている場合もあります。例えば、サーバー側アプリケーション用のアプリと、別の Android アプリを作成することができます。各アプリには独自のアプリのクライアント ID があります。

アプリクライアントの種類

Amazon Cognito でアプリクライアントを作成する際、標準の OAuth クライアントタイプであるパブリッククライアントと機密クライアントに基づいたオプションを事前に入力できます。クライアントシークレットを持つ機密クライアントを設定します。クライアントの種類の詳細については、IETF RFC 6749 #2.1 を参照してください。

パブリッククライアント

パブリッククライアントは、ブラウザまたはモバイルデバイスで実行されます。信頼できるサーバー側リソースがないため、クライアントシークレットはありません。

機密クライアント

機密クライアントには、認証されていない API オペレーションのクライアントシークレットで信頼できるサーバー側のリソースがあります。アプリケーションは、バックエンドサーバー上でデーモンまたはシェルスクリプトとして実行されることがあります。

クライアントシークレット

クライアントシークレットまたはクライアントパスワードは、アプリクライアントへのすべての API リクエストでアプリが使用する必要がある固定文字列です。アプリクライアントには、アプリクライアントには、`client_credentials` 付与を実行するためのクライアントシークレットが必要です。詳細については、IETF RFC 6749 #2.3.1 を参照してください。

アプリを作成した後は、シークレットを変更できません。シークレットキーを更新したい場合、新規のシークレットで新しいアプリケーションを作成できます。また、そのアプリのクライアント ID を使用するアプリケーションからのアクセスを遮断するためにアプリケーションを削除できます。

パブリックアプリでは、機密クライアントとクライアントシークレットを使用できます。Amazon CloudFront プロキシを使用して、転送`SECRET_HASH`中の を追加します。詳細については、ブログの「Amazon プロキシを使用して Amazon Cognito のパブリッククライアントを保護する CloudFront AWS 」を参照してください。

JSON ウェブトークン

Amazon Cognito アプリクライアントは、以下のタイプの JSON ウェブトークン (JWT) を発行できます。

アイデンティティ (ID) トークン

ユーザーがユーザープールから認証されていることを示す検証可能なステートメント。OpenID Connect (OIDC) は、OAuth 2.0 で定義されているアクセストークンとリフレッシュトークンの標準に ID トークンの仕様を追加しました。ID トークンには、アプリがユーザープロファイルの作成やリソースのプロビジョニングに使用できるユーザー属性などの ID 情報が含まれています。詳細については、「ID トークンの使用」を参照してください。

アクセストークン

ユーザーのアクセス権を示す検証可能なステートメント。アクセストークンには、OIDC と OAuth 2.0 の機能であるスコープが含まれています。アプリは、バックエンドリソースにスコープを提示することで、ユーザープールが API のデータや独自のユーザーデータへのアクセスをユーザーまたはマシンに許可していることを証明できます。カスタムスコープを持つアクセストークンは、多くの場合、M2M クライアント認証情報の付与により、リソースサーバーへのアクセスを許可します。詳細については、「アクセストークンの使用」を参照してください。

更新トークン

ユーザーのトークンが有効期限切れになったときにアプリがユーザープールに提示できる、暗号化された初期認証ステートメント。更新トークンをリクエストすると、有効期限が切れていない新しいアクセストークンと ID トークンが返されます。詳細については、「更新トークンの使用」を参照してください。

各アプリクライアントのこれらのトークンの有効期限は、Amazon Cognito コンソールでユーザープールの [アプリケーションの統合] タブから設定できます。

アプリクライアントの用語

Amazon Cognito コンソールで、アプリクライアントに使用できる用語は次のとおりです。

許可されているコールバック URL

コールバック URL は、ユーザーがサインインに成功したときにリダイレクトされる先を指定します。少なくとも 1 つのコールバック URL を選択してください。コールバック URL は、以下が必要です。

- 絶対 URI である。
- クライアントに事前登録されている。
- フラグメントコンポーネントを含まない。

「OAuth 2.0 - リダイレクトエンドポイント」を参照してください。

Amazon Cognito は、テスト目的限定の HTTPS を除き、HTTP ではなく `http://localhost` を使用します。

また、アプリのコールバック URL (例: `myapp://example`) もサポートされています。

許可されているサインアウト URL

サインアウト URL は、ユーザーがサインアウトしたときにリダイレクトされる先を指定します。

属性の読み取りおよび書き込み許可

ユーザープールには多数の顧客があり、それぞれに独自のアプリクライアントと があります IdPs。アプリに関連したユーザー属性のみの読み取りおよび書き込みアクセスを許可するようにアプリクライアントに設定できます。 machine-to-machine (M2M) 認証などの場合は、どのユーザー属性にもアクセスを許可できません。

属性の読み取りおよび書き込みアクセス権限の設定に関する考慮事項

- アプリケーションクライアントを作成し、属性の読み取りおよび書き込みアクセス許可をカスタマイズしない場合、Amazon Cognito はすべてのユーザープール属性に読み取りおよび書き込みアクセス許可を付与します。
- イミュータブルなカスタム属性への書き込みアクセス権を付与できます。ユーザーの作成時またはサインアップ時に、アプリケーションクライアントはイミュータブルな属性に値を書き込むことができます。その後は、ユーザーのイミュータブルなカスタム属性に値を書き込むことはできません。
- アプリクライアントは、ユーザープール内の必須属性への書き込み権限を持っている必要があります。Amazon Cognito コンソールは、必須の属性を自動的に書き込み可能に設定します。
- アプリケーションクライアントに対して、`email_verified` または `phone_number_verified` への書き込みアクセス権限を付与することはできません。ユーザープール管理者は、これらの値を変更できます。ユーザーは、これらの属性の値を属性の検証でのみ変更できます。

認証フロー

アプリクライアントが許可するサインイン方法。アプリは、ユーザー名とパスワードによる認証、セキュアリモートパスワード (SRP)、Lambda トリガーによるカスタム認証、およびトークンの更新をサポートできます。セキュリティのベストプラクティスとして、SRP 認証を主なサインイン方法として使用します。ホストされた UI は、SRP を使用してユーザーの自動サインインを行います。

カスタムスコープ

カスタムスコープは、[リソースサーバー] で各自のリソースサーバー用に定義したものです。形式は *`resource-server-identifier`*/*`scope`* です。スコープ、M2M 、およびリソースサーバーによる API 認証 を参照してください。

デフォルトのリダイレクト URI

サードパーティーの を使用するユーザーの認証リクエストの `redirect_uri`パラメータを置き換えます IdPs。CreateUserPoolClient または UpdateUserPoolClient API リクエストの `DefaultRedirectURI`パラメータを使用して、このアプリケーションクライアント設定を構成します。この URL は、アプリクライアントの `CallbackURLs` のメンバーである必要があります。Amazon Cognito は、次の場合に認証されたセッションをこの URL にリダイレクトします。

1. アプリクライアントには、1 つの ID プロバイダーが割り当てられ、複数のコールバック URLsが定義されています。ユーザープールは、`redirect_uri`パラメータが含まれていない場合、認証リクエストをデフォルトのリダイレクト URI にリダイレクトします。
2. アプリクライアントには、1 つの ID プロバイダーが割り当てられ、1 つのコールバック URLsが定義されています。このシナリオでは、デフォルトのコールバック URL を定義する必要はありません。`redirect_uri` パラメータを含まないリクエストは、使用可能な 1 つのコールバック URL にリダイレクトされます。

ID プロバイダ

ユーザープールの外部 ID プロバイダー (IdPs) の一部またはすべてを選択して、ユーザーを認証できます。アプリクライアントは、ユーザープール内のローカルユーザーのみを認証することもできます。IdP をアプリクライアントに追加すると、IdP への認証リンクを生成して、ホストされた UI サインインページで表示できます。複数の を割り当てることはできますが IdPs、少なくとも 1 つを割り当てる必要があります。外部 の使用の詳細については、 IdPs「」を参照してくださいサードパーティー経由のユーザープールへのサインインの追加。

OpenID Connect のスコープ

次の `OAuth` スコープを 1 つ以上選択し、アクセストークン用にリクエストできるアクセス権限を指定します。

- `openid` スコープでは、ID トークンとユーザーの固有 ID を取得することを宣言します。また、リクエストに含まれる追加のスコープに応じて、すべてまたは一部のユーザー属性もリクエストします。`openid` スコープをリクエストしない限り、Amazon Cognito は ID トークンを返しません。`openid` スコープは、有効期限やキー ID などの構造的な ID トークンのクレームを承認し、UserInfo エンドポイント からのレスポンスで受け取るユーザー属性を決定します。
  - リクエストするスコープが `openid` のみである場合、Amazon Cognito は現在のアプリケーションクライアントが読み取ることができるすべてのユーザー属性を ID トークンに入力します。このスコープだけを使用したアクセストークンへの `userInfo` レスポンスにより、すべてのユーザー属性が返されます。

- phone、email、profile などの他のスコープで openid をリクエストすると、ID トークンと userInfo はユーザー固有の ID と追加のスコープで定義された属性を返します。

- phone スコープは phone_number クレームおよび phone_number_verified クレームへのアクセスを付与します。このスコープは openid スコープを使用する場合にのみリクエストできます。
- email スコープは email クレームおよび email_verified クレームへのアクセスを付与します。このスコープは openid スコープを使用する場合にのみリクエストできます。
- aws.cognito.signin.user.admin スコープは、 UpdateUserAttributesや などのアクセストークンを必要とする Amazon Cognito ユーザープール API オペレーションへのアクセスを許可しますVerifyUserAttribute。
- profile スコープはクライアントが読み取り可能なすべてのユーザー属性へのアクセスを付与します。このスコープは openid スコープを使用する場合にのみリクエストできます。

スコープの詳細については、標準 OIDC スコープのリストを参照してください。

OAuth 付与タイプ

OAuth 付与は、ユーザープールのトークンを取得する認証方法です。Amazon Cognito は、以下の付与タイプをサポートしています。これらの OAuth 付与をアプリに統合するには、ユーザープールにドメインを追加する必要があります。

認証コード付与

認証コード付与は、アプリがトークンエンドポイントを使用してユーザープールのトークンと交換できるコードを生成します。認証コードを交換すると、アプリは ID、アクセス、更新の各トークンを受け取ります。この OAuth フローは、暗黙的な付与と同様に、ユーザーのブラウザで発生します。認証コード付与は、トークンがユーザーのセッションで表示されないため、Amazon Cognito が提供する最も安全な付与です。代わりに、アプリはトークンを返すリクエストを生成し、保護されたストレージにトークンをキャッシュできます。詳細については、IETF RFC 6749 #1.3.1 の [認証コード] を参照してください。

> **Note**
>
> パブリッククライアントアプリにおけるセキュリティのベストプラクティスとして、認証コード付与の OAuth フローのみを有効にし、Proof Key for Code Exchange (PKCE) を実装してトークン交換を制限します。PKCE では、クライアントは、元の認証リクエストで提示したのと同じシークレットをトークンエンドポイントに提供した場合にのみ、認

証コードを交換できます。PKCE の詳細については、IETF RFC 7636 を参照してください。

暗黙的な付与

暗黙的な付与は、認可エンドポイント からアクセストークンと ID トークンをユーザのブラウザセッションに直接配信します (更新トークンは配信しません)。暗黙的な付与では、トークンエンドポイントへの個別のリクエストは不要になりますが、PKCE とは互換性がなく、更新トークンは返されません。この付与は、認証コード付与を完了できないテストシナリオやアプリアーキテクチャに対応できます。詳細については、IETF RFC 6749 #1.3.2 の「暗黙的な付与」を参照してください。認証コード付与と暗黙的な付与の両方をアプリクライアントで有効にして、必要に応じて使い分けることができます。

クライアント認証情報の付与

クライアント認証情報の付与は machine-to-machine (M2M) 通信用です。認証コード付与と暗黙的な付与は、認証された人間のユーザーにトークンを発行します。クライアント認証情報の付与は、非対話型システムから API にスコープベースの認証を付与します。アプリは、トークンエンドポイントに対してクライアント認証情報を直接リクエストし、アクセストークンを受け取ることができます。詳細については、IETF RFC 6749 #1.3.4 の「クライアント認証情報」を参照してください。クライアント認証情報の付与は、クライアントシークレットを持ち、認証コード付与や暗黙的な付与をサポートしていないアプリクライアントでのみ有効化できます。

> **Note**
>
> ユーザーとしてクライアント認証情報フローを呼び出すことはないため、このフローでアクセストークンに追加できるのはカスタムスコープのみです。カスタムスコープは、各自のリソースサーバー用に定義したものです。`openid` や `profile` などのデフォルトのスコープは、人間以外のユーザーには適用されません。
> ID トークンはユーザー属性の検証であるため、M2M 通信には関係なく、クライアント認証情報の付与では ID トークンを発行しません。スコープ、M2M 、およびリソースサーバーによる API 認証 を参照してください。

クライアント認証情報の付与は、 AWS 請求書にコストを追加します。詳細については、「Amazon Cognito の料金」を参照してください。

アプリクライアントの作成

AWS Management Console

アプリクライアントを作成する (コンソール)

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択、またはユーザープールを作成します。
4. [App integration] (アプリケーションの統合) タブを選択します。
5. [App clients] (アプリケーションクライアント) で、[Create an app client] (アプリケーションクライアントの作成) を選択します。
6. [App type] (アプリケーションタイプ) を選択します。[Public client] (パブリッククライアント)、[Confidential client] (機密保持クライアント)、または [Other] (その他)。
7. [App client name] (アプリケーションクライアント名) を入力します。
8. [クライアントシークレットを生成] を選択して、Amazon Cognito でクライアントシークレットを自動生成します。クライアントシークレットは通常、機密クライアントに関連付けられます。
9. アプリケーションクライアントで許可したい [Authentication flows] (認証のフロー) を選択します。
10. 認証フローセッション持続期間を設定します。これは、セッショントークンが期限切れになる前にユーザーが各認証チャレンジを完了しなければならない期間です。
11. (オプション) トークンの有効期限を設定する場合は、以下の手順を実行します。
    a. アプリケーションクライアントの [Refresh token expiration] (トークンの有効期限を更新) を指定します。デフォルト値は 30 日です。これは、1 時間から 10 年の任意の値に変更できます。
    b. アプリケーションクライアントの [Access token expiration] (アクセストークンの有効期限) を指定します。デフォルト値は 1 時間です。これは、5 分から 24 時間の任意の値に変更できます。
    c. アプリケーションクライアントの [ID token expiration] (ID トークンの有効期限) を指定します。デフォルト値は 1 時間です。これは、5 分から 24 時間の任意の値に変更できます。

> **Important**
>
> ホストされた UI を使用してトークンの有効期間を 1 時間未満に設定した場合、ユーザーは、現在 1 時間に固定されているセッション cookie の期間に基づいてトークンを使用できるようになります。

12. このアプリケーションクライアントに、[Enable token revocation] (トークンの失効を有効化) するかどうかを選択してください。これにより、Amazon Cognito が発行するトークンサイズが増加します。
13. このアプリクライアントに対して、[ユーザー存在エラーを防ぐ] かどうかを選択します。Amazon Cognito は、ユーザー名またはパスワードのいずれかが間違っているという一般的なメッセージで、存在しないユーザーのサインインリクエストに応答します。
14. ホストされた UI をこのアプリクライアントで使用する場合は、[ホストされた UI 設定] を設定します。

    a. 許可されているコールバック URL を 1 つ以上入力します。これらは、認証が完了したユーザーを Amazon Cognito でリダイレクトする先のウェブ URL またはアプリ URL です。

    b. 許可されているサインアウト URL を 1 つ以上入力します。これらは、ログアウトエンドポイント へのリクエストにおいてアプリで受け入れる URL です。

    c. アプリにユーザーをサインインできるようにする ID プロバイダーを 1 つ以上選択します。既存の の任意の組み合わせを選択できます IdPs。ユーザープールだけで、またはユーザープールで IdPs 設定した 1 つ以上のサードパーティーでユーザーを認証できます。

    d. アプリクライアントで受け入れる OAuth 2.0 付与タイプを選択します。

    - トークンエンドポイント でトークンと交換できるコードをアプリに渡すには、[認証コード付与] を選択します。
    - ID トークンとアクセストークンをアプリに直接渡すには、[暗黙的な付与] を選択します。暗黙的な付与フローは、トークンをユーザーに直接公開します。
    - ユーザー認証情報ではなく、クライアントシークレットの知識に基づいてアプリにアクセストークンを渡すには、[クライアント認証情報] を選択します。クライアント認証情報の付与フローは、認証コード付与フローや暗黙的な付与フローと相互に排他的です。

e. このアプリクライアントでの使用を認可する [OpenID Connect スコープ] を選択します。アクセストークンは、ユーザープール API を使用して `aws.cognito.signin.user.admin` スコープでのみ生成できます。その他のスコープについては、トークンエンドポイント にアクセストークンをリクエストする必要があります。

f. アプリクライアントで認可するカスタムスコープを選択します。カスタムスコープは、サードパーティ API へのアクセスを認可するために最もよく使用されます。

15. このアプリクライアントに、属性の読み取りおよび書き込み許可を設定します。アプリクライアントは、ユーザープールの属性スキーマのすべて、または限定されたサブセットの読み取りと書き込みのアクセス許可を持つことができます。

16. [Create app client] を選択します。

17. [Client id] (クライアント ID) を書き留めます。これにより、サインアップリクエストとサインインリクエストでアプリケーションクライアントを識別します。

AWS CLI

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-
name myApp
```

> **Note**
>
> コールバック URL とログアウト URL は、JSON 形式を使用することで、CLI でリモートパラメータファイルとして処理されないようにします。
>
> ```
> --callback-urls "["https://example.com"]"
> --logout-urls "["https://example.com"]"
> ```

詳細については、 AWS CLI コマンドリファレンスを参照してください。 create-user-pool-client

Amazon Cognito user pools API

CreateUserPoolClient API リクエストを生成します。デフォルト値に設定しないすべてのパラメータには、値を指定する必要があります。

ユーザープールアプリケーションクライアント (AWS CLI および AWS API) の更新

で AWS CLI、次のコマンドを入力します。

```
aws cognito-idp update-user-pool-client --user-pool-id  "MyUserPoolID" --client-id
 "MyAppClientID" --allowed-o-auth-flows-user-pool-client --allowed-o-auth-flows "code"
 "implicit" --allowed-o-auth-scopes "openid" --callback-urls "["https://example.com"]"
 --supported-identity-providers "["MySAMLIdP", "LoginWithAmazon"]"
```

コマンドが成功すると、 は確認 AWS CLI を返します。

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

詳細については、 AWS CLI コマンドリファレンスを参照してください。 update-user-pool-client

AWS API: UpdateUserPoolClient

ユーザープールアプリクライアント (AWS CLI および AWS API) に関する情報の取得

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-
id MyClientID
```

詳細については、 AWS CLI コマンドリファレンスを参照してください。 describe-user-pool-client

AWS API: DescribeUserPoolClient

ユーザープール (AWS CLI および AWS API) 内のすべてのアプリケーションクライアント情報を一覧表示する

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

詳細については、 AWS CLI コマンドリファレンスを参照してください。 list-user-pool-clients

AWS API: ListUserPoolClients

ユーザープールアプリケーションクライアント (AWS CLI および AWS API) の削除

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id
 "MyAppClientID"
```

詳細については、 AWS CLI コマンドリファレンスを参照してください。 delete-user-pool-client

AWS API: DeleteUserPoolClient

ユーザープール内のユーザーデバイスの使用

Amazon Cognito ユーザープール API を使用してローカルユーザープールのユーザーをサインインすると、高度なセキュリティ機能からのユーザーのアクティビティログを各デバイスに関連付けることができます。また、信頼できるデバイスを使用している場合は、オプションでユーザーが多要素認証 (MFA) をスキップするのを可能にします。Amazon Cognito には、デバイス情報がまだ含まれていないサインインへのレスポンスにデバイスキーが含まれます。デバイスキーの形式は *Region_UUID* です。デバイスキー、Secure Remote Password (SRP) ライブラリ、およびデバイス認証を許可するユーザープールがあれば、アプリ内のユーザーに現在のデバイスを信頼するように求めることができ、サインイン時に MFA コードの入力を求める必要がなくなります。

トピック

- 記憶済みデバイスのセットアップ

- デバイスキーの取得
- デバイスでのサインイン
- デバイスの表示、更新、削除

記憶済みデバイスのセットアップ

Amazon Cognito ユーザープールでは、ユーザーの各デバイスを固有のデバイス識別子、つまりデバイスキーに関連付けることができます。サインイン時にデバイスキーを提示してデバイス認証を行うと、2 つの機能を利用できます。

1. 高度なセキュリティ機能により、セキュリティと分析を目的として、特定のデバイスでのユーザーアクティビティをモニタリングできます。ユーザーがログインすると、アプリでは、オプションで各ユーザーとそのデバイスを認証して、アクティビティログにデバイス情報を追加することができます。
2. デバイスを記憶することで、信頼できるデバイスの認証フローもサポートされます。これにより、ユーザーは、アプリのセキュリティ要件に適した期間、MFA なしでログインすることを選択できます。ユーザーに MFA コードの送信を再度求める場合は、ユーザーのデバイスの記憶状態を変更できます。

   記憶されているデバイスは、MFA がアクティブなユーザープールでのみ MFA をオーバーライドできます。

ユーザーが記憶されているデバイスでログインする場合、認証フロー中に追加のデバイス認証を実行する必要があります。詳細については、「デバイスでのサインイン」を参照してください。

ユーザープールの [サインインエクスペリエンス] タブの [デバイスの追跡] で、ユーザープールがデバイスを記憶するように設定します。Amazon Cognito コンソールを使用して記憶済みデバイス機能をセットアップするときは、[常に]、[User Opt-In] (ユーザーオプトイン)、および [いいえ] の 3 つのオプションがあります。

記憶しない

ユーザープールでは、ログイン時にデバイスを記憶するように求めるメッセージは表示されません。

常に記憶する

アプリがユーザーのデバイスを確認すると、ユーザープールは常にデバイスを記憶し、今後デバイスへのログインが成功しても MFA チャレンジを返しません。

ユーザーオプトイン

アプリがユーザーのデバイスを確認しても、ユーザープールは MFA チャレンジを自動的に抑制しません。デバイスを記憶するかどうかをユーザーが選択するようにユーザーに求める必要があります。

[常に記憶する] または [ユーザーオプトイン] を選択すると、ユーザーが未確認のデバイスからサインインするたびに、Amazon Cognito はデバイス ID キーとシークレットを生成します。デバイスキーは、ユーザーがデバイス認証を実行したときにアプリがユーザープールに送信する最初の識別子です。

確認済みの各ユーザーデバイスでは、自動的に記憶されるかオプトインされたかに関係なく、ユーザーがサインインするたびにデバイス識別子キーとシークレットを使用してデバイスを認証できます。

CreateUserPool または UpdateUserPool API リクエストで、ユーザープールの記憶デバイス設定を構成することもできます。詳細については、DeviceConfiguration のプロパティを参照してください。

Amazon Cognito ユーザープール API には記憶されているデバイスに対してさらに 2 つのオペレーションがあります。

1. ListDevices と AdminListDevices は、ユーザーのデバイスキーとそのメタデータのリストを返します。
2. GetDevice と AdminGetDevice は、単一のデバイスのデバイスキーとメタデータを返します。
3. UpdateDeviceStatus と AdminUpdateDeviceStatus は、ユーザーのデバイスを記憶済みまたは未記憶として設定します。
4. ForgetDevice と AdminForgetDevice は、ユーザーの確認済みデバイスをプロファイルから削除します。

名前が `Admin` で始まる API オペレーションはサーバー側アプリで使用するもので、IAM 認証情報を使用して認証する必要があります。詳細については、「Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用」を参照してください。

デバイスキーの取得

ユーザーがユーザープール API を使用してサインインし、認証パラメータに `DEVICE_KEY` としてデバイスキーを含めない場合、Amazon Cognito はレスポンスに新しいデバイスキーを返します。クラ

イアント側パブリックアプリでは、デバイスキーをアプリストレージに配置することで、今後のリクエストに含められるようになります。機密のサーバー側アプリで、ブラウザー Cookie または別のクライアント側トークンをユーザーのデバイスキーで設定します。

ユーザーが信頼できるデバイスを使用してサインインする前に、アプリはデバイスキーを確認して追加情報を提供する必要があります。デバイスキー、わかりやすい名前、パスワード検証ツール、およびソルトを使用してユーザーのデバイスを確認する ConfirmDevice リクエストを Amazon Cognito に生成します。ユーザープールでオプトインデバイス認証を設定した場合、Amazon Cognito は `ConfirmDevice` リクエストに、ユーザーは現在のデバイスを記憶するかどうかを選択する必要があるプロンプトで応答する必要があります。UpdateDeviceStatus リクエストでユーザーが選択した内容を返信します。

ユーザーのデバイスを確認したが、記憶されるように設定しなかった場合、Amazon Cognito は関連付けを保存しますが、デバイスキーを入力するとデバイス以外のサインインで続行します。デバイスは、ユーザーのセキュリティとトラブルシューティングに役立つログを生成できます。確認済みで記憶されていないデバイスは、サインイン機能を利用しませんが、セキュリティモニタリングログ機能は利用します。アプリクライアントの高度なセキュリティ機能を有効にし、デバイスのフットプリントをリクエストにエンコードすると、Amazon Cognito はユーザーイベントを確認済みのデバイスに関連付けます。

新しいデバイスキーを取得するには

1. InitiateAuth API リクエストを使用してユーザーのサインインセッションを開始します。
2. ユーザーのサインインセッションが完了したことをマークする JSON Web トークン (JWT) を受け取るまで、すべての認証チャレンジに RespondToAuthChallenge で応答します。
3. アプリで、Amazon Cognito がその `RespondToAuthChallenge` または `InitiateAuth` レスポンスの `NewDeviceMetadata` に返す値 (`DeviceGroupKey` および `DeviceKey`) を記録します。
4. ユーザー用の新しい SRP シークレット、つまりソルトおよびパスワード検証ツールを生成します。この関数は SRP ライブラリを提供する SDK で使用できます。
5. ユーザーにデバイス名の入力を求めるか、ユーザーのデバイス特性からデバイス名を生成します。
6. ConfirmDevice API リクエストでユーザーのアクセストークン、デバイスキー、デバイス名、SRP シークレットを指定します。ユーザープールがデバイスを [常に記憶する] に設定されていれば、ユーザー登録は完了です。
7. Amazon Cognito が `"UserConfirmationNecessary": true` で `ConfirmDevice` に応答した場合は、デバイスを記憶するかどうかを選択するようユーザーに促します。ユーザーがデバイスを記憶すると断言したら、ユーザーのアクセストークン、デバイスキー、および

"DeviceRememberedStatus": "remembered" を使用して UpdateDeviceStatus API リクエストを生成します。

8. Amazon Cognito にデバイスを記憶するように指示した場合、次回のサインイン時に MFA チャレンジの代わりに DEVICE_SRP_AUTH チャレンジが表示されます。

デバイスでのサインイン

ユーザーのデバイスを記憶するように設定すると、Amazon Cognito ではユーザーが同じデバイスキーでサインインするときに MFA コードを送信する必要がなくなります。デバイス認証は、MFA 認証チャレンジをデバイス認証チャレンジに置き換えるだけです。グループとしてサインインすることはできません。ユーザーはまずパスワードまたはカスタムチャレンジを使用して認証を完了する必要があります。記憶しているデバイスでのユーザーの認証プロセスは次のとおりです。

カスタム認証チャレンジ Lambda トリガーを使用するフローでデバイス認証を実行するには、InitiateAuth API リクエストで DEVICE_KEY パラメータを渡します。ユーザーがすべてのチャレンジに成功し、CUSTOM_CHALLENGE チャレンジが true の issueTokens 値を返すと、Amazon Cognito は最後の 1 つの DEVICE_SRP_AUTH チャレンジを返します。

デバイスでのサインインするには

1. ユーザーのデバイスキーをクライアントストレージから取得します。
2. InitiateAuth API リクエストを使用してユーザーのサインインセッションを開始します。USER_SRP_AUTH、REFRESH_TOKEN_AUTH、USER_PASSWORD_AUTH、または CUSTOM_AUTH の AuthFlow を選択します。AuthParameters で、ユーザーのデバイスキーを DEVICE_KEY パラメータに追加し、選択したログインフローに必要なその他のパラメータを含めます。
   a. 認証チャレンジへの PASSWORD_VERIFIER レスポンスのパラメータで DEVICE_KEY を渡すこともできます。
3. レスポンスに DEVICE_SRP_AUTH チャレンジを受け取るまで、チャレンジレスポンスを完了します。
4. RespondToAuthChallenge API リクエストで、DEVICE_SRP_AUTH の ChallengeName と USERNAME、DEVICE_KEY、および SRP_A のパラメータを送信します。
5. Amazon Cognito は DEVICE_PASSWORD_VERIFIER チャレンジで応答します。このチャレンジレスポンスには、SECRET_BLOCK および SRP_B の値が含まれます。
6. SRP ライブラリを使用して PASSWORD_CLAIM_SIGNATURE、PASSWORD_CLAIM_SECRET_BLOCK、TIMESTAMP、USERNAME、DEVI

パラメータを生成して送信します。これらは追加の RespondToAuthChallenge リクエストで送信してください。

7. ユーザーの JWT を受け取るまで、追加のチャレンジを完了してください。

次の擬似コードは、DEVICE_PASSWORD_VERIFIER チャレンジレスポンスの値を計算する方法を示しています。

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = Tue Sep 25 00:09:40 UTC 2018
PASSWORD_CLAIM_SIGNATURE = Base64(SHA256_HMAC(K_USER, DeviceGroupKey + DeviceKey +
 PASSWORD_CLAIM_SECRET_BLOCK + TIMESTAMP))
K_USER = SHA256_HASH(S_USER)
S_USER = (SRP_B - k * gx)(a + ux)
x = SHA256_HASH(salt + FULL_PASSWORD)
u = SHA256_HASH(SRP_A + SRP_B)
k = SHA256_HASH(N + g)
```

デバイスの表示、更新、削除

Amazon Cognito API を使用して、アプリに次の機能を実装できます。

1. ユーザーの現在のデバイスに関する情報を表示します。
2. ユーザーのすべてのデバイスのリストを表示します。
3. デバイスの「記憶済み」状態を解除する
4. デバイスの記憶状態を更新します。

以下の説明の API リクエストを承認するアクセストークンには、aws.cognito.signin.user.admin スコープを含める必要があります。Amazon Cognito は、Amazon Cognito ユーザープール API を使用して生成するすべてのアクセストークンに、このスコープのクレームを追加します。サードパーティ IdP は、Amazon Cognito を認証するユーザーのデバイスと MFA を個別に管理する必要があります。ホストされた UI では aws.cognito.signin.user.admin スコープをリクエストできますが、ホストされた UI はデバイス情報を高度なセキュリティのユーザーログに自動的に追加するので、デバイスの記憶は提供しません。

デバイスに関する情報を表示する

ユーザーのデバイスに関する情報をクエリして、そのデバイスが現在も使用されているかどうかを判断できます。例えば、記憶されているデバイスが 90 日間サインインしていない場合に、そのデバイスを非アクティブ化することができます。

- パブリッククライアントアプリにユーザーのデバイス情報を表示するには、ユーザーのアクセスキーとデバイスキーを GetDevice API リクエストで送信します。
- 機密クライアントアプリにユーザーのデバイス情報を表示するには、AWS 認証情報を使用して AdminGetDevice API リクエストに署名し、ユーザーのユーザー名、デバイスキー、およびユーザープールを送信します。

ユーザーのすべてのデバイスのリストを表示する

ユーザーのすべてのデバイスとそのプロパティのリストを表示できます。例えば、現在のデバイスが記憶されているデバイスと一致することを検証することが可能です。

- パブリッククライアントアプリでは、ListDevices API リクエストでユーザーのアクセストークンを送信します。
- 機密クライアントアプリでは、AWS 認証情報を使用して AdminListDevices API リクエストに署名し、ユーザーのユーザー名とユーザープールを送信します。

デバイスの「記憶済み」状態を解除する

ユーザーのデバイスキーは削除できます。これは、ユーザーがデバイスを使用しなくなったと判断した場合や、異常なアクティビティを検出してユーザーに MFA を再度完了するように促す場合に役立ちます。デバイスを後で再登録するには、新しいデバイスキーを生成して保存する必要があります。

- パブリッククライアントアプリでは、ForgetDevice API リクエストでユーザーのデバイスキーとアクセストークンを送信します。
- 機密クライアントアプリでは、AdminForgetDevice API リクエストでユーザーのデバイスキーとアクセストークンを送信します。

Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用

サインアップ、サインイン、ユーザープール内のユーザー管理を行うには、2 つのオプションがあります。

1. ユーザープールのエンドポイントには、ホストされた UI とフェデレーションエンドポイントが含まれます。これらは、ユーザープールのドメインを選択すると Amazon Cognito がアクティブ化するパブリックウェブページのパッケージを構成します。サインアップ、サインイン、パスワード管理、多要素認証 (MFA) のページを含む Amazon Cognito ユーザープールの認証および承認機能をすぐに使い始めるには、ホストされている UI の組み込みユーザーインターフェイスを使用してください。その他のユーザープールエンドポイントを使うと、サードパーティーの ID プロバイダー (IdP) を使った認証が容易になります。行われるサービスには以下が含まれます。

   a. IdPs からの認証済みクレーム用のサービスプロバイダーのコールバックエンドポイント (`saml2/idpresponse` や `oauth2/idpresponse` など)。Amazon Cognito がアプリと IdP の間の中間サービスプロバイダー (SP) である場合、コールバックエンドポイントはサービスを表します。

   b. 環境に関する情報を提供するエンドポイント (`oauth2/userInfo` や `jwks.json` など)。アプリは、AWS SDK や OAuth 2.0 ライブラリを使用してトークンを検証したり、ユーザープロファイルデータを取得したりするときに、これらのエンドポイントを使用します。

2. Amazon Cognito ユーザープール API は、ウェブまたはモバイルアプリが独自のカスタムフロントエンドでサインイン情報を収集した後、ユーザーを認証する一連のツールです。ユーザープール API 認証では、次の JSON ウェブトークンが生成されます。

   a. ユーザーからの検証可能な属性クレームを含む ID トークン。

   b. ユーザーが AWS サービスエンドポイントにトークン認証された API リクエストを作成するのを認証するアクセストークン。

   > **Note**
   >
   > デフォルトでは、ユーザープール API 認証のアクセストークンには `aws.cognito.signin.user.admin` スコープのみが含まれます。サードパーティー API へのリクエストを許可するなど、スコープを追加したアクセストークンを生成するには、ユーザープールエンドポイントで認証中にスコープをリクエストするか、トークン生成前の Lambda トリガー でカスタムスコープを追加します。アクセストークンをカスタマイズすると、AWS 請求額にコストが追加されます。

通常はユーザープールのエンドポイントを使用してサインインするフェデレーションユーザーを、プロファイルがユーザープールに対してローカルであるユーザーとリンクできます。ローカルユーザーは、外部 IdP を介したフェデレーションなしに、ユーザープールディレクトリにのみ存在します。AdminLinkProviderForUser API リクエストでフェデレーティッド ID をローカルユーザーにリン

クすると、そのユーザーはユーザープール API でサインインできます。詳細については、「フェデレーションユーザーを既存のユーザープロファイルにリンクする」を参照してください。

Amazon Cognito ユーザープール API には 2 つの用途があります。Amazon Cognito ユーザープールリソースを作成して設定します。例えば、ユーザープールの作成、AWS Lambdaトリガーの追加、ホストされている UI ドメインの設定を行うことができます。ユーザープール API は、ローカルとリンクされたユーザーのサインアップ、サインイン、その他のユーザーオペレーションも実行します。

Amazon Cognito ユーザープール API を使用したシナリオ例

1. ユーザーは、アプリで作成した [Create an account] (アカウントを作成) ボタンを選択します。E メールアドレスとパスワードを入力します。
2. アプリケーションは SignUp API リクエストを送信し、ユーザープールに新しいユーザーを作成します。
3. アプリケーションは、ユーザーに E メールの確認コードを求めます。ユーザーは、E メールメッセージで受け取ったコードを入力します。
4. アプリは、ユーザーの確認コードを含む ConfirmSignUp API リクエストを送信します。
5. アプリケーションは、ユーザーにユーザー名とパスワードの入力を求め、情報を入力します。
6. アプリは InitiateAuth API リクエストを送信し、ID トークン、アクセストークン、更新トークンを保存します。アプリは OIDC ライブラリを呼び出して、ユーザーのトークンを管理し、そのユーザーの永続セッションを維持します。

Amazon Cognito ユーザープール API では、IdP を介してフェデレートするユーザーをサインインすることはできません。これらのユーザーは、ユーザープールのエンドポイントを使用して認証する必要があります。ホストされた UI を含むユーザープールエンドポイントの詳細については、「ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス」を参照してください。フェデレーションユーザーは、ホストされた UI で起動して IdP を選択することも、ホスト UI をスキップしてユーザーを IdP に直接送信してサインインさせることもできます。認可エンドポイント への API リクエストに IdP パラメータが含まれている場合、Amazon Cognito はユーザーを IdP サインインページにそのままリダイレクトします。

ユーザープールエンドポイントのサンプルシナリオ

1. ユーザーは、アプリで作成した [Create an account] (アカウントを作成) ボタンを選択します。
2. 開発者の認証情報を登録したソーシャル ID プロバイダーのリストをユーザーに提示します。ユーザーは Apple を選択します。

3. アプリは、プロバイダー名 `SignInWithApple` で 認可エンドポイント へのリクエストを開始します。
4. ユーザーのブラウザが Apple OAuth 認証ページを開きます。ユーザーは、Amazon Cognito がプロファイル情報を読み取ることを許可することを選択します。
5. Amazon Cognito は Apple アクセストークンを確認し、ユーザーの Apple プロファイルを照会します。
6. ユーザーは Amazon Cognito 認証コードをアプリケーションに提示します。
7. アプリケーションは認証コードを トークンエンドポイント と交換し、ID トークン、アクセストークン、更新トークンを保存します。アプリは OIDC ライブラリを呼び出して、ユーザーのトークンを管理し、そのユーザーの永続セッションを維持します。

ユーザープール API とユーザープールエンドポイントは、このガイドで説明されているように、さまざまなシナリオをサポートしています。次のセクションでは、ユーザープール API がサインアップ、サインイン、およびリソース管理の要件をサポートするクラスにさらにどのように分類されるかを見ていきます。

Amazon Cognito ユーザープールの認証済みおよび未認証の API オペレーション

Amazon Cognito ユーザープール API は、リソース管理インターフェイスとユーザー向け認証および承認インターフェイスの両方であり、運用に続く承認モデルを組み合わせます。API オペレーションによっては、IAM 認証情報、アクセストークン、セッショントークン、クライアントシークレット、またはこれらの組み合わせを使用して認証を行う必要がある場合があります。多くのユーザー認証および承認操作では、リクエストの認証済みバージョンと認証されていないバージョンを選択できます。認証されていない操作は、モバイルアプリなど、ユーザーに配布するアプリのセキュリティ上のベストプラクティスです。コードにシークレットを含める必要はありません。

IAM ポリシーでは、IAM 認証による管理オペレーション と IAM 認証済みユーザーオペレーション にのみアクセス許可を割り当てることができます。

IAM 認証による管理オペレーション

IAM 認証管理オペレーションでは、AWS Management Console で行う場合と同様に、ユーザープールとアプリケーションクライアントの設定を変更および表示できます。

例えば、UpdateUserPool API リクエストでユーザープールを変更するには、リソースを更新するための AWS 認証情報と IAM アクセス許可を提示する必要があります。

AWS Command Line Interface (AWS CLI) または AWS SDK でこれらのリクエストを承認するには、環境変数またはリクエストに IAM 認証情報を追加するクライアント設定を使用して環境を設定します。詳細については、「AWS 全般のリファレンス」の「AWS 認証情報を使用して AWS にアクセスする」を参照してください。Amazon Cognito ユーザープール API のサービスエンドポイントにリクエストを直接送信することもできます。リクエストのヘッダーに埋め込んだ AWS 認証情報を使用して、これらのリクエストを承認または署名する必要があります。詳細については、「AWS API リクエストの署名」を参照してください。

IAM 認証による管理オペレーション

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

DeleteGroup

DeleteIdentityProvider

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

IAM 認証による管理オペレーション

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

IAM 認証による管理オペレーション

`SetUICustomization`

`SetUserPoolMfaConfig`

`StartUserImportJob`

`StopUserImportJob`

`TagResource`

`UntagResource`

`UpdateGroup`

`UpdateIdentityProvider`

`UpdateResourceServer`

`UpdateUserPool`

`UpdateUserPoolClient`

`UpdateUserPoolDomain`

IAM 認証済みユーザーオペレーション

IAM 認証済みユーザーオペレーションは、ユーザーのサインアップ、サインイン、認証情報の管理、変更、および表示を行います。

例えば、ウェブフロントエンドをバックアップするサーバー側のアプリケーション層を設定できます。これは、Amazon Cognito リソースへの特権アクセスで信頼する OAuth 機密クライアントです。ユーザーをアプリに登録するには、サーバーで AdminCreateUser API リクエストに AWS 認証情報を含めることができます。OAuth クライアントの種類の詳細については、OAuth 2.0 承認フレームワークの「Client Types」(クライアントの種類) を参照してください。

AWS CLI または AWS SDK でこれらのリクエストを承認するには、環境変数またはリクエストに IAM 認証情報を追加するクライアント設定を使用してサーバー側のアプリケーション環境を設定します。詳細については、「AWS 全般のリファレンス」の「AWS 認証情報を使用して AWS にアクセスする」を参照してください。Amazon Cognito ユーザープール API のサービスエンドポイントにリ

クエストを直接送信することもできます。リクエストのヘッダーに埋め込んだ AWS 認証情報を使用して、これらのリクエストを承認または署名する必要があります。詳細については、「AWS API リクエストの署名」を参照してください。

アプリケーションクライアントにクライアントシークレットがある場合は、IAM 認証情報と、オペレーションによっては、AuthParameters で SecretHash パラメータまたは SECRET_HASH 値の両方を指定する必要があります。詳細については、「シークレットハッシュ 値の計算」を参照してください。

IAM 認証済みユーザーオペレーション

AdminAddUserToGroup

AdminConfirmSignUp

AdminCreateUser

AdminDeleteUser

AdminDeleteUserAttributes

AdminDisableProviderForUser

AdminDisableUser

AdminEnableUser

AdminForgetDevice

AdminGetDevice

AdminGetUser

AdminInitiateAuth

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

IAM 認証済みユーザーオペレーション

`AdminRemoveUserFromGroup`

`AdminResetUserPassword`

`AdminRespondToAuthChallenge`

`AdminSetUserMFAPreference`

`AdminSetUserPassword`

`AdminSetUserSettings`

`AdminUpdateAuthEventFeedback`

`AdminUpdateDeviceStatus`

`AdminUpdateUserAttributes`

`AdminUserGlobalSignOut`

認証されていないユーザーオペレーション

認証されていないユーザーオペレーションでは、ユーザーのサインアップ、サインイン、およびパスワードのリセットが開始されます。インターネット上のすべてのユーザーがアプリにサインアップしてサインインできるようにする場合は、認証されていない API オペレーション、またはパブリック API オペレーションを使用します。

例えば、アプリケーションにユーザーを登録するには、シークレットへの特権アクセスを提供しない OAuth パブリッククライアントを配布できます。このユーザーは、認証されていない API オペレーション SignUp で登録できます。

AWS SDK で開発したパブリッククライアントでこれらのリクエストを送信する場合、認証情報を設定する必要はありません。Amazon Cognito ユーザープール API のサービスエンドポイントに、追加の認証なしでリクエストを直接送信することもできます。

アプリケーションクライアントにクライアントシークレットがある場合は、オペレーションによっては、`AuthParameters` で `SecretHash` パラメータまたは `SECRET_HASH` 値を指定する必要があります。詳細については、「シークレットハッシュ 値の計算」を参照してください。

認証されていないユーザーオペレーション

`SignUp`

`ConfirmSignUp`

`ResendConfirmationCode`

`ForgotPassword`

`ConfirmForgotPassword`

`InitiateAuth`

トークン認証によるユーザーオペレーション

トークン認証によるユーザーオペレーションは、ユーザーがサインインしたか、サインインプロセスを開始した後に、ユーザーのサインアウト、認証情報の管理、変更、表示を行います。アプリケーション内でシークレットを配布しない場合や、ユーザー自身の認証情報でリクエストを承認する場合は、トークン認証による API オペレーションを使用してください。ユーザーがサインインを完了した場合、ユーザーのトークン認証された API リクエストをアクセストークンで承認する必要があります。ユーザーがサインインプロセス中である場合は、Amazon Cognito が前のリクエストへのレスポンスで返したセッショントークンを使用して、トークン認証による API リクエストを承認する必要があります。

例えば、パブリッククライアントでは、書き込みアクセスをユーザー自身のプロファイルのみに制限する方法でユーザーのプロファイルを更新する場合があります。この更新を行うには、クライアントが UpdateUserAttributes API リクエストにユーザーのアクセストークンを含めることができます。

AWS SDK で開発したパブリッククライアントでこれらのリクエストを送信する場合、認証情報を設定する必要はありません。リクエストには `AccessToken` または `Session` パラメータを含めてください。Amazon Cognito ユーザープール API のサービスエンドポイントにリクエストを直接送信することもできます。サービスエンドポイントへのリクエストを承認するには、リクエストの POST 本文にアクセストークンまたはセッショントークンを含めます。

トークン認証オペレーションの API リクエストに署名するには、アクセストークンをリクエストの `Authorization` ヘッダーとして `Bearer` *`<Base64-encoded access token>`* 形式で含めます。

トークン認証によるユーザーオペレーション	AccessToken	セッション
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	

トークン認証によるユーザーオペレーション	AccessToken	セッション
SetUserMFAPreference	✓	
GlobalSignOut	✓	
AssociateSoftwareToken	✓	✓
UpdateAuthEventFeedback		✓
VerifySoftwareToken	✓	✓
RevokeToken [1]		

[1] RevokeToken は更新トークンをパラメータとして受け取ります。更新トークンは、認証トークンとして、またターゲットリソースとして機能します。

ユーザープール設定の更新

で Amazon Cognito ユーザープールの設定を変更するには AWS Management Console、このガイドの他の領域で説明されているように、ユーザープール設定の機能ベースのタブに移動し、フィールドを更新します。ユーザープールを作成した後に、この設定を変更することはできません。以下の設定を変更する場合は、新しいユーザープールまたはアプリクライアントを作成する必要があります。

ユーザープール名

API パラメータ名: PoolName

ユーザープールに割り当てた分かりやすい名前。ユーザープール名を変更するには、新しいユーザープールを作成します。

Amazon Cognito ユーザープールサインインオプション

API パラメータ名: AliasAttributes および UsernameAttributes

ユーザーがサインインするときにユーザー名として渡すことができる属性。ユーザープールを作成するときに、ユーザー名、E メールアドレス、電話番号、または優先ユーザー名を使用してサインインを許可するかどうかを選択できます。ユーザープールサインインオプションを変更するには、新しいユーザープールを作成します。

ユーザー名の大文字と小文字を区別する

API パラメータ名: UsernameConfiguration

大文字と小文字以外の別のユーザー名と一致するユーザー名を作成した場合、Amazon Cognito では同じユーザーまたは一意のユーザーとして扱うことができます。詳細については、「ユーザープールの大文字と小文字の区別」を参照してください。大文字と小文字の区別を変更するには、新しいユーザープールを作成します。

クライアントシークレット

API パラメータ名: GenerateSecret

アプリクライアントを作成するときに、信頼されたソースだけがユーザープールにリクエストを送信できるように、クライアントシークレットを生成できます。詳細については、「ユーザープールアプリクライアント」を参照してください。クライアントシークレットを変更するには、同じユーザープール内に新しいアプリクライアントを作成します。

必須属性

API パラメータ名: スキーマ

ユーザーのサインアップ時または作成時に値を提供する必要がある属性。詳細については、「ユーザープール属性」を参照してください。必要な属性を変更するには、新しいユーザープールを作成します。

カスタム属性

API パラメータ名: スキーマ

カスタム名を持つ属性です。ユーザーのカスタム属性の値を変更することはできますが、ユーザープールからカスタム属性を削除することはできません。詳細については、「ユーザープール属性」を参照してください。カスタム属性の最大数に達してリストを変更する場合は、新しいユーザープールを作成します。

SMS 設定

ユーザープールで SMS メッセージをアクティブ化した後は、非アクティブ化することはできません。

- ユーザープールの作成時に SMS メッセージを設定することを選択した場合、セットアップの完了後に SMS を非アクティブ化することはできません。
- 作成したユーザープールで SMS メッセージをアクティブ化することはできますが、それ以降は SMS を非アクティブ化することはできません。
- Amazon Cognito は、ユーザーアカウントの招待と復旧、属性検証、および多要素認証 (MFA) に SMS メッセージを使用できます。SMS メッセージをアクティブ化した後は、これらの関数に対していつでも SMS メッセージを有効または無効にできます。
- SMS メッセージ設定には、Amazon SNS でメッセージを送信するために Amazon Cognito Amazon SNSロールが含まれています。割り当てられたロールはいつでも変更できます。

AWS SDK、、 AWS CDKまたは REST API を使用したユーザープールの更新

Amazon Cognito コンソールでは、ユーザープール設定を一度に 1 つのパラメータで変更できます。例えば、Lambda トリガーを追加するには、Lambda トリガーを追加を選択し、関数とトリガータイプを選択します。Amazon Cognito ユーザープール API は、ユーザープールとアプリケーションクライアントの更新オペレーションで、ユーザープールのパラメータの完全なセットが必要になるように構造化されています。ただし、コンソールは、この更新オペレーションを他のユーザープール設定で透過的に自動化します。

内の他の場所で変更を行うと AWS アカウント 、変更する設定に関連しない更新でエラーが発生することがあります。例えば AWS WAF、削除された Amazon SES ID や の IAM アクセス許可の変更。現在のパラメータのいずれかが無効になった場合、修正するまで設定を更新することはできません。このようなエラーが発生した場合は、エラーレスポンスを調べて、言及されている設定を検証します。

AWS Cloud Development Kit (AWS CDK)、Amazon Cognito ユーザープール REST API および AWS SDKsは、Amazon Cognito リソースの自動化とプログラムによる設定のためのツールです。これらのツールを使用したリクエストは、Amazon Cognito コンソールと同様に、リクエスト本文の完全なリソース設定で設定を更新する必要があります。大まかに言うと、次のプロセスを実行する必要があります。

1. 既存のリソース の設定を記述する オペレーションから出力をキャプチャします。
2. 設定を変更して出力を変更します。
3. リソースを更新するオペレーションで、変更された設定を送信します。

次の手順では、 UpdateUserPool API オペレーションを使用して設定を更新します。入力フィールドが異なる同じアプローチが UpdateUserPoolClientに適用されます。

> **Important**
>
> 既存のパラメータの値を提供しない場合、Amazon Cognito はそれらをデフォルト値に設定します。例えば、既存の `LambdaConfig` がある場合に、空の `LambdaConfig` で `UpdateUserPool` を送信すると、すべての Lambda 関数のユーザープールトリガーへの割り当てが削除されます。ユーザープール設定の変更を自動化する場合は、適宜計画を立ててください。

1. を使用して、ユーザープールの既存の状態をキャプチャします DescribeUserPool。
2. `DescribeUserPool` の出力を `UpdateUserPool` のリクエストパラメータと一致するようにフォーマットします。出力される JSON から次のトップレベルフィールドとその子オブジェクトを削除します。
   - `Arn`
   - `CreationDate`
   - `CustomDomain`
     - UpdateUserPoolDomain API オペレーションでこのフィールドを更新します。
   - `Domain`
     - UpdateUserPoolDomain API オペレーションでこのフィールドを更新します。
   - `EmailConfigurationFailure`
   - `EstimatedNumberOfUsers`
   - `Id`
   - `LastModifiedDate`
   - `Name`
   - `SchemaAttributes`
   - `SmsConfigurationFailure`

- `Status`

3. 生成された JSON が `UpdateUserPool` のリクエストパラメータと一致することを確認します。
4. 生成された JSON で変更するパラメータを変更します。
5. 変更した JSON をリクエスト入力として使用する `UpdateUserPool` API リクエストを送信します。

また、 AWS CLIの `update-user-pool` の `--cli-input-json` パラメータで、この修正した `DescribeUserPool` の出力を使用することもできます。

または、次の AWS CLI コマンドを実行して、 の受け入れられた入力フィールドの値が空白の JSON を生成しますupdate-user-pool。その後、これらのフィールドにユーザープールの既存の値を入力できます。

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

次のコマンドを実行して、アプリクライアント用に同じ JSON オブジェクトを生成します。

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用

ドメインを持つ Amazon Cognito ユーザープールは、OAuth-2.0 準拠の認証サーバーであり、認証用の ready-to-use ホスト型ユーザーインターフェイス (UI) です。認証サーバーでは、認証リクエストのルーティング、JSON ウェブトークン (JWT) の発行、管理、ユーザー属性情報の配信が行われます。ホストされた UI は、ユーザープール内の基本的なサインアップ、サインイン、多要素認証、パスワードリセットのアクティビティを行う Web インターフェイスの集まりです。また、アプリに関連付けるサードパーティー ID プロバイダー (IdPs) による認証の中心的なハブでもあります。ユーザーを認証および承認する際、アプリはホストされた UI と承認エンドポイントを呼び出すことができます。独自の口ゴと CSS のカスタマイズを使うことで、ホスト UI のユーザーエクスペリエンスをブランドに合ったものにすることができます。ホストされた UI と認証サーバーのコンポーネントについて詳しくは、「ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス」を参照してください。

> **Note**
>
> Amazon Cognito でホストされた UI は、カスタム認証チャレンジの Lambda トリガーを使用したカスタム認証フローをサポートしていません。

トピック

- でホストされた UI を設定する AWS Amplify
- Amazon Cognito コンソールでの ホストされた UI のセットアップ
- サインインページの表示
- Amazon Cognito ユーザープールでホストされた UI について知っておくべきこと
- ユーザープールのドメインを設定する
- 組み込みのサインインおよびサインアップウェブページのカスタマイズ
- ホストされた UI でのサインアップとサインイン

でホストされた UI を設定する AWS Amplify

AWS Amplify を使用してウェブまたはモバイルアプリに認証を追加する場合は、 AWS Amplify フレームワークのコマンドラインインターフェイス (CLI) とライブラリを使用して、ホストされた UI を設定できます。アプリケーションに認証を追加するには、 AWS Amplify CLI を使用して `Auth` カテゴリをプロジェクトに追加します。次に、クライアントコードで、 AWS Amplify ライブラリを使用して Amazon Cognito ユーザープールでユーザーを認証します。

事前構築済みのホストされた UI を表示する、または Facebook、Google、Amazon、Apple などのソーシャルサインインプロバイダーにリダイレクトする OAuth 2.0 エンドポイント経由でユーザーをフェデレートすることができます。ユーザーがソーシャルプロバイダーで正常に認証されると、AWS Amplify は必要に応じてユーザープールに新しいユーザーを作成し、ユーザーの OIDC トークンをアプリケーションに提供します。

次の例は、 AWS Amplify を使用して、アプリ内のソーシャルプロバイダーでホストされた UI を設定する方法を示しています。

- AWS Amplify の認証 JavaScript。
- AWS Amplify Swift の 認証。
- AWS Amplify Flutter の認証。

- AWS Amplify Android の 認証。

Amazon Cognito コンソールでの ホストされた UI のセットアップ

アプリケーションクライアントの作成

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [App integration] (アプリケーションの統合) タブを選択します。
5. [App clients] (アプリケーションクライアント) で、[Create an app client] (アプリケーションクライアントの作成) を選択します。
6. [App type] (アプリケーションタイプ) を選択します。[Public client] (パブリッククライアント)、[Confidential client] (機密保持クライアント)、または []Other (その他)。パブリッククライアントは通常、ユーザーのデバイスから操作し、認証されていない API とトークン認証された API を使用します。機密クライアントは通常、クライアントシークレットと API 認証情報で信頼する中央サーバー上のアプリケーションから動作し、認証ヘッダーと AWS Identity and Access Management 認証情報を使用してリクエストに署名します。ユースケースが、パブリッククライアントまたは機密保持クライアントの事前構成されたアプリケーションクライアント設定と異なる場合、その他を選択します。
7. [App client name] (アプリケーションクライアント名) を入力します。
8. アプリケーションクライアントで許可したい [Authentication flows] (認証のフロー) を選択します。
9. 認証フローセッション持続期間を設定します。これは、セッショントークンが期限切れになる前にユーザーが各認証チャレンジを完了しなければならない期間です。
10. (オプション) トークンの有効期限を設定します。
    a. アプリケーションクライアントの [Refresh token expiration] (トークンの有効期限を更新) を指定します。デフォルト値は 30 日です。これは、1 時間から 10 年の任意の値に変更できます。
    b. アプリケーションクライアントの [Access token expiration] (アクセストークンの有効期限) を指定します。デフォルト値は 1 時間です。これは、5 分から 24 時間の任意の値に変更できます。

c. アプリケーションクライアントの [ID token expiration] (ID トークンの有効期限) を指定します。デフォルト値は 1 時間です。これは、5 分から 24 時間の任意の値に変更できます。

> **Important**
>
> ホストされた UI を使用してトークンの有効期間を 1 時間未満に設定した場合、ユーザーは、現在 1 時間に固定されているセッション cookie の期間に基づいてトークンを使用できるようになります。

11. [Generate client secret] (クライアントシークレットの生成) を選択して、 Amazon Cognito にクライアントシークレットを生成させます。クライアントシークレットは通常、機密クライアントに関連付けられます。
12. このアプリケーションクライアントに、[Enable token revocation] (トークンの失効を有効化) するかどうかを選択してください。これにより、トークンサイズが増加します。トークンの取り消しの詳細については、「Revoking Tokens」を参照してください。
13. このアプリケーションのクライアントに対して、[Prevent error messages that reveal user existence] (ユーザーの存在を知らせるエラーメッセージを表示しない) ようにするか選択します。Amazon Cognito は、ユーザー名またはパスワードのいずれかが間違っているという一般的なメッセージで、存在しないユーザーのサインインリクエストに応答します。
14. (オプション) このアプリケーションクライアントに、[Attribute read and write permissions] (属性の読み取りと書き込みのアクセス許可) を構成します。アプリケーションクライアントは、ユーザープールの属性スキーマの限定サブセットに対して、読み取りと書き込みの許可が得られます。
15. [Create] (作成) を選択します。
16. [Client id] (クライアント ID) を書き留めます。これにより、サインアップリクエストとサインインリクエストでアプリケーションクライアントを識別します。

アプリを設定する

1. [App integration] (アプリケーションの統合) タブで、[App clients] (アプリケーションクライアント) のアプリケーションクライアントを選択します。現在のホストされた UI 情報を確認してください。
2. [Allowed callback URL(s)] (許可されたコールバック URL) にコールバック URL を追加します。コールバック URL は、ユーザーがサインインに成功したときのリダイレクト先です。

3. [Allowed sign-out URL(s)] (許可されたサインアウト URL) にサインアウト URL を追加します。サインアウト URL は、ユーザーがサインアウトした後のリダイレクト先です。
4. ID プロバイダーリストから少なくとも 1 つ追加してください。
5. [OAuth 2.0 grant types] (OAuth 2.0 付与タイプ) で、[Authorization code grant] (認証コード付与) を選択して、ユーザープールトークンと交換される認証コードを返します。トークンはエンドユーザーに直接公開されないため、侵害される可能性は低くなります。ただし、ユーザープールトークンの認証コードを交換するために、バックエンドでカスタムアプリケーションが必要です。セキュリティ上の理由から、モバイルアプリケーション用の認証コード付与フローとコード交換用証明キー (PKCE) を併用することをお勧めします。
6. [OAuth 2.0 grant types] (OAuth 2.0 付与タイプ) で、[Implicit grant] (暗黙的付与) を選択し、Amazon Cognito からユーザープールの JSON Web トークン (JWT) が返されるようにします。このフローは、トークンの認証コードを交換できるバックエンドがない場合に使用でき、トークンのデバッグにも役立ちます。
7. [Authorization code] (認証コード) と [Implicit code] (暗黙的コード) 付与の両方を有効にして、必要に応じて各権限を使用することができます。[Authorization code] (認証コード) または [Implicit code] (暗黙コード) の付与も選択されておらず、アプリケーションクライアントにクライアントシークレットがある場合は、クライアント認証情報の付与を有効にできます。アプリケーションがユーザーの代わりではなく自身の代わりとしてアクセストークンをリクエストする必要がある場合にのみ、[Client credentials] (クライアント認証情報) を選択します。
8. このアプリケーションクライアントに対して認可する [OpenID Connect scopes] (OpenID Connect スコープ) を選択します。
9. [変更を保存] を選択します。

ドメインを設定する

1. ユーザープール用の [App integration] (アプリケーションの統合) タブに移動します。
2. [Domain] (ドメイン) の [Actions] (アクション) を選択し、[Create custom domain] (カスタムドメインの作成) または [Create Cognito domain] (Cognitoドメインの作成) を選択します。ユーザープールドメインを既に構成している場合は、新しいカスタムドメインを作成する前に、[Delete Cognito domain] (Cognito ドメインの削除) または [Delete custom domain] (カスタムドメインの削除) を選択して削除します。
3. Amazon Cognito ドメインで使用するための使用可能なドメインプレフィックスを入力します。[Custom domain] (カスタムドメイン) のセットアップに関する詳細については、「ホストされた UI で所有するドメインを使用する」を参照してください。
4. [Create] (作成) を選択します。

サインインページの表示

Amazon Cognito コンソールで、[App integration] (アプリの統合) タブの [App clients and analytics] (アプリクライアントと分析) の下にあるアプリクライアントの設定で、[View Hosted UI] (ホストされた UI を表示) ボタンを選択します。このボタンを選択すると、ホストされた UI のサインインページに移動し、次の基本パラメータが表示されます。

- アプリクライアント ID
- 認証コード付与リクエスト
- 現在のアプリクライアント用に有効にしたすべてのスコープのリクエスト
- 現在のアプリクライアントのリストにある最初のコールバック URL

[View hosted UI] (ホストされた UI を表示) ボタンは、ホストされた UI の基本機能をテストする場合に便利です。追加および変更されたパラメータを使用してサインイン URL をカスタマイズすることできます。ほとんどの場合、[View hosted UI] (ホストされた UI を表示) リンクの自動生成されたパラメータは、アプリのニーズに完全には一致しません。このような場合は、ユーザーのサインイン時にアプリが呼び出す URL をカスタマイズする必要があります。サインインパラメータのキーと値の詳細については、「ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス」を参照してください。

ホストされた UI のサインインウェブページは、以下の URL 形式を使用します。この例では、`response_type=code` パラメーターを使って認証コードの付与をリクエストしています。

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your callback url>
```

ユーザープールのドメイン文字列は、[アプリケーションの統合] タブから取得できます。同じタブで、アプリケーションクライアント ID、コールバック URL、許可されたスコープ、その他の設定を [アプリクライアントと分析] で確認できます。

カスタムパラメータを使用して `/oauth2/authorize` エンドポイントに移動すると、Amazon Cognito は `/oauth2/login` エンドポイントにリダイレクトするか、`identity_provider` または `idp_identifier` パラメータがある場合は、IdP サインインページにそのままリダイレクトします。ホストされた UI をバイパスする URL の例については、「Amazon Cognito ユーザープールでの SAML セッション開始」を参照してください。

暗黙的な付与に対するホストされた UI リクエストの例

response_type=token である暗黙的なコード付与の次の URL を使用して、ホストされた UI サインインウェブページを表示できます。サインインが正常に行われると、Amazon Cognito がユーザープールトークンをウェブブラウザのアドレスバーに返します。

```
            https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=token&client_id=1example23456789&redirect_uri=https://
mydomain.example.com
```

ID トークンとアクセストークンは、リダイレクト URL に追加されるパラメーターとして表示されます。

ここでは、暗黙的な許可リクエストからのレスポンス例を示します。

```
            https://mydomain.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

Amazon Cognito ユーザープールでホストされた UI について知っておくべきこと

ホストされた UI と管理者としてのユーザーの確認

ユーザープールのローカルユーザーの場合、ホストされた UI は、検証および確認用のメッセージの自動送信を Cognito に許可するようにユーザープールを設定するときに最も効果的です。この設定を有効にすると、Amazon Cognito は、サインアップしたユーザーに確認コードを含むメッセージを送信します。代わりにユーザーをユーザープール管理者として確認すると、ホストされた UI はサインアップ後にエラーメッセージを表示します。この状態で、Amazon Cognito は新しいユーザーを作成しましたが、検証メッセージを送信できませんでした。引き続きユーザーを管理者として確認することはできますが、エラーが発生すると、ユーザーはサポートデスクに連絡する可能性があります。管理者の確認の詳細については、「ユーザーにアプリケーションへのサインアップを許可するがユーザープール管理者として確認する」を参照してください。

ホストされた UI 設定の変更の表示

ホストされている UI ページに変更がすぐに反映されない場合は、数分間待ってから、ページを更新してください。

ユーザープールトークンのデコード

Amazon Cognito ユーザープールトークンは、RS256 アルゴリズムを使用して署名されます。を使用してユーザープールトークンをデコードおよび検証できます。「 での Amazon Cognito JWT トークンのデコードおよび検証 AWS Lambda」を参照してください GitHub。

ホストされた UI と TLS のバージョン

ホストされた UI には、転送中の暗号化が必要です。Amazon Cognito が提供するユーザープールドメインには、1.2 の最小 TLS バージョンが必要です。カスタムドメインは をサポートしていますが、TLS バージョン 1.2 は必要ありません。Amazon Cognito はホストされた UI と認証サーバーのエンドポイントの設定を管理するため、ユーザープールドメインの TLS 要件を変更することはできません。

ホストされた UI と CORS ポリシー

Amazon Cognito がホストする UI は、カスタムの Cross-Origin Resource Sharing (CORS) オリジンポリシーをサポートしていません。ホストされた UI の CORS ポリシーは、ユーザーがリクエストの認証パラメータを渡すことを防ぎます。代わりに、アプリのウェブフロントエンドに CORS ポリシーを実装します。Amazon Cognito は、次の OAuth エンドポイントへのリクエストに対して `Access-Control-Allow-Origin: *` レスポンスヘッダーを返します。

1. トークンエンドポイント
2. エンドポイントの取り消し
3. UserInfo エンドポイント

ホストされた UI と認証サーバーの Cookie

Amazon Cognito ユーザープールエンドポイントは、ユーザーのブラウザに Cookie を設定します。Cookie は、サイトがサードパーティー Cookie を設定しない一部のブラウザの要件に準拠しています。これらはユーザープールエンドポイントのみを対象とし、以下が含まれます。

- 各リクエストの `XSRF-TOKEN` Cookie。
- ユーザーがリダイレクトされたときのセッション整合性のための `csrf-state`Cookie。
- 成功したサインイン試行を 1 時間保持する`cognito`セッション Cookie。

ユーザープールのドメインを設定する

アプリクライアントを設定したら、サインアップおよびサインインのウェブページアドレスを設定できます。Amazon Cognito でホストされるドメインを使用して、利用可能なドメインプレフィックスを選択する、または独自のウェブアドレスをカスタムドメインとして使用することができます。

AWS Management Console を使用してアプリケーションクライアントと Amazon Cognito でホストされるドメインを追加するには、「ホストされたウェブ UI を有効にするためのアプリケーションの追加」を参照してください。

> **Note**
>
> ドメインプレフィックスで `aws`、`amazon`、または `cognito` テキストを使用することはできません。

トピック

- ホストされた UI への Amazon Cognito ドメインの使用
- ホストされた UI への独自のドメインの使用

ホストされた UI への Amazon Cognito ドメインの使用

アプリケーションクライアントを設定したら、サインアップおよびサインインのウェブページアドレスを設定できます。ホストされた Amazon Cognito ドメインを独自のドメインプレフィックスと共に使用できます。

> **Note**
>
> Amazon Cognito アプリケーションのセキュリティを強化するために、ユーザープールエンドポイントの親ドメインはパブリックサフィックスリスト (PSL) に登録されます。PSL は、ユーザーのウェブブラウザが、ユーザーが設定するユーザープールエンドポイントと Cookie を一貫して把握するのに役立ちます。
> ユーザープールエンドポイントの親ドメインには次の形式があります。
>
> ```
> auth.Region.amazoncognito.com
> auth-fips.Region.amazoncognito.com
> ```

でアプリケーションクライアントと Amazon Cognito がホストするドメインを追加するには AWS Management Console、「」を参照してくださいアプリクライアントの作成。

トピック

前提条件

開始するには、以下が必要です。

- アプリクライアントを持つユーザープール。詳細については、「ユーザープールの開始方法」を参照してください。

ステップ 1: ホストされるユーザープールドメインを設定する

ホストされたユーザープールドメインを設定する

ユーザープールドメインを設定するには、 AWS Management Console または AWS CLI または API のいずれかを使用できます。

Amazon Cognito console

ドメインを設定する

1. ユーザープール用の [App integration] (アプリケーションの統合) タブに移動します。
2. [ドメイン] の横で、[アクション] を選択し、[カスタムドメインの作成] または [Amazon Cognito ドメインの作成] のどちらかを選択します。ユーザープールドメインを既に設定している場合は、新しいカスタムドメインを作成する前に、[Amazon Cognitoドメインの削除] または [カスタムドメインの削除] を選択します。
3. Amazon Cognito ドメインで使用するための使用可能なドメインプレフィックスを入力します。[Custom domain] (カスタムドメイン) のセットアップに関する詳細については、「ホストされた UI で所有するドメインを使用する」を参照してください。
4. [Create] (作成) を選択します。

CLI/API

ドメインプレフィックスを作成してユーザープールに割り当てるには、次のコマンドを使用します。

ユーザープールのドメインを設定する

- AWS CLI: `aws cognito-idp create-user-pool-domain`

  例: `aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name>`
- AWS API: CreateUserPoolDomain

ドメインに関する情報を取得する

- AWS CLI: `aws cognito-idp describe-user-pool-domain`

  例: `aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- AWS API: DescribeUserPoolDomain

ドメインを削除する

- AWS CLI: `aws cognito-idp delete-user-pool-domain`

  例: `aws cognito-idp delete-user-pool-domain --domain <domain_name>`
- AWS API: DeleteUserPoolDomain

ステップ 2: サインインページを検証する

- Amazon Cognito でホストされるドメインから、サインインページが利用可能であることを検証します。

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

ドメインは、Amazon Cognito コンソールの [ドメイン名] ページに表示されます。アプリクライアント ID およびコールバック URL は [アプリクライアントの設定] ページに表示されています。

ホストされた UI への独自のドメインの使用

アプリクライアントをセットアップしたら、Amazon Cognito がホストする UI および auth APIエンドポイントのカスタムドメインを使用してユーザープールを設定できます。カスタムドメインを使用すると、ユーザーは、独自のウェブアドレスを使用して、アプリケーションにサインインできるようになります。

トピック

- ユーザープールへのカスタムドメインの追加
- カスタムドメインの SSL 証明書を変更する

ユーザープールへのカスタムドメインの追加

カスタムドメインをユーザープールに追加するには、Amazon Cognito コンソールでドメイン名を指定し、AWS Certificate Manager (ACM) を使用してユーザー管理の証明書を提供します。ドメインを追加したら、Amazon Cognito がエイリアスターゲットを提供するので、これを DNS 設定に追加します。

前提条件

開始するには、以下が必要です。

- アプリクライアントを持つユーザープール。詳細については、「ユーザープールの開始方法」を参照してください。
- 所有するウェブドメイン。その親ドメインには、DNS に有効な A レコードが必要です。このレコードには任意の値を割り当てることができます。親は、ドメインのルート、またはドメイン階層の 1 つ上の子ドメインです。例えば、カスタムドメインが auth.xyz.example.com の場合、Amazon Cognito は xyz.example.com を IP アドレスに解決できる必要があります。お客様のインフラストラクチャへの偶発的な影響を防ぐため、Amazon Cognito は、最上位ドメイン (TLD) のカスタムドメインへの使用をサポートしていません。詳細については、「ドメイン名」を参照してください。
- カスタムドメインのサブドメインを作成する機能。サブドメインとして [auth] を使用することをお勧めします。たとえば、「*auth.example.com*」を使用します。

> **Note**
>
> 「ワイルドカード証明書」がない場合は、カスタムドメインのサブドメイン用に新しい証明書の取得が必要な場合があります。

- ACM によって管理されている Secure Sockets Layer (SSL) 証明書。

> **Note**
>
> 証明書をリクエストまたはインポートする前に、ACM コンソールで AWS リージョンを米国東部 (バージニア北部) に変更する必要があります。

- ユーザープール認証サーバーがユーザーセッションに Cookie を追加することを許可するアプリケーション。Amazon Cognito は、ホストされた UI に必要な複数の Cookie を設定します。たとえばcognito、cognito-fl、およびXSRF-TOKENに適用されます。個々の Cookie はブラウザのサイズ制限に準拠していますが、ユーザープールの設定を変更すると、ホストされた UI Cookie のサイズが大きくなる可能性があります。カスタムドメインの前の Application Load Balancer (ALB) などの中間サービスでは、最大ヘッダーサイズまたは合計 Cookie サイズが適用される場合があります。アプリケーションが独自の Cookie も設定している場合、ユーザーのセッションはこれらの制限を超える可能性があります。サイズ制限の競合を避けるため、アプリケーションはホストされた UI サブドメインに Cookie を設定しないことをお勧めします。
- Amazon CloudFront ディストリビューションを更新するアクセス許可。そのためには、次の IAM ポリシーステートメントを AWS アカウントのユーザーにアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

でのアクションの承認の詳細については CloudFront、「 でのアイデンティティベースのポリシー (IAM ポリシー) の使用 CloudFront」を参照してください。

Amazon Cognito は、最初は IAM アクセス許可を使用してディストリビューションを設定します CloudFrontが、ディストリビューションは によって管理されます AWS。Amazon Cognito がユーザープールに関連付けた CloudFront ディストリビューションの設定を変更することはできません。例えば、セキュリティポリシーでサポートされている TLS バージョンを更新することはできません。

ステップ 1: カスタムドメイン名を入力する

ドメインは、Amazon Cognito コンソールまたは API を使用することでユーザープールに追加できます。

Amazon Cognito console

Amazon Cognito コンソールからユーザープールにドメインを追加する。

1. Amazon Cognito コンソールにサインインします。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User pools] (ユーザープール) を選択します。
3. ドメインを追加するユーザープールを選択します。
4. [App integration] (アプリケーションの統合) タブを選択します。
5. [Domain] (ドメイン) の横にある[Actions] (アクション) を選択し、[Create custom domain] (カスタムドメインの作成) を選択します。

   > **Note**
   >
   > ユーザープールドメインを既に構成している場合は、新しいカスタムドメインを作成する前に、[Delete Cognito domain] (Cognito ドメインの削除) または [Delete custom domain] (カスタムドメインの削除) を選択して既存のドメインを削除します。

6. [Custom domain] (カスタムドメイン) には、Amazon Cognito で使用するドメインの URL を入力します。ドメイン名で使用できるのは、小文字の英文字、数字、およびハイフンのみです。最初または最後の文字にハイフンを使用しないでください。サブドメイン名は、ピリオドで区切ります。

7. [ACM certificate] (ACM 証明書) には、ドメインに使用する SSL 証明書を選択します。ユーザープールの に関係なく、Amazon Cognito カスタムドメインで使用できるのは、米国東部 (バージニア北部) AWS リージョン の ACM 証明書のみです。

   利用可能な証明書がない場合は、ACM を使用して米国東部 (バージニア北部) にプロビジョニングできます。詳細については、AWS Certificate Manager ユーザーガイドの「使用開始」を参照してください。
8. [作成] を選択します。
9. Amazon Cognito は、[App integration] (アプリケーションの統合) タブに戻ります。メッセージ「Create an alias record in your domain's DNS」(ドメインの DNS にエイリアスレコードを作成する) が表示されます。コンソールに表示される [Domain] (ドメイン) と [Alias target] (エイリアスターゲット) を書き留めます。これらは、次のステップでカスタムドメインにトラフィックを転送するために使用します。

API

Amazon Cognito API を使用して、ドメインをユーザープールに追加する

- CreateUserPoolDomain アクションを使用します。

ステップ 2: エイリアスターゲットとサブドメインを追加する

このステップでは、ドメイン名サーバー (DNS) サービスプロバイダーを使用して、前のステップのエイリアスターゲットを指すエイリアスを設定します。DNS アドレスの解決に Amazon Route 53 を使用している場合は、Route 53 を使用してエイリアスターゲットとサブドメインを追加するセクションを選択します。

エイリアスターゲットおよびサブドメインを現在の DNS 設定に追加する

- DNS アドレス解決に Route 53 を使用していない場合は、DNS サービスプロバイダーの設定ツールを使用して、前のステップで設定したエイリアスターゲットをドメインの DNS レコードに追加する必要があります。また、DNS プロバイダーで、カスタムドメインのサブドメインを設定する必要があります。

Route 53 を使用してエイリアスターゲットとサブドメインを追加する

1. Route 53 コンソールにサインインします。プロンプトが表示されたら、 AWS 認証情報を入力します。

2. Route 53 にホストゾーンがない場合は、カスタムドメインの親であるルートでホストゾーンを作成します。詳細については、以下を参照してください。

   a. [ホストゾーンの作成] を選択します。

   b. ドメイン名のリストから、カスタムドメイン *myapp.auth.example.com* の親ドメイン *auth.example.com* を入力します。

   c. ホストゾーンの説明を入力します。

   d. パブリックホストゾーンのホストゾーン [Type] (タイプ) を選択して、パブリッククライアントがカスタムドメインを解決できるようにします。Private hosted zone (プライベートホストゾーン) はサポートされていません。

   e. 必要に応じてタグを適用します。

   f. [ホストゾーンの作成] を選択します。

   > **Note**
   >
   > また、カスタムドメインの新しいホストゾーンを作成し、親ホストゾーンに委任セットを作成して、サブドメインのホストゾーンにクエリを送信することもできます。それ以外の場合は、A レコードを作成します。この方法により、ホストゾーンの柔軟性とセキュリティが向上します。詳細については、「Amazon Route 53 を通してホストされるドメインのサブドメインの作成」をご覧ください。

3. [ホストゾーン] ページで、ホストゾーンの名前を選択します。

4. カスタムドメインの親ドメインの DNS レコードがまだない場合は、追加します。親ドメインの DNS Aレコードを追加し、レコードの作成 を選択します。以下に、ドメイン *auth.example.com* のレコードの例を示します。

   ```
   auth.example.com. 60 IN A 198.51.100.1
   ```

   > **Note**
   >
   > Amazon Cognito は、本番ドメインの予想外の乗っ取りから保護するために、カスタムドメインの親ドメインで使用する DNS レコードがあることを確認します。親ドメインの DNS レコードがない場合、カスタムドメインを設定しようとすると、 Amazon Cognito はエラーを返します。Start of Authority (SOA) レコードは、親ドメインの検証のために十分な DNS レコードではありません。

5. カスタムドメインの DNS レコードを追加します。レコードは、カスタムドメインエイリアスターゲット をポイントする必要があります`123example.cloudfront.net`。例えば、。[Create Record] (レコードの作成) を選択します。
6. カスタムドメインに一致する [レコード名] (レコード名) を入力します。例えば、*myapp* は、*myapp.auth.example.com* のレコードを作成します。
7. Alias (エイリアス) オプションを有効にします。
8. [Route traffic to] (トラフィックのルーティング先) で、[Alias to CloudFront distribution] (CloudFront ディストリビューションへのエイリアス) を選択します。カスタムドメインの作成時に、Amazon Cognito から提供された [Alias target] (エイリアスターゲット) を入力します。
9. [レコードを作成] を選択します。

> **Note**
>
> 新しいレコードがすべての Route 53 の DNS サーバーに伝播されるまでに約 60 秒かかる場合があります。Route 53 GetChange API メソッドを使用して、変更が反映されたことを確認できます。

ステップ 3: サインインページを検証する

- カスタムドメインでサインインページが表示できることを確認します。

  このアドレスをブラウザに入力して、カスタムドメインとサブドメインでサインインします。以下は、サブドメイン *auth* を使用したカスタムドメイン *example.com* のサンプル URL です。

```
https://myapp.auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

カスタムドメインの SSL 証明書を変更する

必要な場合は、Amazon Cognito を使用して、カスタムドメインに適用した証明書を変更することができます。

通常、これは ACM での定期的な証明書更新後には不要です。ACM で既存の証明書を更新するときは、証明書の ARN がそのまま維持され、カスタムドメインは自動的に新しい証明書を使用します。

ただし、既存の証明書を新しい証明書に置き換える場合は、ACM が新しい証明書に新しい ARN を提供します。新しい証明書をカスタムドメインに適用するには、この ARN を Amazon Cognito に提供する必要があります。

新しい証明書の提供後、Amazon Cognito では、それがカスタムドメインに配布されるまで最大 1 時間かかります。

> **開始する前に**
>
> Amazon Cognito で証明書を変更する前に、証明書を ACM に追加する必要があります。詳細については、AWS Certificate Manager ユーザーガイドの「使用開始」を参照してください。
> 証明書を ACM に追加するときは、 AWS リージョンとして米国東部 (バージニア北部) を選択する必要があります。

証明書は、Amazon Cognito コンソールまたは API を使用して変更できます。

AWS Management Console

Amazon Cognito コンソールで証明書を更新する

1. にサインイン AWS Management Console し、 で Amazon Cognito コンソールを開きますhttps://console.aws.amazon.com/cognito/home。
2. [User Pools] (ユーザープール) を選択します。
3. 証明書を更新するユーザープールを選択します。
4. [App integration] (アプリケーションの統合) タブを選択します。
5. [Actions] (アクション) 、[Edit ACM certificate] (ACM 証明書の編集) を選択します。
6. カスタムドメインに関連付ける新しい証明書を選択します。
7. [変更を保存] を選択します。

API

証明書を更新する (Amazon Cognito API)

- UpdateUserPoolDomain アクションを使用します。

組み込みのサインインおよびサインアップウェブページのカスタマイズ

AWS Management Console、または AWS CLI や API を使用して、組み込みアプリの UI エクスペリエンスのカスタマイズ設定を指定します。アプリに表示するカスタムのロゴイメージをアップロードできます。カスケードスタイルシート (CSS) を使用して UI をカスタマイズすることもできます。

単一のクライアント (`clientId`) またはすべてのクライアント (`clientId` を `ALL` に設定) にアプリ UI のカスタマイズ設定を指定できます。`ALL` を指定した場合は、以前に UI のカスタマイズが設定されていないすべてのクライアントにデフォルトの設定が使用されます。特定のクライアントに UI のカスタマイズ設定が指定される場合、`ALL` 設定にフォールバックされなくなります。

UI のカスタマイズを設定するリクエストのサイズは 135 KB を超えてはなりません。まれに、リクエストヘッダー、CSS ファイル、ロゴの合計が 135 KB を超えることがあります。Amazon Cognito はイメージファイルを Base64 にエンコードします。これにより、100 KB のイメージのサイズが 130 KB に増え、リクエストヘッダーと CSS 用に 5 KB が残ります。リクエストが大きすぎる場合、AWS Management Console または `SetUICustomization` API `request parameters too large` リクエストはエラーを返します。ロゴイメージを 100 KB 以下、CSS ファイルを 3 KB 以下に調整します。CSS とロゴのカスタマイズを個別に設定することはできません。

> **Note**
>
> UI をカスタマイズするには、ユーザープールのドメインを設定する必要があります。

アプリケーションのカスタムロゴを指定する

Amazon Cognito は、ログインエンドポイント の入力フィールドの上の中央にカスタムロゴを配置します。

ホストされているカスタムの UI ロゴには、350 x 178 ピクセルまで拡大できる PNG、JPG、または JPEG ファイルを選択します。ロゴファイルのサイズは 100 KB 以下で、Amazon Cognito が Base64 にエンコードした後は 130 KB 以下にしてください。API で SetUICustomization に `ImageFile` を設定するには、ファイルを Base64 でエンコードされたテキスト文字列に変換するか、AWS CLI でファイルパスを指定して、Amazon Cognito にエンコードさせます。

アプリケーションの CSS カスタマイズを指定する

ホストされたアプリページの CSS をカスタマイズできます。ただし、以下の制限があります。

- 以下の CSS クラス名を使用できます。
  - background-customizable
  - banner-customizable
  - errorMessage-customizable
  - idpButton-customizable
  - idpButton-customizable:hover
  - idpDescription-customizable
  - inputField-customizable
  - inputField-customizable:focus
  - label-customizable
  - legalText-customizable
  - logo-customizable
  - passwordCheck-valid-customizable
  - passwordCheck-notValid-customizable
  - redirect-customizable
  - socialButton-customizable
  - submitButton-customizable
  - submitButton-customizable:hover
  - textDescription-customizable
- プロパティ値は HTML を含めることができます。ただし、@import,@supports,@page, または@mediaステートメント、または Javascript の値を除きます。

次の CSS プロパティをカスタマイズできます。

Labels

- [font-weight] は 100 ~ 900 までの 100 の倍数です。

入力フィールド

- [width] は含まれるブロックの幅に対してパーセンテージで表したものです。
- [height] は入力フィールドの高さ (ピクセル (px)) です。
- [color] はテキストの色です。任意の標準 CSS 色値を使用できます。
- [background-color] は、入力フィールドの背景色です。任意の標準 CSS 色値を使用できます。

- [border] は、アプリウィンドウの境界の幅、透明度、色を指定する標準 CSS 値です。幅は 1 px から 100 px までの任意の値を使用できます。透明度は solid または none です。色は任意の標準色値を使用できます。

テキストの説明

- [padding-top] は、説明欄の上のパディング量です。
- [padding-bottom] は、説明欄の下のパディング量です。
- [display] には `block` または `inline` のいずれかを設定できます。
- [font-size] はテキストの説明のフォントサイズです。

送信ボタン

- [font-size] はボタンテキストのフォントサイズです。
- [font-weight] はボタンテキストのフォントウエイトです。`bold`、`italic`、または `normal` です。
- [margin] はボタンの上下左右のマージンサイズを示す 4 つの値の文字列です。
- [font-size] はテキストの説明のフォントサイズです。
- [width] はボタンテキストのブロックに対する幅 (パーセント) です。
- [height] はボタンの高さ (ピクセル (px)) です。
- [color] はボタンの色です。任意の標準 CSS 色値を使用できます。
- [background-color] は、ボタンの背景色です。任意の標準色値を使用できます。

バナー

- [padding] はバナーの上下左右のパディングサイズを示す 4 つの値の文字列です。
- [background-color] は、バナーの背景色です。任意の標準 CSS 色値を使用できます。

送信ボタンのホバー

- [color] は、ボタンにカーソルを合わせたときのボタンの前景色です。任意の標準 CSS 色値を使用できます。
- [background-color] は、ボタンにカーソルを合わせたときのボタンの背景色です。任意の標準 CSS 色値を使用できます。

ID プロバイダーボタンのホバー

- [color] は、ボタンにカーソルを合わせたときのボタンの前景色です。任意の標準 CSS 色値を使用できます。
- [background-color] は、ボタンにカーソルを合わせたときのボタンの背景色です。任意の標準 CSS 色値を使用できます。

パスワードのチェックが有効ではありません

- [color] は "Password check not valid" メッセージのテキスト色です。任意の標準 CSS 色値を使用できます。

背景

- [background-color] は、アプリウィンドウの背景色です。任意の標準 CSS 色値を使用できます。

エラーメッセージ

- [margin] は上下左右のマージンサイズを示す 4 つの値の文字列です。
- [padding] はパディングサイズです。
- [font-size] はフォントサイズです。
- [width] はエラーメッセージの幅を含まれるブロックに対するパーセンテージで表したものです。
- [background] は、エラーメッセージの背景色です。任意の標準 CSS 色値を使用できます。
- [border] は境界の幅、透明度、色を指定する 3 つの値の文字列です。
- [color] はエラーメッセージのテキスト色です。任意の標準 CSS 色値を使用できます。
- [box-sizing] は含める必要があるサイズのプロパティ (幅と高さ) をブラウザに指示するために使用されます。

ID プロバイダーボタン

- [height] はボタンの高さ (ピクセル (px)) です。
- [width] はボタンテキストの幅を含まれるブロックに対するパーセンテージで示したものです。
- [text-align] はテキストの整列設定です。`left`、`right` または `center` のいずれかを設定できます。
- [margin-bottom] は下側マージンの設定です。
- [color] はボタンの色です。任意の標準 CSS 色値を使用できます。
- [background-color] は、ボタンの背景色です。任意の標準 CSS 色値を使用できます。
- [border-color] は、ボタンの境界の色です。任意の標準 CSS 色値を使用できます。

ID プロバイダーの説明

- [padding-top] は、説明欄の上のパディング量です。
- [padding-bottom] は、説明欄の下のパディング量です。
- [display] には `block` または `inline` のいずれかを設定できます。

- [font-size] は説明のフォントサイズです。

法務関連のテキスト

- [color] はテキストの色です。任意の標準 CSS 色値を使用できます。
- [font-size] はフォントサイズです。

> **Note**
>
> リーガルテキストをカスタマイズする際は、サインインページのソーシャル ID プロバイダーに表示されるメッセージ「We won't post to any of your accounts without asking first」(最初に確認せずにアカウントに投稿することはありません) をカスタマイズします。

ロゴ

- [max-width] は最大幅を含まれるブロックに対するパーセンテージで表したものです。
- [max-height] は最大高を含まれるブロックに対するパーセンテージで表したものです。

入力フィールドのフォーカス

- [border-color] は、入力フィールドの色です。任意の標準 CSS 色値を使用できます。
- [outline] は入力フィールドの境界線の幅 (ピクセル) です。

ソーシャルボタン

- [height] はボタンの高さ (ピクセル (px)) です。
- [text-align] はテキストの整列設定です。`left`、`right` または `center` のいずれかを設定できます。
- [width] はボタンテキストの幅を含まれるブロックに対するパーセンテージで示したものです。
- [margin-bottom] は下側マージンの設定です。

パスワードのチェックが有効です

- [color] は `"Password check valid"` メッセージのテキスト色です。任意の標準 CSS 色値を使用できます。

ユーザープールのアプリケーション UI カスタマイズ設定の指定 (AWS Management Console)

AWS Management Consoleを使用して、アプリの UI カスタマイズ設定を指定できます。

> **Note**
>
> 以下の URL を、ユーザープールの指定を含めて作成し、ブラウザに入力することで、カスタマイズが反映されているホストされた UI を表示できます。 `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callbac`
> 変更がコンソールに反映されるまで最大 1 分待ってからブラウザを最新表示にする必要がある場合があります。
> ドメインは、[Domain] (ドメイン) の下の [App integration] (アプリケーション統合) タブに表示されます。アプリケーションクライアント ID およびコールバック URL は [App clients] (アプリケーションクライアント) に表示されています。

アプリ UI のカスタマイズ設定を指定する

1. Amazon Cognito コンソールにサインインします。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [App integration] (アプリケーションの統合) タブを選択します。
4. すべてのアプリケーションクライアントの UI 設定をカスタマイズするには、[Hosted UI customization] (ホストされた UI のカスタマイズ) を探し、[Edit] (編集) を選択します。
5. 1 つのアプリケーションクライアントの UI 設定をカスタマイズするには、[App clients] (アプリケーションクライアント) を探し、変更したいアプリケーションクライアントを選択します。[ホストされた UI のカスタマイズ] を探し、[編集] を選択します。アプリケーションクライアントをユーザープールのデフォルトのカスタマイズからクライアント固有のカスタマイズに切り替えるには、[Use client-level settings] (クライアントレベルの設定を使用する) を選択します。
6. 独自のロゴイメージファイルをアップロードするには、[Choose file] (ファイルを選択) または [Replace current file] (現在のファイルを置き換える) を選択します。
7. ホストされた UI CSS をカスタマイズするには、CSS template.css をダウンロードし、テンプレートをカスタマイズ値に変更します。ホストされた UI で使用できるのは、テンプレートに含まれるキーのみです。追加された CSS キーは UI に反映されません。CSS ファイルをカスタマイズしたら、[Choose file] (ファイルの選択) または [Replace current file] (現在のファイルを置き換える) を選択し、カスタム CSS ファイルをアップロードします。

ユーザープールのアプリケーション UI カスタマイズ設定の指定 (AWS CLI および AWS API)

次のコマンドを使用して、ユーザープールのアプリ UI カスタマイズ設定を指定します。

ユーザープールの組み込みアプリ UI の UI カスタマイズ設定を取得するには、次の API オペレーションを使用します。

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API:GetUICustomization

ユーザープールの組み込みアプリ UI の UI カスタマイズ設定を設定するには、次の API オペレーションを使用します。

- イメージファイルからの AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file fileb://"<path-to-logo-image-file>" --css ".label-customizable{ color: <color>;}"`
- Base64 バイナリテキストとしてエンコードされたイメージを含む AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <base64-encoded-image-file> --css ".label-customizable{ color: <color>;}"`
- AWS API:SetUICustomization

ホストされた UI でのサインアップとサインイン

ユーザープールとアプリクライアント用に Amazon Cognito でホストされた UI を設定およびカスタマイズすると、アプリはそれをユーザーに提示できます。ホストされた UI は、以下の例を含む複数の Amazon Cognito 認証オペレーションをサポートしています。

- アプリの新しいユーザーとしてのサインアップ
- E メールアドレスおよび電話番号の検証
- 多要素認証 (MFA) のセットアップ
- ローカルユーザー名とパスワードを使用してサインインします。
- サードパーティーの ID プロバイダー (IdP) でのサインイン

- パスワードのリセット

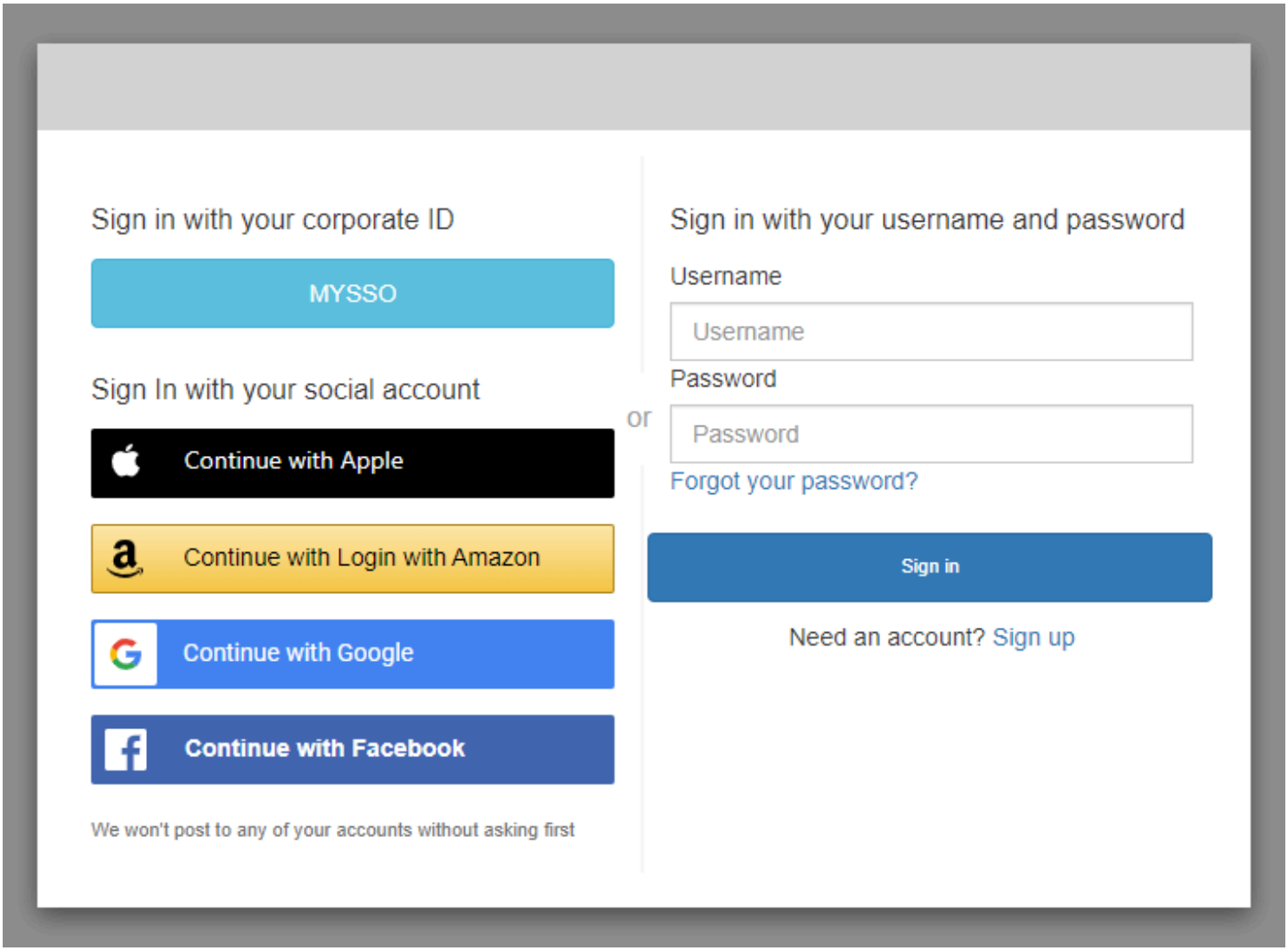
Amazon Cognito でホストされた UI は、ログインエンドポイント で始まります。サインインページの URL は、ユーザープール用に選択したドメインと、発行したい OAuth 2.0 付与、アプリクライアント、アプリへのパス、リクエストしたい OpenID Connect (OIDC) スコープを反映したパラメータの組み合わせになります。

```
https://<your user pool domain>/authorize?client_id=<your app client
 ID>&response_type=<code/token>&scope=<scopes to request>&redirect_uri=<your callback
 URL>
```

次の URL は、上記のプレースホルダーフィールドをサンプル値に置き換えます。

```
https://auth.example.com/authorize? /
client_id=1example23456789 /
&response_type=code /
&scope=aws.cognito.signin.user.admin+email+openid+profile /
&redirect_uri=https%3A%2F%2Faws.amazon.com
```

Amazon Cognito でホストされた UI のサインインページには、ユーザープールまたはユーザーがリクエストしているアプリクライアントに割り当てた任意の ID プロバイダー (IdP) を通してサインインするオプションがあります。また、ユーザープールに新しいユーザーアカウントをサインアップしたり、忘れたパスワードをリセットしたりするためのリンクも含まれています。



トピック

- Amazon Cognito でホストされた UI での新規アカウントのサインアップの方法
- Amazon Cognito でホストされた UI でサインインする方法
- Amazon Cognito でホストされた UI を使用してパスワードをリセットする方法

Amazon Cognito でホストされた UI での新規アカウントのサインアップの方法

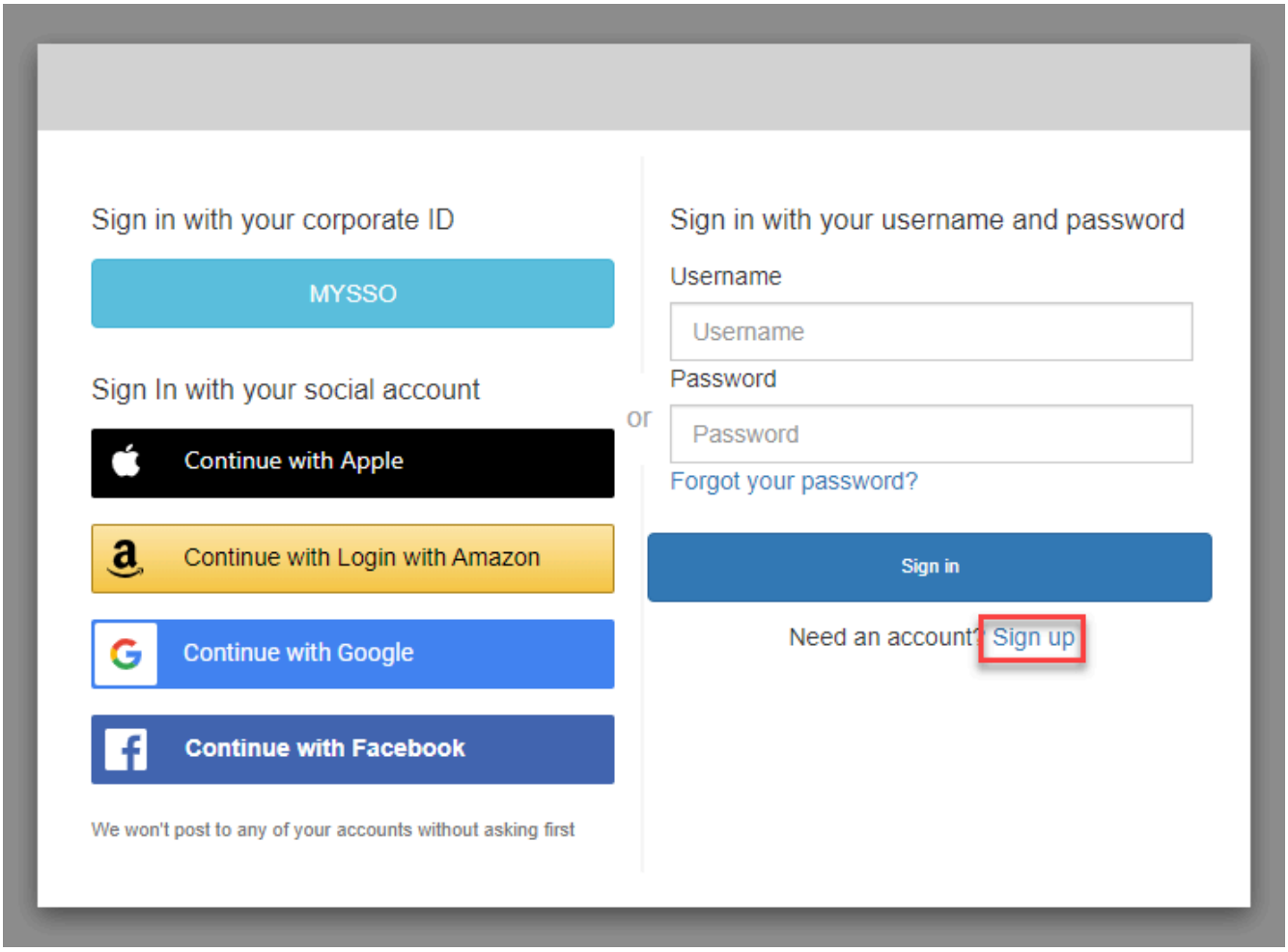
このガイドでは、Amazon Cognito を使用するアプリでユーザーアカウントにサインアップする方法を示します。

> **Note**
>
> Amazon Cognito でホストされるユーザーインターフェイス (UI) を使用するアプリにサインインすると、このガイドに示されている基本設定以外に、アプリ所有者がカスタマイズしたページが表示される場合があります。

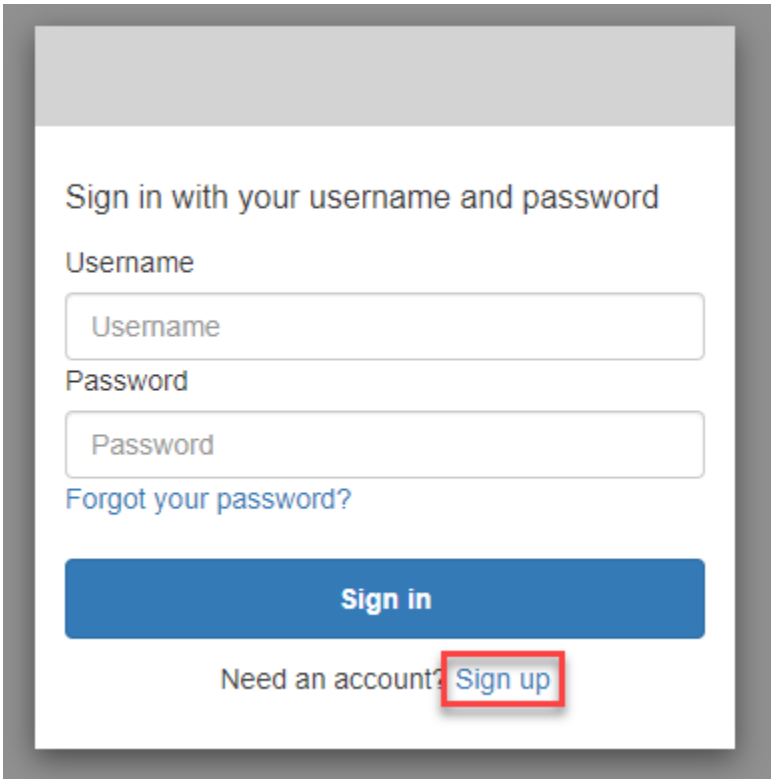
1. アプリの所有者がリストしたサードパーティーのサインインプロバイダーではなく、Amazon Cognito からユーザー名とパスワードを使用してサインインする場合は、サインインページから [Sign up] (サインアップ) を選択してください。

   サインインプロバイダーが Amazon Cognito 以外の場合は、サードパーティープロバイダーのボタンを選択するとサインアップが完了します。アプリの所有者が選択したオプションによっては、サインインに使用するプロバイダーを選択できる場合もあれば、ユーザー名とパスワードの入力を求めるメッセージのみが表示される場合もあります。

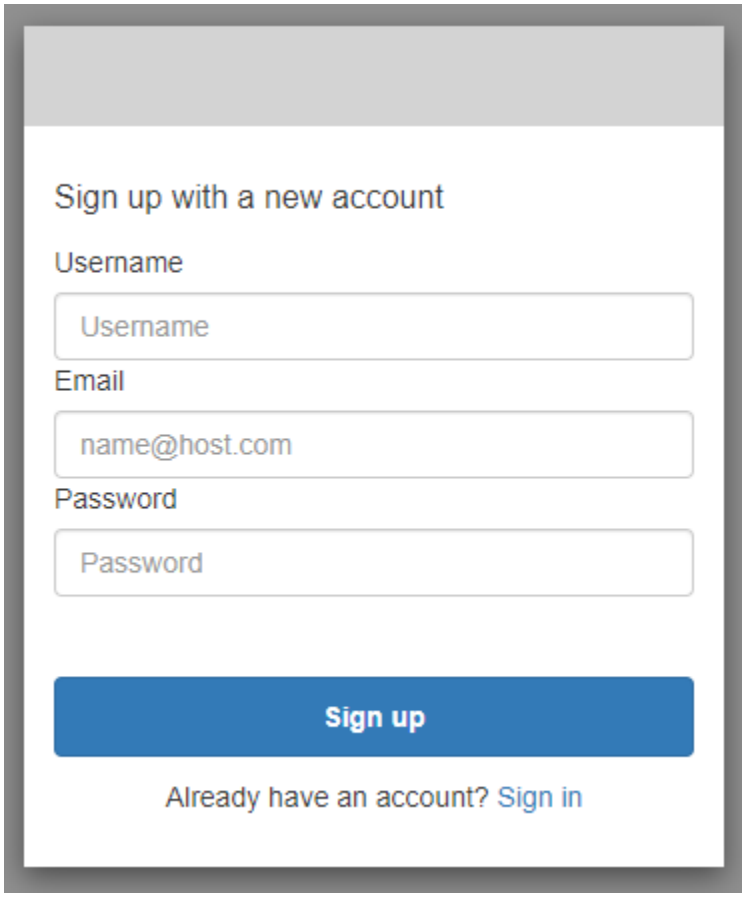
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

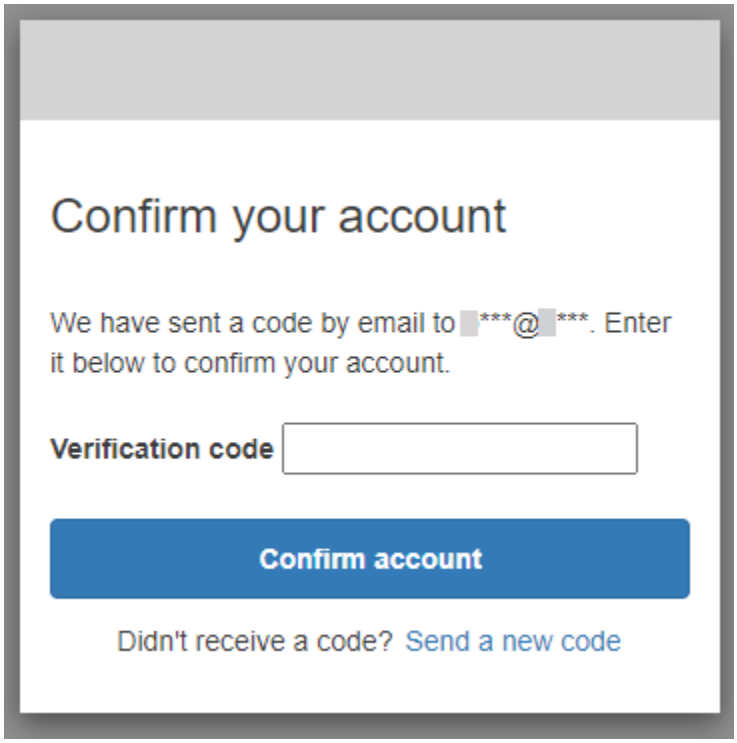


2. [Sign up with a new account] (新しいアカウントでサインアップ) ページで、アプリの所有者がサインアップに必要な情報を求めます。ユーザー名、E メールアドレス、または電話番号を尋ねる場合があります。必要な情報を入力し、パスワードを選択します。



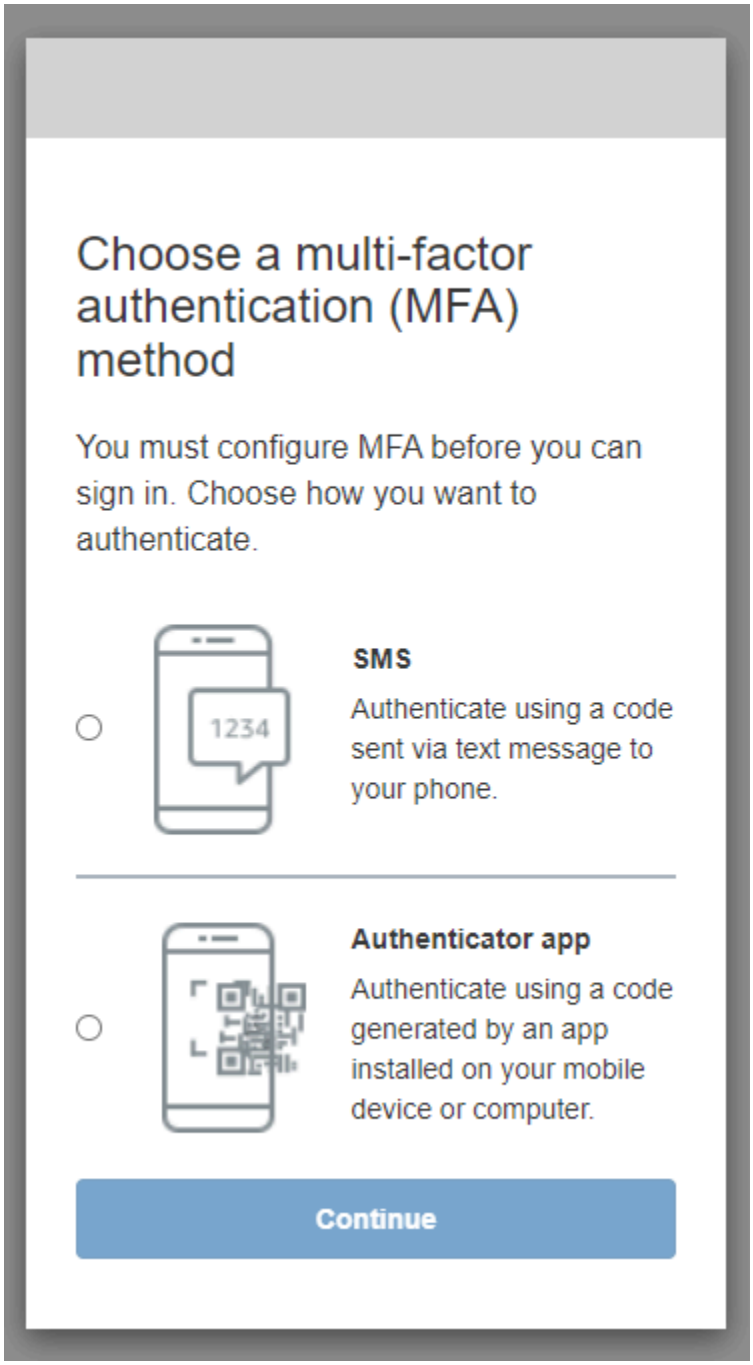
3. [Confirm your account] (アカウントを確認する) ページでは、アプリの所有者が、指定した E メールアドレスや電話番号でメッセージを受信できることを確認するために、アカウントの確認を要求する場合があります。

   E メールまたは SMS テキストメッセージでコードを受信します。フォームにコードを入力して、正しい連絡先情報を入力したことを確認します。



4. アプリの所有者は、多要素認証 (MFA) のセットアップを要求する場合があります。MFA メソッドを選択するように求めるプロンプトが表示される場合や、アプリが次のステップにスキップする場合があります。

   [Choose a multi-factor authentication (MFA) method] (多要素認証 (MFA) の方法を選択する) ページで、MFA メソッドを選択します。[SMS] を選択した場合、MFA パスコードは SMS テキストメッセージで受信されます。[Authenticator app] (認証機能付きアプリケーション) 選択した場合、時間ベースの MFA パスコードを生成するためのアプリをデバイスにインストールする必要があります。3 分以内に選択する必要があります。



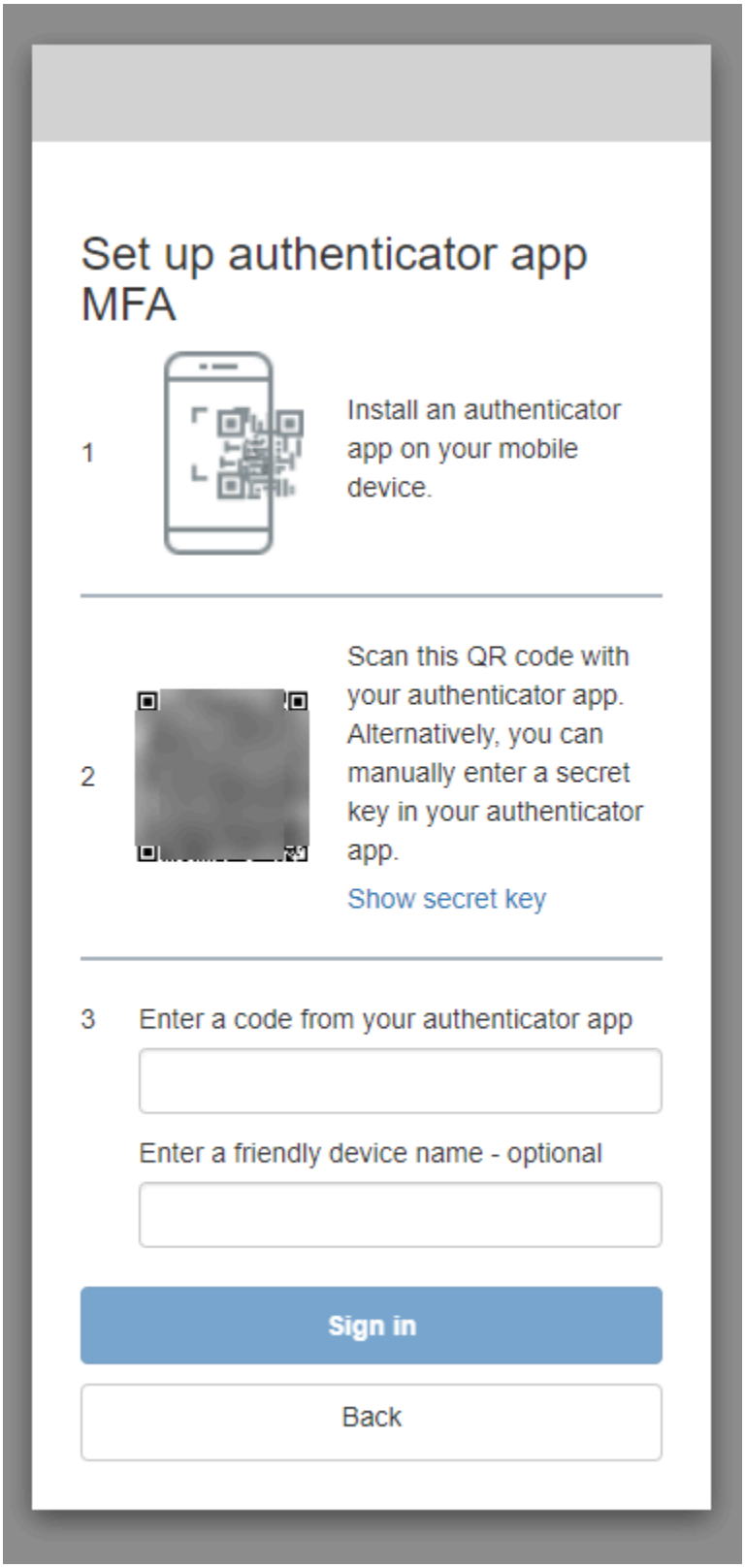
5. Amazon Cognito は、認証機能付きアプリケーションまたは SMS テキストメッセージからのコードの入力を求めます。3 分以内に受け取ったコードを入力します。

   Authenticator app

   1. ダウンロードした認証機能付きアプリケーションを開きます。
   2. ページ上の QR コードをカメラでスキャンします。カメラを使用するにはアプリの承認が必要な場合があります。

QR コードをスキャンできない場合は、[Show secret key] (シークレットキーを表示) を選択すると、認証機能付きアプリケーションに手動で入力できるコードが表示されます。

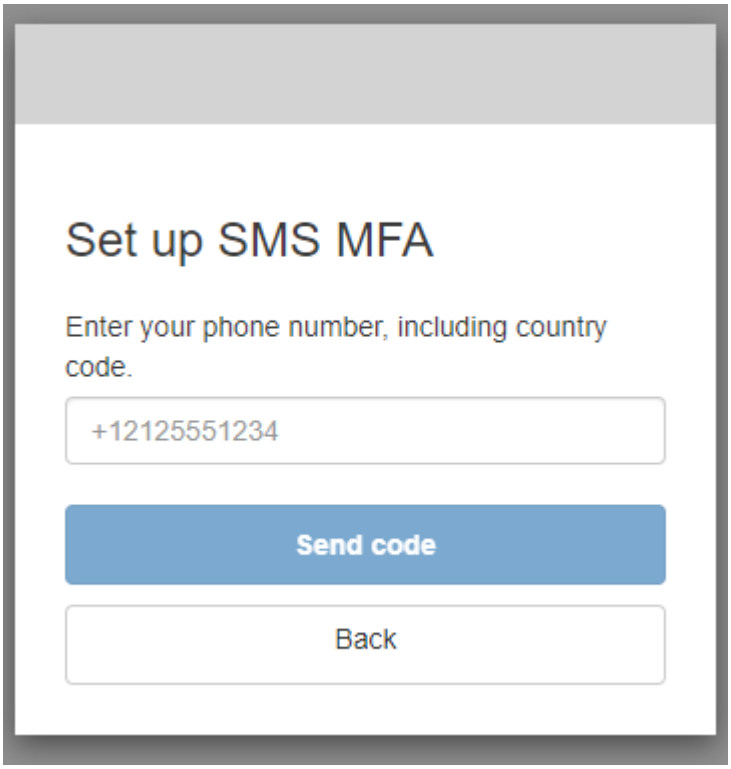
3. 認証機能付きアプリケーションは、数秒ごとに変化するコードの表示を開始します。アプリから現在のコードを入力します。
4. (オプション) [Set up authenticator app MFA] (認証機能付きアプリケーション MFA をセットアップ) ページで、デバイスの名前を選択します。サインインすると、Amazon Cognito は、ここで指定した名前のデバイスからのコードをリクエストします。


Set up authenticator app MFA
1
Install an authenticator app on your mobile device.
2
Scan this QR code with your authenticator app. Alternatively, you can manually enter a secret key in your authenticator app.
Show secret key
3
Enter a code from your authenticator app
Enter a friendly device name - optional
Sign in
Back

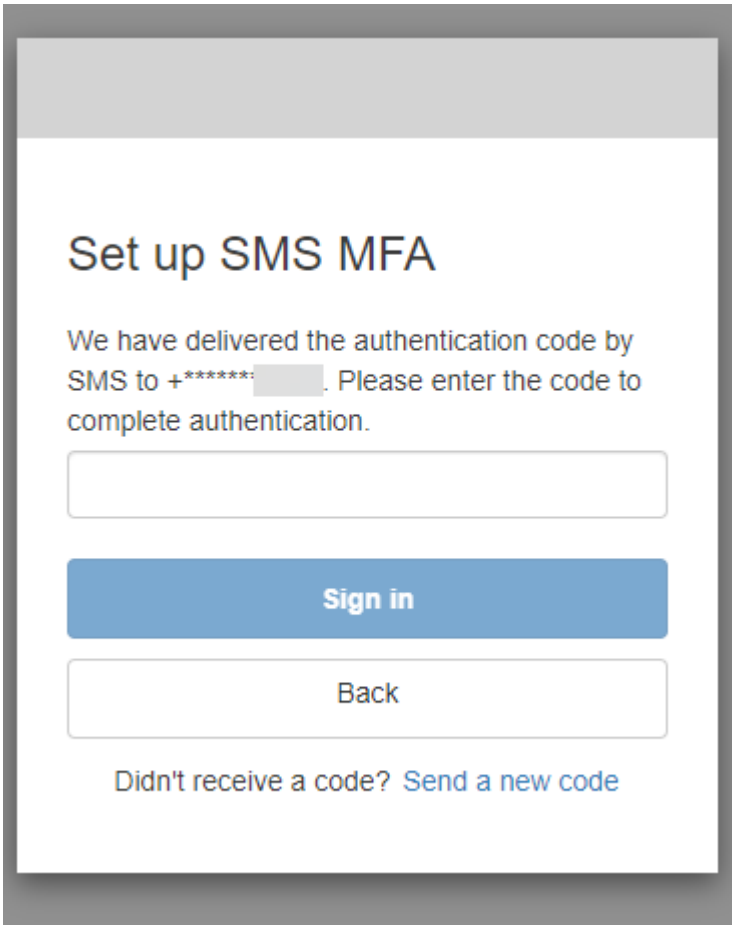
SMS text message

1. アプリ所有者が電話番号をまだ収集していない場合、Amazon Cognito は電話番号をリクエストします。

   [Set up SMS MFA] (SMS MFA のセットアップ) ページで、+ 記号と国コード (例: +12125551234) を含む電話番号を入力します。



2. コードが記載された SMS メッセージを受信します。[Set up SMS MFA] (SMS MFA のセットアップ) ページで、コードを入力します。コードが届かず、もう一度やり直したい場合は、[Send a new code] (新しいコードを送信) を選択します。[Back] (戻る) をクリックして、新しい電話番号を入力します。



6. 初回にサインアップして詳細を確認すると、このプロセスの完了後に Amazon Cognito がアプリへのアクセスを許可します。

Amazon Cognito でホストされた UI でサインインする方法

このガイドでは、Amazon Cognito を使用するアプリにサインインする方法について説明します。

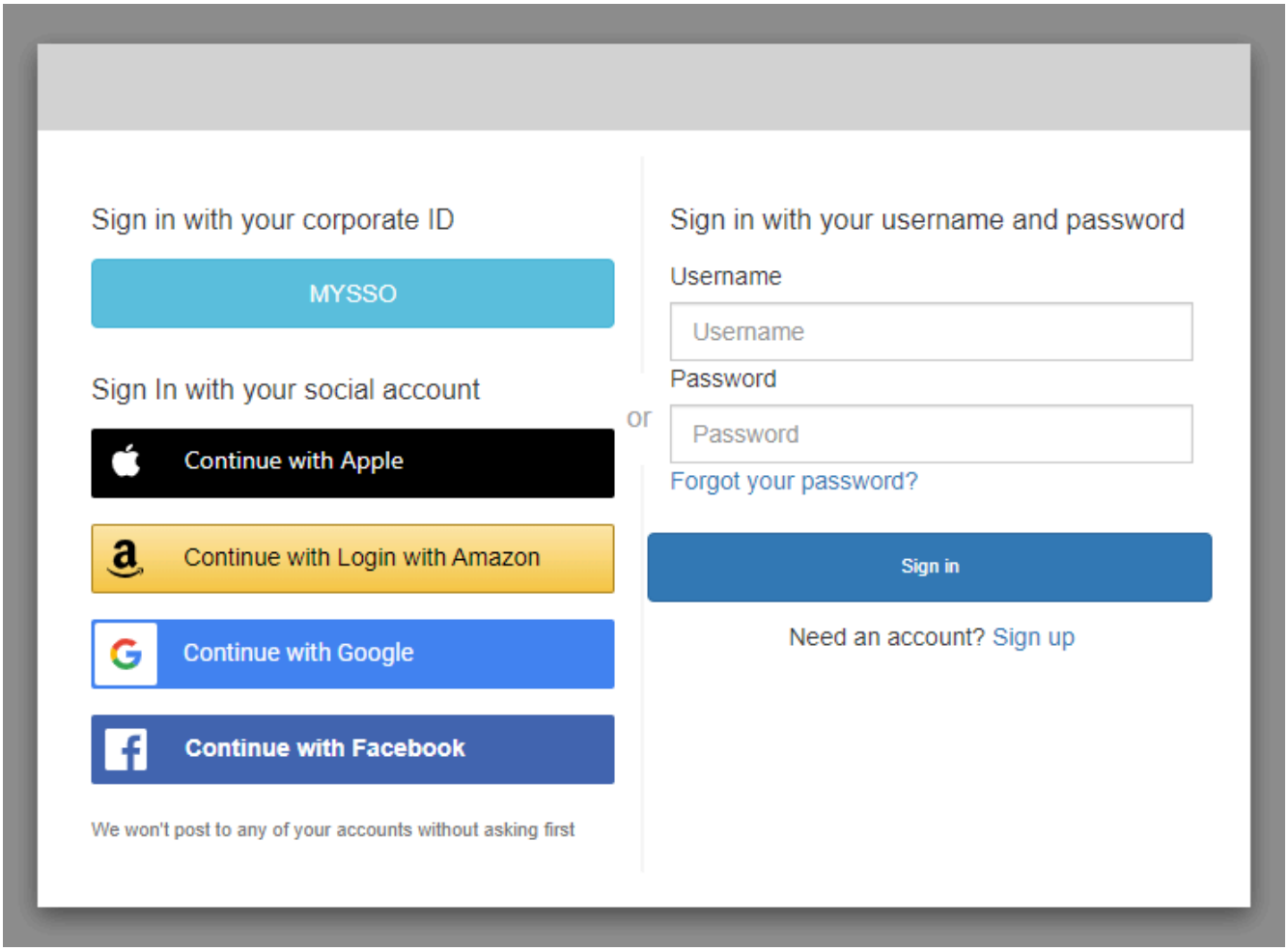
> **Note**
>
> Amazon Cognito でホストされるユーザーインターフェイス (UI) を使用するアプリにサインインすると、このガイドに示されている基本設定以外に、アプリ所有者がカスタマイズしたページが表示される場合があります。

1. アプリの所有者が選択したオプションによっては、サインインに使用するプロバイダーを選択できる場合もあれば、ユーザー名とパスワードの入力を求めるメッセージのみが表示される場合もあります。このページからユーザー名とパスワードを使用してサインインすると、Amazon

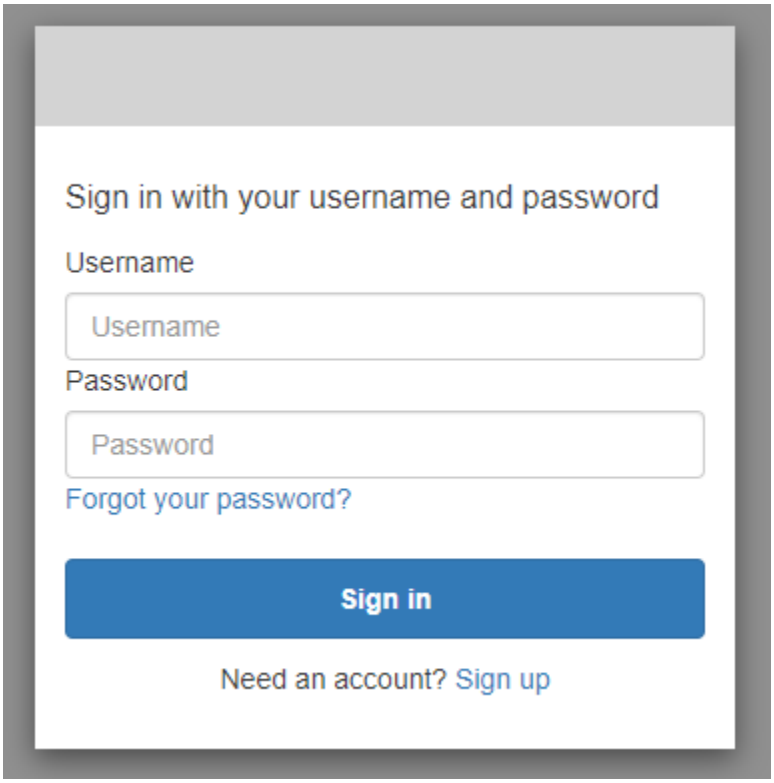
Cognito がサインインプロバイダーになります。それ以外の場合は、選択したボタンがサインインプロバイダーとして表されます。

ここでプロバイダーを選択するか、ユーザー名とパスワードを入力すると、すぐにアプリにアクセスできます。Amazon Cognito がサインインプロバイダーの場合、アプリの所有者も多要素認証を要求する可能性があります。

With multiple sign-in providers

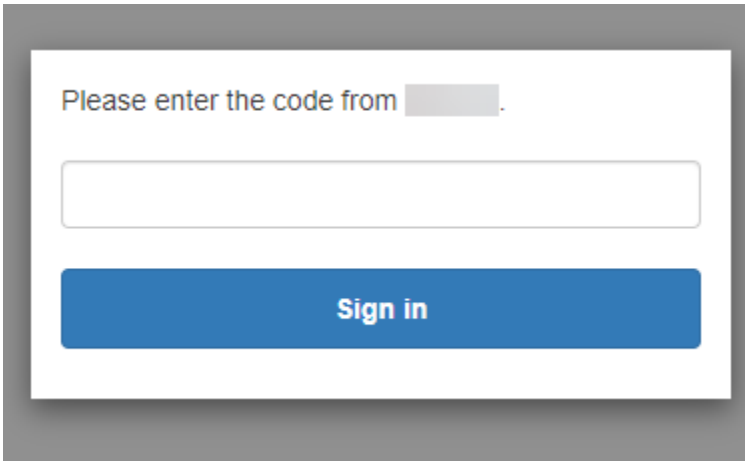


With only Amazon Cognito as a sign-in provider

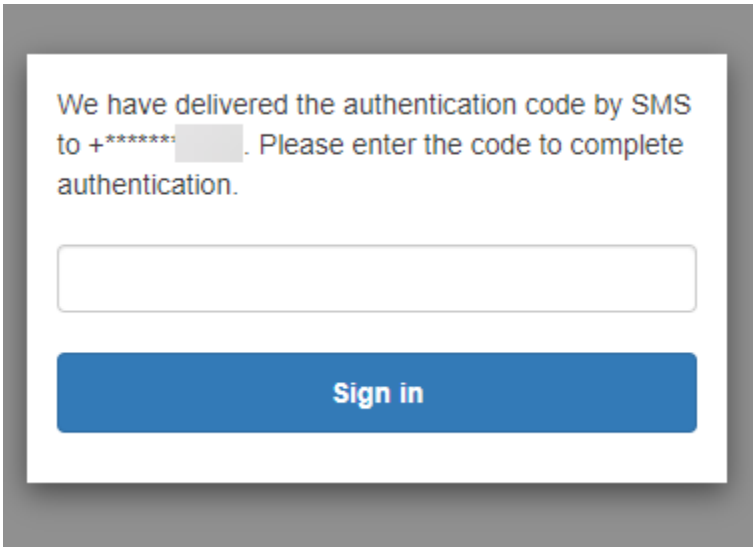


2. アプリでサインアップしたときに MFA をセットアップした可能性があります。SMS メッセージで受信した、または認証機能付きアプリケーションに表示された MFA コードを入力します。このコードは 3 分以内に入力する必要があります。

With an authenticator app



With an SMS code



3. サインインして MFA を完了すると、Amazon Cognito はアプリへのアクセスを許可します。

Amazon Cognito でホストされた UI を使用してパスワードをリセットする方法

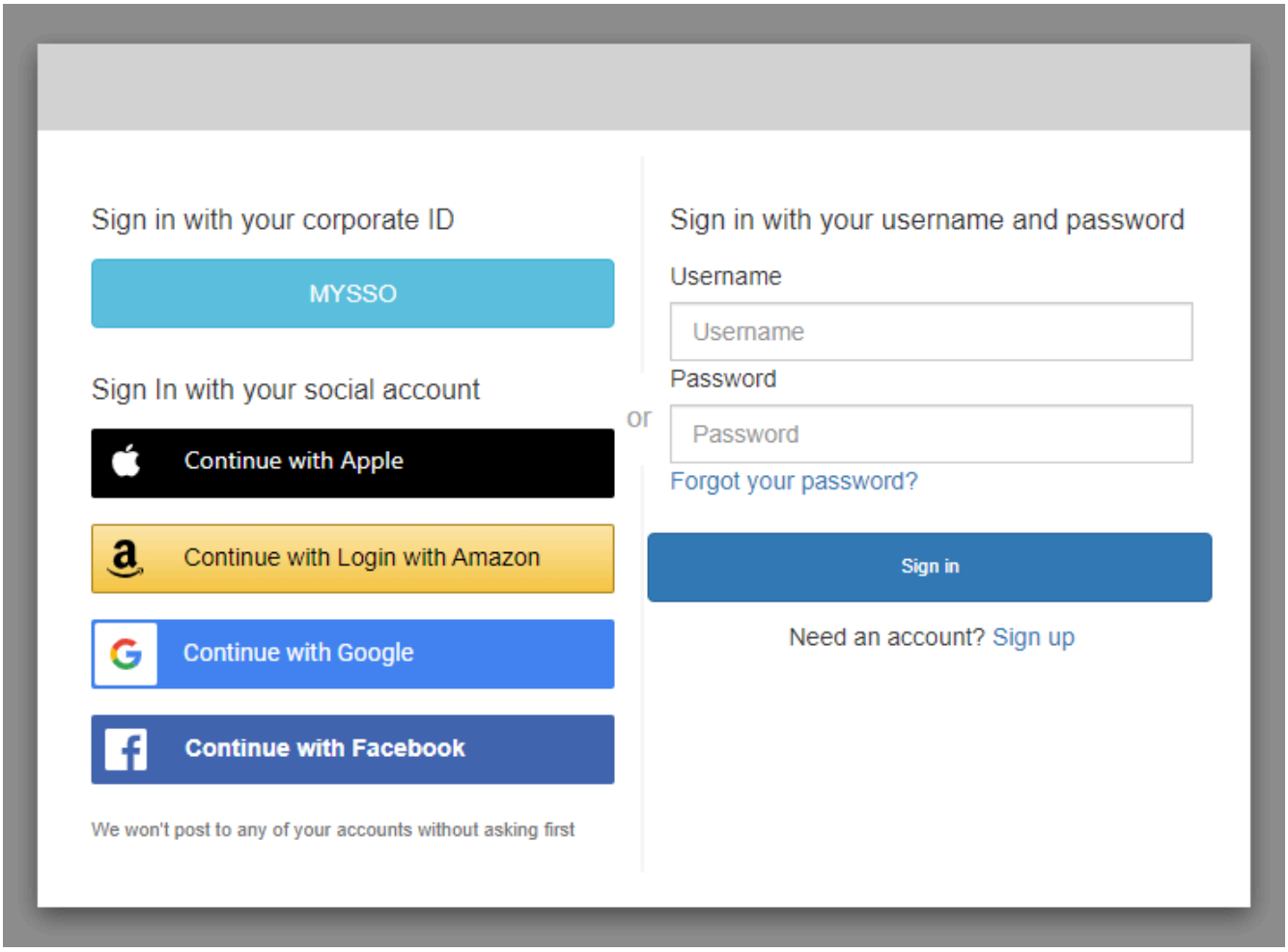
このガイドでは、Amazon Cognito を使用するアプリでパスワードをリセットする方法について説明します。

> **Note**
>
> Amazon Cognito でホストされたユーザーインターフェイス (UI) を使用するアプリにサインインすると、このガイドに示されている基本設定以外に、アプリ所有者がカスタマイズしたページが表示される場合があります。

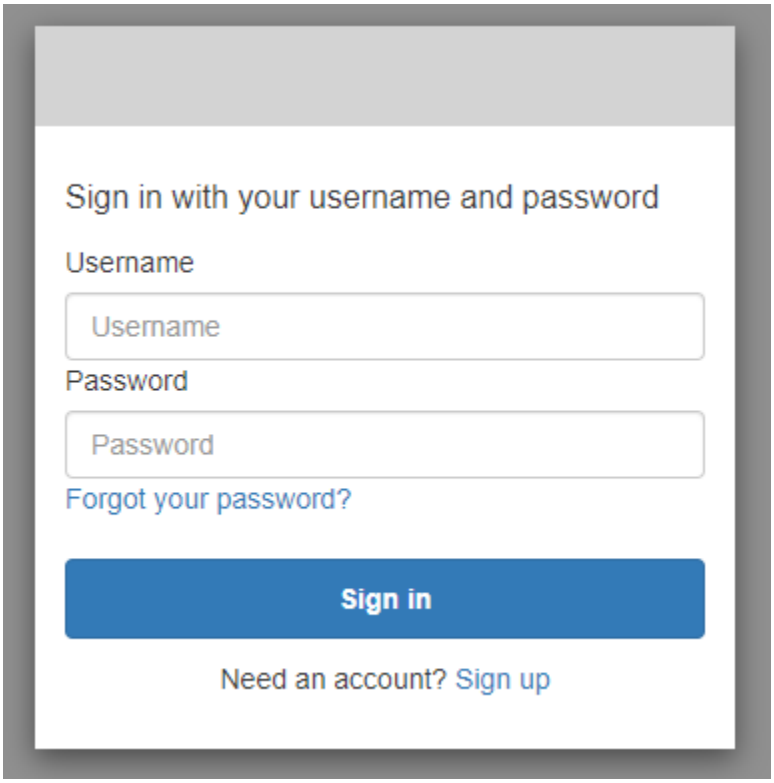
1. アプリの所有者が選択したオプションによっては、サインインに使用するプロバイダーを選択できる場合もあれば、ユーザー名とパスワードの入力を求めるメッセージのみが表示される場合もあります。このページからユーザー名とパスワードを使用してサインインすると、Amazon Cognito がサインインプロバイダーになります。それ以外の場合は、選択したボタンがサインインプロバイダーとして表されます。

   通常、サインインページからプロバイダーを選択し、パスワードが機能しない場合は、手順に従ってプロバイダーのパスワードをリセットします。Amazon Cognito がサインインプロバイダーの場合は、[Forgot your password?] (パスワードをお忘れですか。) を選択します。

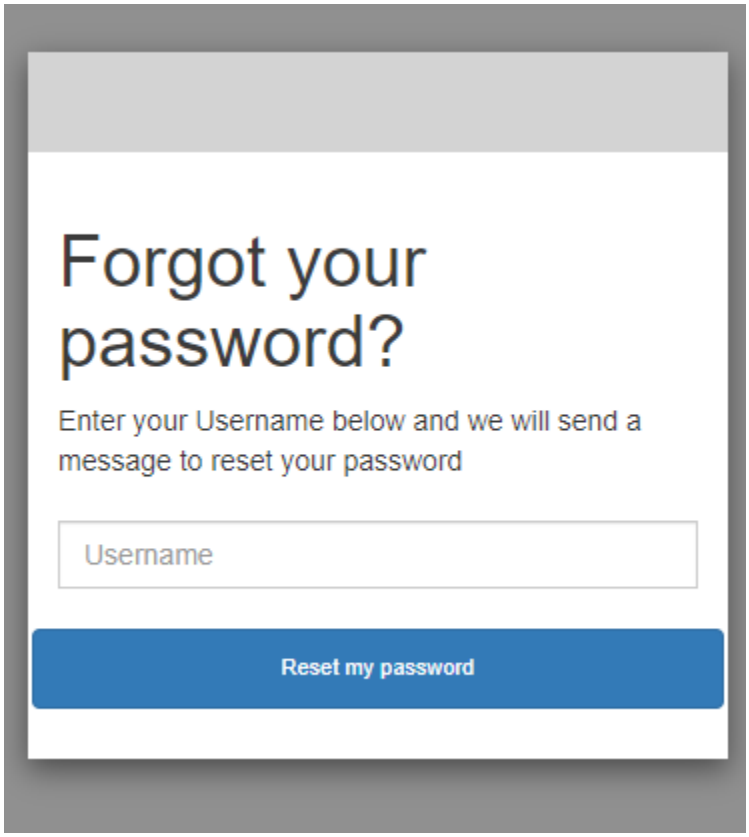
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

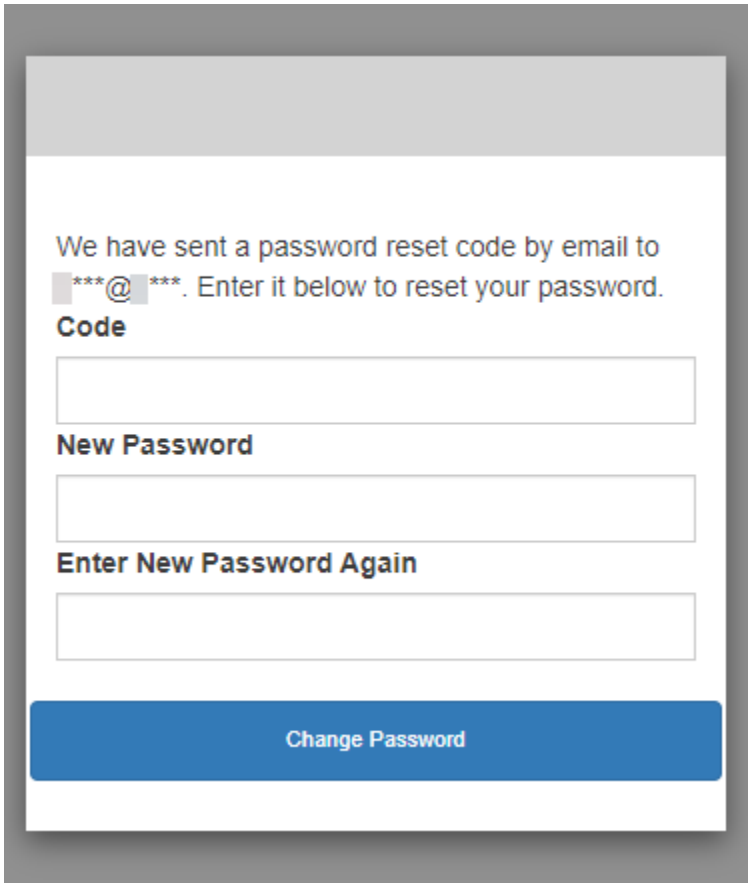


2. Amazon Cognito では、[Forgot your password?] (パスワードをお忘れですか。) ページでサインインに使用する情報の入力が求められます。ユーザー名、E メールアドレス、電話番号です。



3. Amazon Cognito は、コードを E メールメッセージまたは SMS テキストメッセージとして送信します。

   受け取ったコードを入力し、表示されたフィールドに新しいパスワードを 2 回入力します。リセットコードは 88 分以内に入力する必要があります。



4. パスワードを変更したら、サインインページに戻り、新しいパスワードでサインインします。

スコープ、M2M 、およびリソースサーバーによる API 認証

ユーザープールのドメインを設定すると、Amazon Cognito は OAuth 2.0 認可サーバーおよびホストされたウェブ UI を自動的にプロビジョニングします。このウェブ UI により、アプリケーションはユーザーにサインアップページとサインインページを表示できます。詳細については、ホストされた UI でアプリクライアントを追加するを参照してください。認証サーバーによってアクセストークンに追加するスコープを選択できます。スコープはリソースサーバーとユーザーデータへのアクセスを許可します。

リソースサーバーとは、OAuth 2.0 API サーバーのことです。リソースサーバーは、アクセスが保護されたリソースを保護するために、保護対象の API でリクエストされたメソッドとパスを承認するスコープが、ユーザープールのアクセストークンに含まれていることを検証します。トークンの署名に基づく発行者の検証、トークンの有効期限に基づく有効性の検証、トークンクレームのスコープに基づくアクセスレベルの検証を行います。ユーザープールスコープはアクセストークンscopeクレームにあります。Amazon Cognito アクセストークンのクレームの詳細については、「アクセストークンの使用」を参照してください。

Amazon Cognito を使用すると、アクセストークンのスコープは、外部 APIs またはユーザー属性へのアクセスを許可できます。ローカルユーザー、フェデレーティッドユーザー、またはマシン ID にアクセストークンを発行できます。

M achine-to-machine (M2M) 認証

Amazon Cognito は、マシン ID を使用して API データにアクセスするアプリケーションをサポートします。ユーザープール内のマシン ID は、アプリケーションサーバー上で実行され、リモート API に接続する機密クライアントです。 APIs オペレーションは、スケジュールされたタスク、データストリーム、アセットの更新など、ユーザーとのやり取りなしに行われます。これらのクライアントは、アクセストークンを使用してリクエストを承認すると、マシン間 、または M2M 認証を実行します。M2M 認証では、共有シークレットがアクセスコントロールのユーザー認証情報を置き換えます。

M2M 認証で API にアクセスするアプリケーションには、クライアント ID とクライアントシークレットが必要です。ユーザープールでは、クライアント認証情報の付与をサポートするアプリケーションクライアントを構築する必要があります。クライアント認証情報をサポートするには、アプリケーションクライアントにクライアントシークレットが必要であり、ユーザープールドメインが必要です。このフローでは、マシン ID は から直接アクセストークンをリクエストしますトークンエンドポイント。クライアント認証情報の付与には、アクセストークン内のリソースサーバーからのカスタムスコープのみを許可できます。アプリケーションクライアントの設定の詳細については、「」を参照してくださいユーザープールアプリクライアント。

クライアント認証情報付与からのアクセストークンは、マシン ID が API からリクエストすることを許可するオペレーションの検証可能なステートメントです。アクセストークンが API リクエストを承認する方法の詳細については、「」を参照してください。アプリケーションの例については、Amazon Cognito および API Gateway ベースのマシン間認証 CDK を使用した AWS マシン間認証」を参照してください。

M2M 認証には、毎月のアクティブユーザー (MAUs請求する方法とは異なる請求モデルがあります。ユーザー認証にアクティブなユーザーあたりのコストがかかる場合、M2M 請求にはアクティブなクライアント認証情報アプリケーションクライアントとトークンリクエストの合計ボリュームが反映されます。詳細については、「Amazon Cognito の料金」を参照してください。M2M 認証のコストを制御するには、アクセストークンの期間とアプリケーションが行うトークンリクエストの数を最適化します。API Gateway キャッシュを使用して M2M 認証の新しいトークンのリクエストを減らす方法については、キャッシュトークン「」を参照してください。

AWS 請求書にコストを追加する Amazon Cognito オペレーションの最適化については、「」を参照してください のコスト管理。

スコープについて

スコープはアプリがリソースにリクエストできるアクセスのレベルです。Amazon Cognito アクセストークンのスコープは、ユーザープールで設定した信頼 (既知のデジタル署名を持つアクセストークンの信頼できる発行者) によってバックアップされます。ユーザープールが生成できるアクセストークンのスコープでは、顧客が自分のユーザープロファイルの一部または全部の管理や、バックエンド API からのデータの取得を許可されていることを証明します。Amazon Cognito ユーザープールが発行するアクセストークンのスコープには、ユーザープールのリザーブド API スコープ、カスタムスコープ、および標準スコープがあります。

ユーザープールのリザーブド API スコープ

`aws.cognito.signin.user.admin` スコープは Amazon Cognito ユーザープール API を承認します。これにより、アクセストークンのベアラーが、 および UpdateUserAttributes API オペレーションなどを使用して、ユーザープールユーザーに関するすべての情報をクエリGetUserおよび更新することを許可されます。Amazon Cognito ユーザープール API でユーザーを認証する場合は、このスコープだけをアクセストークンで受け取ります。また、アプリケーションクライアントにユーザー属性の読み取りと書き込みを許可している場合、このユーザー属性の読み取りと書き込みに必要な唯一のスコープでもあります。このスコープは、認可エンドポイント へのリクエストでリクエストすることもできます。このスコープだけでは、UserInfo エンドポイント にユーザー属性をリクエストするには不十分です。ユーザープール API とユーザーへの `userInfo` リクエストの両方を承認するアクセストークンについては、`/oauth2/authorize` リクエストで `openid` スコープと `aws.cognito.signin.user.admin` スコープの両方をリクエストする必要があります。

カスタムスコープ

カスタムスコープは、リソースサーバーで保護する外部 API へのリクエストを承認します。カスタムスコープを他の種類のスコープと一緒にリクエストできます。カスタムスコープの詳細については、このページ全体を通じて説明しています。

標準スコープ

ユーザープール OAuth 2.0 認証サーバー (ホストされた UI を含む) でユーザーを認証する場合は、スコープをリクエストする必要があります。ユーザープールのローカルユーザーとサードパーティのフェデレーションユーザーは、Amazon Cognito 認証サーバーで認証できます。標準 OAuth 2.0 スコープは、ユーザープールの UserInfo エンドポイント からユーザー情報を読み取ることをアプリケーションに許可します。`userInfo` エンドポイントからユーザー属性をクエリする OAuth モデルでは、ユーザー属性に対する大量のリクエストに合わせてアプリケーションを最適化できます。`userInfo` エンドポイントは、アクセストークンのスコープによって決まるアクセス許可レベ

ルで属性を返します。以下の標準 OAuth 2.0 スコープでアクセストークンを発行することをアプリクライアントに許可できます。

openid

OpenID Connect (OIDC) クエリの最小スコープ。ID トークン、一意の識別子のクレーム sub、および他のスコープをリクエストする機能を承認します。

> **Note**
>
> openid スコープをリクエストし、それ以外をリクエストしない場合、ユーザープール ID トークンと userInfo レスポンスには、アプリケーションクライアントが読み取ることができるすべてのユーザー属性のクレームが含まれます。openid や他の標準スコープ (profile、email、phone など) をリクエストすると、ID トークンと userInfo レスポンスの内容は追加のスコープの制約に制限されます。
> 例えば、認可エンドポイント へのリクエストでパラメータ scope=openid+email を使用すると、sub、email、email_verified を含む ID トークンが返されます。このリクエストのアクセストークンは、UserInfo エンドポイント から同じ属性を返します。リクエストでパラメータ scope=openid を使用すると、ID トークンと userInfo 内のクライアントが読み取り可能なすべての属性が返されます。

profile

アプリクライアントが読み取ることができるすべてのユーザー属性を承認します。

email

ユーザー属性 email および email_verified を承認します。Amazon Cognito は、値が明示的に設定されている場合、email_verified を返します。

phone

ユーザー属性 phone_number および phone_number_verified を承認します。

リソースサーバーについて

リソースサーバー API は、データベース内の情報へのアクセスを許可したり、IT リソースを制御したりすることができます。Amazon Cognito のアクセストークンは、OAuth 2.0 をサポートする API へのアクセスを許可できます。Amazon API Gateway REST API には、Amazon Cognito のアクセストークンによる認証の組み込みサポートがあります。アプリは、API コール内でアクセストークンを

リソースサーバーに渡します。リソースサーバーはアクセストークンを検査し、アクセスを付与するかどうかを決定します。

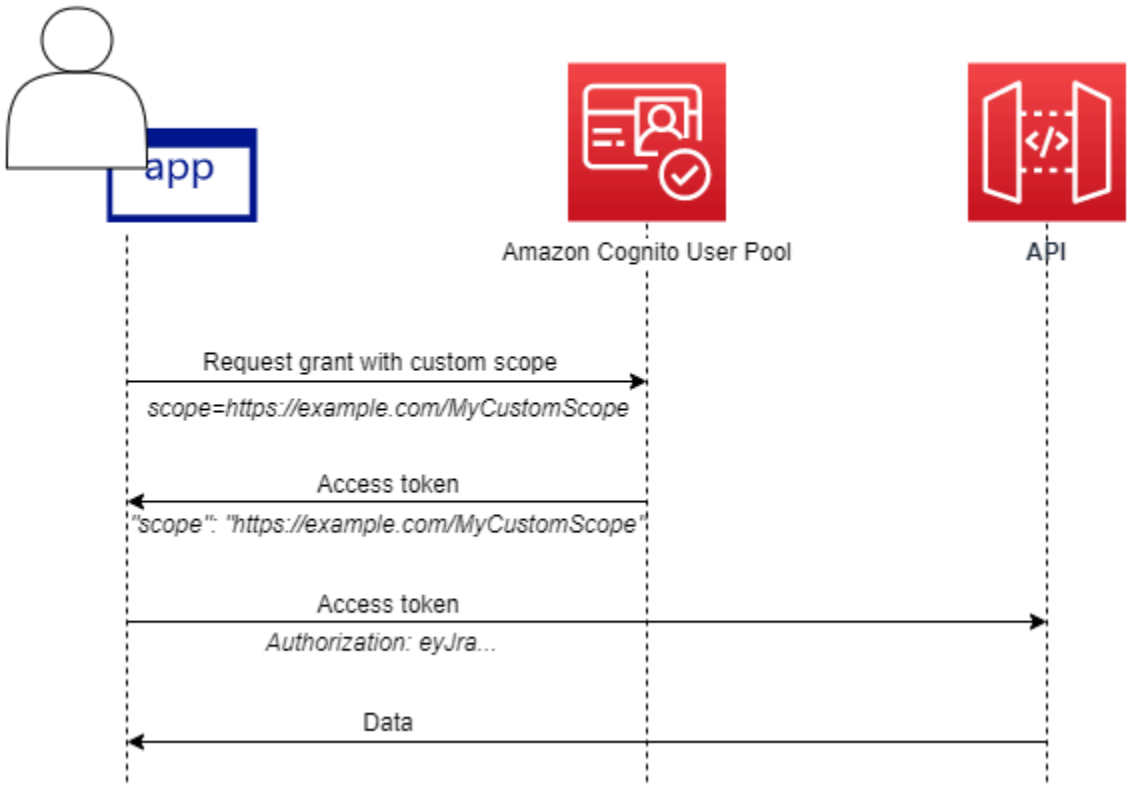
Amazon Cognito は、ユーザープールのアクセストークンのスキーマを将来更新する可能性があります。アプリケーションがアクセストークンを API に渡す前に、アクセストークンの内容を分析する場合は、スキーマの更新を受け入れるようにコードを設計する必要があります。

カスタムスコープはユーザーが定義するスコープであり、ユーザープールの承認機能を拡張して、ユーザーとユーザー属性のクエリや変更とは関係のない目的にも対応できるようにします。例えば、写真用のサーバーリソースがある場合、2 つのスコープを定義することが考えられます。写真への読み取りアクセス用の `photos.read` と、書き込み/削除アクセス用の `photos.write` です。アクセストークンを承認して許可するように API を設定し、`scope` クレームでの `photos.read` によるアクセストークンへの `HTTP GET` リクエスト、および `photos.write` によるトークンへの `HTTP POST`リクエストを許可できます。これらはカスタムスコープです。

> **Note**
>
> リソースサーバーはトークン内のクレームを処理する前に、アクセストークンの署名と有効期限を検証する必要があります。トークンの検証の詳細については、「JSON Web トークンの検証」を参照してください。Amazon API Gateway でユーザープールのトークンを検証および使用する方法の詳細については、ブログ「Amazon Cognito ユーザープールと API Gateway の統合」を参照してください。API Gateway は、アクセストークンの検証とリソースの保護に適したオプションです。API Gateway カスタムオーソライザーの詳細については、「API Gateway Lambda オーソライザーを使用する」を参照してください。

概要

Amazon Cognito を使用すると、OAuth2.0 リソースサーバーを作成し、これらにカスタムスコープを関連付けることができます。アクセストークンのカスタムスコープは、API の特定のアクションを許可します。ユーザープール内のどのアプリケーションクライアントに対しても、任意のリソースサーバーからカスタムスコープを発行することを許可できます。カスタムスコープをアプリケーションクライアントに関連付け、これらのスコープを トークンエンドポイント から OAuth2.0 認証コード付与、暗黙的な付与、クライアント認証情報付与でリクエストできます。Amazon Cognito は、アクセストークンの `scope` クレームにカスタムスコープを追加します。クライアントはサーバーリソースに対してアクセストークンを使用して、トークンに存在するスコープに基づいて承認を判断できます。アクセストークンスコープの詳細については、「ユーザープールのトークンの使用」を参照してください。



カスタムスコープでアクセストークンを取得するには、アプリは認証コードを引き換えるか、クライアント認証情報の付与をリクエストするために、トークンエンドポイント にリクエストを行う必要があります。ホストされた UI では、暗黙的付与からアクセストークンのカスタムスコープをリクエストすることもできます。

> **Note**
>
> これらは IdP としてユーザープールとのヒューマンインタラクティブ認証用に設計されているため、 InitiateAuthAdminInitiateAuthリクエストは単一の値 を持つアクセストークンにのみscopeクレームを生成しますaws.cognito.signin.user.admin。

リソースサーバーおよびカスタムスコープの管理

リソースを作成するときに、リソースサーバー名とリソースサーバー識別子を指定する必要があります。リソースサーバーに作成したスコープごとに、スコープ名と説明を指定する必要があります。

- リソースサーバー名: リソースサーバーのわかりやすい名前 (`Solar system object tracker` や `Photo API` など)。

- リソースサーバーの識別子: リソースサーバーの一意の識別子。識別子は、API に関連付ける任意の名前 (`solar-system-data` など) です。API URI パスへの直接参照のように長い識別子 (`https://solar-system-data-api.example.com` など) を設定できますが、文字列が長くなるとアクセストークンのサイズが大きくなります。
- スコープ名: `scope` クレームに必要な値。例えば `sunproximity.read` です。
- 説明: スコープのわかりやすい説明。例えば `Check current proximity to sun` です。

Amazon Cognito では、ユーザープールのローカルユーザーであるか、サードパーティ ID プロバイダーのフェデレーションユーザーであるかに関係なく、すべてのユーザーのアクセストークンにカスタムスコープを含めることができます。ホストされた UI を含む OAuth 2.0 認証サーバーでは、認証フロー中にユーザーのアクセストークンのスコープを選択できます。ユーザーの認証は、リクエストパラメータの 1 つとして `scope` を使用して 認可エンドポイント で開始する必要があります。リソースサーバーには次の形式が推奨されます。識別子として、API のわかりやすい名前を使用します。カスタムスコープには、リソースサーバーが許可するアクションを使用します。

```
resourceServerIdentifier/scopeName
```

例えば、カイパーベルトで新しい小惑星を発見し、それを `solar-system-data` API を通じて登録するとします。小惑星のデータベースへの書き込み操作を許可するスコープは `asteroids.add` です。発見の登録を許可するアクセストークンをリクエストするときは、`scope` HTTPS リクエストパラメータを `scope=solar-system-data/asteroids.add` としてフォーマットします。

スコープをリソースサーバーから削除しても、すべのクライアントとの関連付けは削除されません。代わりに、スコープが inactive とマークされます。Amazon Cognito はアクセストークンに非アクティブなスコープを追加しませんが、それ以外はアプリからのスコープのリクエストを通常どおりに処理します。後でスコープをリソースサーバーに再度追加すると、Amazon Cognito はそのスコープを再度アクセストークンに書き込みます。アプリケーションクライアントに関連付けていないスコープをリクエストすると、ユーザープールのリソースサーバーからスコープを削除したかどうかに関係なく、認証は失敗します。

AWS Management Console、API、または CLI を使用して、ユーザープールのリソースサーバーとスコープを定義できます。

ユーザープールのリソースサーバーを定義する (AWS Management Console)

を使用して AWS Management Console 、ユーザープールのリソースサーバーを定義できます。

リソースサーバーを定義する

1. Amazon Cognito コンソールにサインインします。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [App integration] (アプリケーションの統合) タブを開き、検索する [Resource servers] (リソースサーバー) を検索します。
4. [Create a resource server] (リソースサーバーの作成) を選択します。
5. [Resource server name] (リソースサーバー名) を入力します。例えば `Photo Server` です。
6. [Resource server identifier] (リソースサーバー識別子) を入力します。例えば `com.example.photos` です。
7. リソースの [Custom scopes] (カスタムスコープ) (例: `read` および `write`) を入力します。
8. [Scope name] (スコープ名) ごとに [Description] (説明) (例: `view your photos` および `update your photos`) を入力します。
9. [作成] を選択します。

カスタムスコープは、[Custom scopes] (カスタムスコープ) 列にある、[Resource servers] (リソースサーバー) の [App integration] (アプリケーションの統合) タブで確認できます。カスタムスコープは、アプリケーションクライアントに対して [App integration] (アプリケーションクライアント) タブの下にある [App integration] (アプリケーションの統合) で有効にできます。アプリケーションクライアントを選択し、[Hosted UI settings] (ホストされた UI の設定) を検索して [Edit] (編集) を選択します。[Custom scopes] (カスタムスコープ) を追加し、[Save changes] (変更の保存) を選択します。

ユーザープール (AWS CLI および AWS API) のリソースサーバーの定義

次のコマンドを使用して、ユーザープールのリソースサーバー設定を指定します。

リソースサーバーを作成する

- AWS CLI: `aws cognito-idp create-resource-server`
- AWS API: CreateResourceServer

リソースサーバーの設定に関する情報を取得する

- AWS CLI: `aws cognito-idp describe-resource-server`
- AWS API: DescribeResourceServer

ユーザープールのすべてのリソースサーバーに関する情報を一覧表示する

- AWS CLI: `aws cognito-idp list-resource-servers`
- AWS API: ListResourceServers

リソースサーバーを削除する

- AWS CLI: `aws cognito-idp delete-resource-server`
- AWS API: DeleteResourceServer

リソースサーバーの設定を更新する

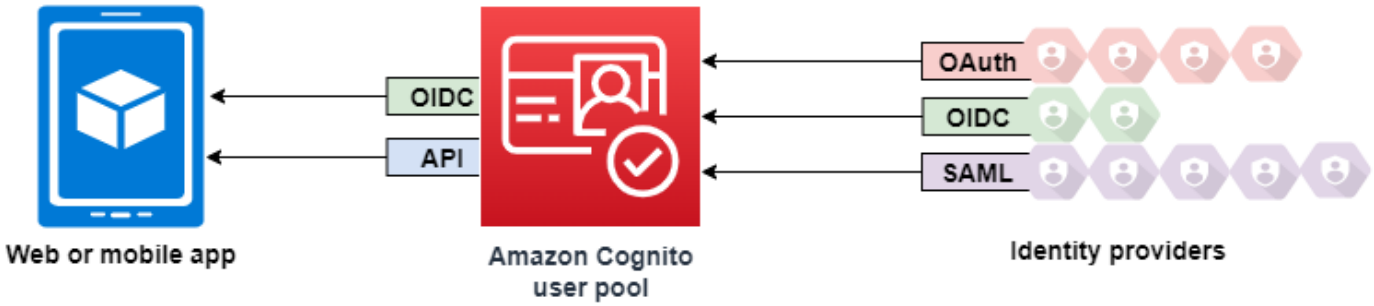
- AWS CLI: `aws cognito-idp update-resource-server`
- AWS API: UpdateResourceServer

サードパーティー経由のユーザープールへのサインインの追加

アプリユーザーは、ユーザープールから直接サインインすることも、サードパーティー ID プロバイダー (IdP) を介してフェデレーションすることもできます。ユーザープールは、Facebook、Google、Amazon、Apple を介したソーシャルサインイン、および OpenID Connect (OIDC) と SAML から返されるトークンの処理のオーバーヘッドを管理します IdPs。組み込みのホストウェブ UI により、Amazon Cognito はすべての の認証済みユーザーのトークン処理と管理を提供します IdPs。このように、バックエンドシステムは 1 セットのユーザープールトークンで標準化できます。

Amazon Cognito ユーザープールでのフェデレーションサインインの仕組み

サードパーティー (フェデレーション) 経由のサインインは、Amazon Cognito のユーザープールで使用できます。この機能は、Amazon Cognito ID プール (フェデレーティッド ID) 経由のフェデレーションとは無関係です。



Amazon Cognito はユーザーディレクトリであり、OAuth 2.0 ID プロバイダー (IdP) です。Amazon Cognito ディレクトリにローカルユーザーをサインインさせると、ユーザープールがアプリの IdP になります。ローカルユーザーは、外部 IdP を介したフェデレーションなしに、ユーザープールディレクトリにのみ存在します。

Amazon Cognito をソーシャル、SAML、または OpenID Connect (OIDC) に接続すると IdPs、ユーザープールは複数のサービスプロバイダーとアプリ間のブリッジとして機能します。IdP にとって、Amazon Cognito はサービスプロバイダー (SP) です。 IdPs は OIDC ID トークンまたは SAML アサーションを Amazon Cognito に渡します。Amazon Cognito は、トークンまたはアサーション内のユーザーに関するクレームを読み取り、それらのクレームをユーザープールディレクトリ内の新しいユーザープロファイルにマッピングします。

Amazon Cognito は、フェデレーションユーザーのユーザープロファイルを独自のディレクトリに作成します。Amazon Cognito は、IdP からのクレームと、OIDC およびソーシャル ID プロバイダーの場合は、IdP によるパブリック `userinfo` エンドポイントに基づいて、ユーザーに属性を追加します。マッピングされた IdP 属性が変更されると、ユーザーの属性はユーザープール内で変化します。さらに、IdP の属性から独立した属性を追加することもできます。

Amazon Cognito がフェデレーションユーザーのプロファイルを作成すると、その機能が変更され、SP となったアプリに IdP として表示されます。Amazon Cognito は OIDC と OAuth 2.0 IdP の組み合わせです。アクセストークン、ID トークン、およびリフレッシュトークンを生成します。トークンの詳細については、ユーザープールでのトークンの使用 を参照してください。

フェデレーションまたはローカルを問わず、ユーザーを認証して承認するために、Amazon Cognito と統合するアプリを設計する必要があります。

Amazon Cognito のサービスプロバイダーとしてのアプリの責任

トークンの情報を検証して処理する

ほとんどのシナリオでは、Amazon Cognito は認証されたユーザーを、認証コードを追加したアプリの URL にリダイレクトします。アプリは、アクセス、ID、およびリフレッシュの各トークンとこのコードを交換します。そして、トークンの有効性を確認し、トークン内のクレームに基づいて、ユーザーに情報を提供する必要があります。

Amazon Cognito API リクエストによる認証イベントへの応答

アプリは、Amazon Cognito ユーザープール API と認証 API エンドポイントと統合する必要があります。認証 API は、ユーザーのサインインとサインアウトを行い、トークンを管理します。ユーザープール API には、ユーザープール、ユーザー、および認証環境のセキュリティを管理するさまざまなオペレーションがあります。アプリは。Amazon Cognito からの応答を受信したときに次に何をすべきかを知っている必要があります。

Amazon Cognito ユーザープールサードパーティーサインインについて知っておくべきこと

- フェデレーションプロバイダーでユーザーにサインインさせたい場合は、ドメインを選択する必要があります。これにより、Amazon Cognito でホストされた UI と ホストされた UI および OIDC エンドポイント が設定されます。詳細については、「ホストされた UI への独自のドメインの使用」を参照してください。
- InitiateAuth や などの API オペレーションを使用してフェデレーティッドユーザーにサインインすることはできませんAdminInitiateAuth。フェデレーションユーザーは、ログインエンドポイント または 認可エンドポイント でのみサインインできます。
- 認可エンドポイント はリダイレクションエンドポイントです。リクエストに `idp_identifier` または `identity_provider` パラメータを指定すると、ホストされている UI をバイパスして、表示なしで IdP にリダイレクトされます。それ以外の場合は、ホストされている UI ログインエンドポイント にリダイレクトされます。例については、シナリオ例: エンタープライズダッシュボードで Amazon Cognito アプリケーションをブックマークするを参照してください。
- ホストされた UI がセッションをフェデレーション IdP にリダイレクトすると、Amazon Cognito は、リクエストに `user-agent`ヘッダー`Amazon/Cognito` を含めます。
- Amazon Cognito は、フェデレーションユーザープロファイルの `username` 属性を、固定識別子と IdP の名前の組み合わせから派生します。カスタム要件に一致するユーザー名を生成するに

は、preferred_username 属性へのマッピングを作成します。詳細については、「マッピングについて知っておくべきこと」を参照してください。

例: MyIDP_bob@example.com

- Amazon Cognito は、フェデレーションユーザーの ID に関する情報を属性および identities と呼ばれる ID トークン内のクレームに記録します。このクレームには、ユーザーのプロバイダーと、プロバイダーからの一意の ID が含まれます。ユーザープロファイル内の identities 属性を直接変更することはできません。フェデレーションユーザーをリンクする方法の詳細については、「フェデレーションユーザーを既存のユーザープロファイルにリンクする」を参照してください。
- UpdateIdentityProvider API リクエストで IdP を更新する場合、変更がホスト UI に反映されるまでに最長で 1 分かかることがあります。
- Amazon Cognito は、自身と IdP との間で最大 20 の HTTP リダイレクトをサポートします。
- ユーザーがホストされた UI でサインインすると、ブラウザは暗号化されたログインセッション Cookie を保存し、サインインに使用したクライアントとプロバイダーを記録します。同じパラメータを使用して再度サインインしようとすると、ホストされた UI は有効期限が切れていない既存のセッションをすべて再利用し、ユーザーが認証情報を再入力せずに認証を行います。ローカルユーザープールログインへの切り替えやローカルユーザープールログインからの切り替えなど、ユーザーが別の IdP で再度ログインする場合は、認証情報を入力して新しいログインセッションを生成する必要があります。

  任意のユーザープール IdPs を任意のアプリクライアントに割り当てることができ、ユーザーはアプリクライアントに割り当てた IdP でのみサインインできます。

トピック

- ユーザープールの ID プロバイダーの設定
- ユーザープールでのソーシャル ID プロバイダーの使用
- ユーザープールでの SAML ID プロバイダーの使用
- ユーザープールでの OIDC ID プロバイダーの使用
- ユーザープールの ID プロバイダー属性マッピングを指定する
- フェデレーションユーザーを既存のユーザープロファイルにリンクする

ユーザープールの ID プロバイダーの設定

フェデレーティッド ID プロバイダーのサインインの下のサインインエクスペリエンスタブで、ユーザープールに ID プロバイダー (IdPs) を追加できます。 詳細については、「サードパーティー経由のユーザープールへのサインインの追加」を参照してください。

トピック

- ソーシャル IdP でユーザーサインインを設定する
- OIDC IdP でユーザーサインインを設定する
- SAML IdP でユーザーサインインを設定する

ソーシャル IdP でユーザーサインインを設定する

フェデレーションを使用して、Amazon Cognito のユーザープールを Facebook、Google、Login with Amazon などのソーシャル ID プロバイダーと統合することができます。

ソーシャル ID プロバイダーを追加するには、最初に ID プロバイダーでデベロッパーアカウントを作成します。デベロッパーアカウントが作成されたら、アプリを ID プロバイダーに登録します。ID プロバイダーがアプリ用のアプリ ID とアプリシークレットを作成するので、これらの値を Amazon Cognito ユーザープールで設定します。

- Google Identity Platform
- Facebook for Developers
- Login with Amazon
- Apple でサインイン

ユーザーサインインとソーシャル IdP を統合するには

1. Amazon Cognito コンソールにサインインします。プロンプトが表示されたら、AWS 認証情報を入力します。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [Sign-in experience] (サインインエクスペリエンス) タブを選択し、[Federated sign-in] (フェデレーションサインイン) を検索します。
4. [Add an identity provider] (ID プロバイダーを追加する) を選択、または、設定した [Facebook]、[Google]、[Amazon]、または [Apple] などを選択し、[Identity provider information]

(ID プロバイダー情報) を検索し、[Edit] (編集) をクリックします。ソーシャル ID プロバイダーの追加の詳細については、「ユーザープールでのソーシャル ID プロバイダーの使用」を参照してください。

5. 選択した IdP に基づいて、次のいずれかの手順を実行して、ソーシャル ID プロバイダーの情報を入力します。

   Facebook、Google、および Login with Amazon

   クライアントアプリを作成したときに受け取ったアプリ ID とアプリシークレットを入力します。

   Apple でサインイン

   Apple に提供したサービス ID、およびアプリケーションクライアントを作成したときに受け取ったチーム ID、キー ID、プライベートキーを入力します。

6. [Authorize scopes] (承認スコープ) に、ユーザープール属性にマップするソーシャル ID プロバイダースコープの名前を入力します。スコープは、アプリケーションでアクセスするユーザー属性 (名前や E メールなど) を定義します。スコープを入力するときは、選択した IdP に基づいて、次のガイドラインに従ってください。

   - Facebook — スコープはカンマで区切ります。例:

     ```
     public_profile, email
     ```

   - Google、Login with Amazon、Sign in with Apple — スコープをスペースで区切ります。例:
     - Google: `profile email openid`
     - Login with Amazon: `profile postal_code`
     - Sign In with Apple: `name email`

     > **Note**
     >
     > Sign in with Apple (コンソール) の場合は、チェックボックスを使用してスコープを選択します。

7. [変更の保存] をクリックします。

8. [App client integration] (アプリケーションクライアント統合) タブで、リストにある [App clients] (アプリケーションクライアント) の 1 つを選択して、[Edit hosted UI settings] (ホストされた UI 設定の編集) をクリックします。[Identity providers] (ID プロバイダー) で、新しいソーシャル ID プロバイダーをアプリケーションクライアントに追加します。

9. [変更の保存] をクリックします。

ソーシャル の詳細については IdPs、「」を参照してくださいユーザープールでのソーシャル ID プロバイダーの使用。

OIDC IdP でユーザーサインインを設定する

Salesforce や Ping Identity などの OpenID Connect (OIDC) ID プロバイダーとユーザーサインインを統合することができます。

OIDC プロバイダーをユーザープールに追加するには

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、AWS 認証情報を入力します。
2. ナビゲーションメニューから [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を検索し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. [OpenID Connect] (OpenID 接続) ID プロバイダーを選択します。
6. [Provider name] (プロバイダー名) に一意の名前を入力します。
7. プロバイダーから受け取ったクライアント ID を、[Client ID] (クライアント ID) へ入力します。
8. プロバイダーから受け取ったクライアントシークレットを、[Client secret] (クライアントシークレット) に入力します。
9. このプロバイダーの [Authorized scopes] (承認済みスコープ) を入力します。スコープは、アプリケーションがプロバイダーにリクエストするユーザー属性のグループ (`name` および `email` など) を定義します。OAuth 2.0 仕様に従い、スコープはスペースで区切る必要があります。

   ユーザーはこれらの属性をアプリケーションに提供することに同意する必要があります。
10. [Attribute request method] (属性リクエストメソッド) を選択して、プロバイダーが操作する [userInfo] エンドポイントからユーザーの詳細をフェッチするために必要な HTTP メソッド（ GET または POST ）を Amazon Cognito に提供します。
11. [Setup method] (セットアップ方法) を選択して、OpenID Connect エンドポイントを、[Auto fill through issuer URL] (発行者 URL による自動入力) または [Manual input] (手動入力) で取得します。[Auto fill through issuer URL] (発行者 URL による自動入力) は、Amazon Cognito が

authorization、token、userInfo、および jwks_uri エンドポイントの URL を取得できるパブリック .well-known/openid-configuration エンドポイントを、プロバイダーが持っている場合に使用します。

12. 発行者の URL、または IdP からの authorization、token、userInfo、および jwks_uri エンドポイントの URL を入力します。

> **Note**
>
> 検出、自動入力、および手動で入力された URL には、ポート番号 443 と 80 のみを使用できます。OIDC プロバイダーが標準外の TCP ポートを使用している場合、ユーザーログインは失敗します。
> 発行者の URL は https:// で始まる必要があり、/ 文字で終わらせることはできません。例えば、Salesforce では次の URL を使用します。
> https://login.salesforce.com
> 発行者 URL に関連付けられている openid-configuration ドキュメントには、次の値の HTTPS URL を指定する必要があります: authorization_endpoint、token_endpoint、userinfo_endpoint、および jwks_uri。同様に、[Manual input] (手動入力) を選択する場合は、HTTPS URL のみを入力できます。

13. デフォルトで、OIDC クレームの [sub] (サブ) はユーザープール属性の [Username] (ユーザーネーム) にマッピングされます。他の OIDC クレームをユーザープール属性にマッピングできます。OIDC クレームを入力し、対応するユーザープール属性をドロップダウンリストから選択します。例えば、通常、クレームの [email] はユーザープール属性の [E メール] にマッピングされます。
14. ID プロバイダーからユーザープールに追加の属性をマッピングします。詳細は、「ユーザープール用 ID プロバイダー属性マッピングの特定」を参照してください。
15. [作成] を選択します。
16. [App client integration] (アプリケーションクライアント統合) タブから、リストにある [App clients] の 1 つを選択して、[Edit hosted UI settings] (ホストされた UI 設定の編集) をクリックします。[Identity providers] (ID プロバイダー) で、新しい OIDC ID プロバイダーをアプリケーションクライアントに追加します。
17. [変更の保存] をクリックします。

OIDC の詳細については IdPs、「」を参照してくださいユーザープールでの OIDC ID プロバイダーの使用。

SAML IdP でユーザーサインインを設定する

Amazon Cognito ユーザープールのフェデレーションを使用して、SAML ID プロバイダー (IdP) と統合することができます。メタデータドキュメントを指定します。ファイルをアップロードするか、メタデータドキュメントのエンドポイント URL を入力します。サードパーティー SAML のメタデータドキュメントの取得については IdPs、「」を参照してくださいサードパーティーの SAML ID プロバイダーの設定。

ユーザープールに SAML 2.0 ID プロバイダーを設定する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を検索し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. [SAML] ID プロバイダーを選択します。
6. カンマで区切られた [Identifiers] (識別子) を入力します。識別子は Amazon Cognito に、ユーザーのサインイン E メールアドレスを確認し、ドメインに対応するプロバイダーに誘導する必要があることを伝えます。
7. ユーザーがログアウトしたときに、Amazon Cognito が署名されたサインアウト要求をプロバイダーに送信するためには、[Add sign-out flow] (サインアウトフローの追加) を選択します。SAML 2.0 ID プロバイダーを設定して、ホストされた UI を構成するときに Amazon Cognito が作成する `https://mydomain.us-east-1.amazoncognito.com/saml2/logout` エンドポイントにサインアウト応答を送信します。`saml2/logout` エンドポイントでは、ポストバインディングを使用します。

> **Note**
>
> このオプションを選択し、SAML ID プロバイダーが署名付きログアウトリクエストを期待する場合は、SAML IdP で Amazon Cognito が提供する署名証明書を設定する必要もあります。
> SAML IdP は署名されたログアウトリクエストを処理し、Amazon Cognito セッションからユーザーをログアウトさせます。

8. [Metadata document source] (メタデータドキュメントソース) を選択します。ID プロバイダーがパブリック URL で SAML メタデータを提供する場合は、[Metadata document URL] (メタデータドキュメント URL) を選択してそのパブリック URL を入力できます。それ以外の場合は、[Upload metadata document] (メタデータドキュメントをアップロード) を選択し、プロバイダーから以前ダウンロードしたメタデータファイルを選択します。

   > **Note**
   >
   > プロバイダーにパブリックエンドポイントがある場合は、ファイルをアップロードするのではなく、メタデータドキュメントの URL を入力することをお勧めします。URL を使用する場合、Amazon Cognito はメタデータを自動的に更新します。通常、メタデータの更新は 6 時間ごとまたはメタデータの有効期限が切れる前のいずれか早いタイミングで発生します。

9. [Map attributes between your SAML provider and your app] (SAML プロバイダーとアプリ間で属性をマッピング) をクリックして、SAML プロバイダーの属性をユーザープールのユーザープロファイルにマッピングします。ユーザープールの必須属性を属性マップに含めます。

   たとえば、[User pool attribute] (ユーザープール属性) `email` を選択する場合、ID プロバイダーからの SAML アサーションに表示される SAML 属性名を入力します。ID プロバイダーは、参考として SAML アサーションのサンプルを提供する場合があります。ID プロバイダーの中には、`email` などの単純な名前を使用するものもあれば、次のような URL 形式の属性名を使用するものもあります。

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. [作成] を選択します。

> **Note**
>
> HTTPS メタデータエンドポイント URL を使用して SAML IdP を作成中に `InvalidParameterException` が表示される場合、メタデータエンドポイントの SSL が正しくセットアップされていること、および有効な SSL 証明書が関連付けられていることを確認してください。このような例外の例として、「Error retrieving metadata from *<metadata endpoint>*」が挙げられます。

署名証明書を追加するために SAML IdP をセットアップする

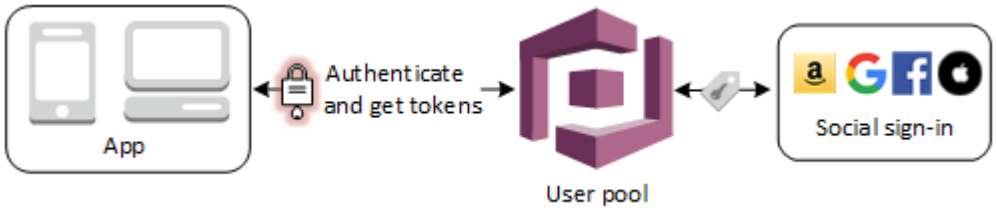
- 署名付きログアウトリクエストを検証するために IdP が使用するパブリックキーを含む証明書を取得するには、[フェデレーション] コンソールページの [ID プロバイダー] にある [SAML] ダイアログで [アクティブな SAML プロバイダー] の [デジタル署名用証明書の表示] を選択します。

SAML の詳細については、 IdPs 「」を参照してくださいユーザープールでの SAML ID プロバイダーの使用。

ユーザープールでのソーシャル ID プロバイダーの使用

ウェブおよびモバイルアプリのユーザーは、Facebook、Google、Amazon、Apple などのソーシャル ID プロバイダー (IdP) 経由でサインインできます。組み込みの Hosted Web UI では、Amazon Cognito がすべての認証済みユーザーに関するトークンの処理と管理を提供します。このように、バックエンドシステムは 1 セットのユーザープールトークンで標準化できます。サポートされているソーシャル ID プロバイダーと統合するには、ホストされた UI を有効にする必要があります。Amazon Cognito がホストされた UI を構築すると、Amazon Cognito と OIDC、およびソーシャルが情報の交換 IdPs に使用する OAuth 2.0 エンドポイントが作成されます。詳細については、Amazon Cognito API リファレンスを参照してください。

ソーシャル IdP を に追加するか AWS Management Console、 AWS CLI または Amazon Cognito API を使用できます。



> **Note**
>
> サードパーティー (フェデレーション) 経由のサインインは、Amazon Cognito のユーザープールで使用できます。この機能は、Amazon Cognito ID プール (フェデレーティッド ID) 経由のフェデレーションとは無関係です。

トピック

- 前提条件
- ステップ 1: ソーシャル IdP に登録する

- ステップ 2: ユーザープールにソーシャル IdP を追加する
- ステップ 3: ソーシャル IdP の設定をテストする

前提条件

開始するには、以下が必要です。

- アプリケーションクライアントとユーザープールドメインを使用するユーザープール。詳細については、「ユーザープールの作成」を参照してください。
- ソーシャル IdP。

ステップ 1: ソーシャル IdP に登録する

Amazon Cognito でソーシャル IdP を作成する前に、アプリケーションをソーシャル IdP に登録して、クライアント ID とクライアントシークレットを取得する必要があります。

アプリケーションを Facebook に登録する

1. Facebook の開発者アカウントを作成します。
2. Facebook 認証情報を使用してサインインします。
3. [My Apps] (マイアプリ) メニューから、[新しいアプリを作成] (新しいアプリを作成) を選択します。
4. Facebook アプリケーションの名前を入力し、[Create App ID] (アプリケーション ID の作成) を選択します。
5. 左のナビゲーションバーで、[設定]、[ベーシック] の順に選択します。
6. [App ID] (アプリ ID)] と [App Secret (アプリシークレット) を書き留めます。これらは次のセクションで使用します。
7. ページの下部で、[+ プラットフォームを追加] を選択します。
8. [Website] (ウェブサイト) を選択します。
9. [Website] (ウェブサイト) で、アプリケーションのサインインページへのパスを [Site URL] (サイト URL) に入力します。

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

10. [変更を保存] を選択します。
11. ユーザープールドメインのルートへのパスを [App Domains] (アプリケーションドメイン) に入力します。

```
https://mydomain.us-east-1.amazoncognito.com
```

12. [変更を保存] を選択します。
13. ナビゲーションバーで [Add Product] (プロダクトの追加) を選択し、[Facebook Login] (Facebook ログイン) の [Set up] (設定) を選択します。
14. ナビゲーションバーで [Facebook ログイン]、[設定] の順に選択します。

    ユーザープールドメインの `/oauth2/idpresponse` エンドポイントへのパスを [Valid OAuth Redirect URIs] (有効な OAuth リダイレクト URI) に入力します。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

15. [変更を保存] を選択します。

アプリケーションを Amazon に登録する

1. Amazon の開発者アカウントを作成します。
2. Amazon 認証情報を使用してサインインします。
3. Amazon クライアント ID およびクライアントシークレットを受け取るには、Amazon セキュリティプロファイルを作成する必要があります。

   ページの上部にあるナビゲーションバーで [Apps and Services (アプリとサービス)] を選択し、[Login with Amazon] を選択します。
4. [Create a New Security Profile (新しいセキュリティプロファイルの作成)] を選択します。
5. [Security Profile Name] (セキュリティプロファイル名)、[Security Profile Description] (セキュリティプロファイルの説明)、[Consent Privacy Notice URL] (プライバシー規約 URL の同意) に入力します。
6. [Save] を選択します。
7. [クライアント ID] および [クライアントシークレット] を選択して、クライアント ID およびシークレットを表示します。これらは次のセクションで使用します。

8. 歯車アイコンにマウスカーソルを合わせ、[Web Settings] (ウェブ設定)、[Edit] (編集) の順に選択します。
9. [Allowed Origins] (許可されたオリジン) にユーザープールのドメインを入力します。

```
https://mydomain.us-east-1.amazoncognito.com
```

10. `/oauth2/idpresponse` エンドポイントを使用するユーザープールドメインを [Allowed Return URLs] (許可されたリターン URL) に入力します。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

11. [Save] を選択します。

アプリケーションを Google に登録する

Google Cloud プラットフォームの OAuth 2.0 の詳細については、「Google Workspace for Developers」(Google Workspace デベロッパーガイド) ドキュメントの「Learn about authentication & authorization」(認証と認可について学ぶ) を参照してください。

1. Google の開発者アカウントを作成します。
2. Google Cloud Platform コンソールにサインインします。
3. 上部のナビゲーションバーから、[Select a project] (プロジェクトの選択) を選択します。Google プラットフォームにプロジェクトが既にある場合は、このメニューには代わりにデフォルトのプロジェクトが表示されます。
4. [NEW PROJECT] (新しいプロジェクト) を選択します。
5. 製品の名前を入力し、[CREATE] (作成) を選択します。
6. 左のナビゲーションバーで、[APIs and Services] (API とサービス) を選択し、[Oauth consent screen] (OAuth 同意画面) を選択します。
7. アプリ情報、[App domain] (アプリドメイン)、[Authorized domains] (承認済みドメイン)、[Developer contact information] (開発者の連絡先情報) を入力します。[Authorized domains] (承認済みドメイン) には、`amazoncognito.com` とカスタムドメインのルート (例: `example.com`) を含める必要があります。[SAVE AND CONTINUE] (保存して続行) を選択します。
8. 1. [Scopes] (スコープ) で、[Add or remove scopes] (スコープの追加または削除) を選択し、少なくとも、次の OAuth スコープを選択します。

    1. `.../auth/userinfo.email`

2. `.../auth/userinfo.profile`
3. openid

9. [Test users] (テストユーザー) で、[Add users] (ユーザーの追加) を選択します。E メールアドレスおよびその他の承認済みテストユーザーを入力して、[SAVE AND CONTINUE] (保存して続行) を選択します。
10. 左のナビゲーションバーを再度展開し、[APIs and Services] (API とサービス) を選択して、[Credentials] (認証情報) を選択します。
11. [CREATE CREDENTIALS] (認証情報の作成)、[OAuth client ID] (OAuth クライアント ID) の順に選択します。
12. [Application type] (アプリケーションタイプ) を選択し、クライアントに [Name] (名前) を入力します。
13. 「許可された JavaScript オリジン」で、「URI を追加」を選択します。ユーザープールのドメインを入力します。

```
https://mydomain.us-east-1.amazoncognito.com
```

14. [Authorized redirect URIs] (承認済みのリダイレクト URI) で、[ADD URI] (URI の追加) を選択します。ユーザープールのドメインの `/oauth2/idpresponse` エンドポイントへのパスを入力します。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

15. [CREATE] (作成) を選択します。
16. [Your client ID] (クライアント ID) と [Your client secret] (クライアントシークレット) の下に Google が表示する値を安全に保存します。Google IdP を追加するときに、Amazon Cognito にこれらの値を指定します。

アプリケーションを Apple に登録する

「Apple でサインイン」の設定 up-to-date の詳細については、Apple デベロッパードキュメントの「Apple でサインインするための環境の設定」を参照してください。 https://developer.apple.com/documentation/sign_in_with_apple/configuring_your_environment_for_sign_in_with_apple

1. Apple の開発者アカウントを作成します。
2. Apple 認証情報を使用してサインインします。

3. 左のナビゲーションバーで、[Certificates, Identifiers & Profiles] (証明書、ID & プロファイル) を選択します。
4. 左のナビゲーションペインで、[Identifiers] (識別子) を選択します。
5. [Identifiers] (識別子) ページで、[+] アイコンを選択します。
6. [Register a New Identifier] (新しい識別子の登録) ページで、[App IDs] (アプリ ID)、[Continue] (続行) の順に選択します。
7. [Select a type] (種類の選択) ページで、[App] (アプリ) を選択し、[Continue] (続行) を選択します。
8. [Register an App ID] ページで、次の操作を行います。
    1. [Description] (説明) で、説明を入力します。
    2. [App ID Prefix] (アプリ ID プレフィックス) に、[Bundle ID] (バンドル ID) を入力します。[App ID Prefix] (アプリケーション ID プレフィックス) にある値を書き留めておきます。この値は、ステップ 2: ユーザープールにソーシャル IdP を追加する で Apple を ID プロバイダーとして選択した後で使用します。
    3. [Capabilities] (機能) で、[Sign In with Apple] (Apple でサインイン) を選択してから [Edit] (編集) を選択します。
    4. [Sign in with Apple: App ID Configuration] (Apple でサインイン: アプリ ID の設定) ページで、アプリをプライマリとして設定するか、他のアプリ ID とグループ化するかを選択し、次に [Save] (保存) を選択します。
    5. [Continue] を選択します。
9. [Confirm your App ID] ページで、[登録] を選択します。
10. [Identifiers] (識別子) ページで、[+] アイコンを選択します。
11. [Register a New Identifier] (新しい識別子の登録) ページで、[Services IDs] (サービス ID)、[Continue] (続行) の順に選択します。
12. [Register a Services ID] ページで、次の操作を行います。
    1. [説明] に、説明を入力します。
    2. [Identifier] に、識別子を入力します。このサービス ID をメモしておきます。この値は、ステップ 2: ユーザープールにソーシャル IdP を追加する で Apple を ID プロバイダーとして選択した後で必要になります。
    3. [Continue] (続行) を選択し、[Register] (登録) を選択します。
13. [Identifiers] (識別子) ページから、先ほど作成した [Services ID] (サービス ID) を選択します。

1. [Sign In with Apple] (Apple でサインイン) を選択後、[Configure] (設定) を選択します。
2. [Web Authentication Configuration] (ウェブ認証の設定) ページで、[Primary App ID] (プライマリアプリ ID) として前に作成したアプリ ID を選択します。
3. [Website URLs] (ウェブサイトの URL) の横の [+] アイコンを選択します。
4. [Domains and subdomains] (ドメインとサブドメイン) で、`https://` プレフィックスなしでユーザープールのドメインを入力します。

   ```
   mydomain.us-east-1.amazoncognito.com
   ```

5. [Return URLs] (URL を返す) で、ユーザープールのドメインの `/oauth2/idpresponse` エンドポイントへのパスを入力します。

   ```
   https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
   ```

6. [Next] (次へ) を選択し、[Done] (完了) を選択します。ドメインを検証する必要はありません。
7. [Continue] (続行) を選択し、次に [Save] (保存) を選択します。

14. 左のナビゲーションペインで [Keys] (キー) を選択します。
15. [Keys (キー)] ページで、[+] アイコンを選択します。
16. [Register a New Key] ページで、次の操作を行います。

    1. [Key Name] (キー名) に、キー名を入力します。
    2. [Sign In with Apple] (Apple でサインイン) を選択後、[Configure] (設定) を選択します。
    3. [Configure Key] (キーの設定) ページで、プライマリアプリ ID として前に作成したアプリ ID を選択します。[保存] を選択します。
    4. [Continue] (続行) を選択し、[Register] (登録) を選択します。

17. [Download Your Key] (鍵のダウンロード) ページで、[Download] (ダウンロード) をクリックしてプライベートキーをダウンロードしてから、表示される [Key ID] (キー ID) を書き留め、[Done] (完了) を選択します。このプライベートキーと、このページに表示されている [Key ID] (キー ID) の値は、[ステップ 2: ユーザープールにソーシャル IdP を追加する] で Apple を ID プロバイダーとして選択した後で必要になります。

ステップ 2: ユーザープールにソーシャル IdP を追加する

を使用してユーザープールのソーシャル IdP を設定するには AWS Management Console

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を探し、[Add an identity provider] (ID プロバイダーを追加する) を選択します。
5. ソーシャル IdP ([Facebook]、[Google]、[Login with Amazon]、[Sign in with Apple]) を選択します。
6. ソーシャル IdP の選択に基づいて、次のステップから選択します。

   - Google と Login with Amazon (Amazon でのログイン) — 前のセクションで生成された [app client ID] (アプリケーションクライアント ID) と [app client secret] (アプリケーションクライアントシークレット) を入力します。
   - Facebook — 前のセクションで生成された [app client ID] (アプリケーションクライアント ID) と [app client secret] (アプリケーションクライアントシークレット) を入力し、API バージョン (例えば、バージョン 2.12) を選択します。Facebook API の各バージョンにはライフサイクルとサポート終了日があるため、可能な限り最新のバージョンを選択することをお勧めします。Facebook のスコープと属性は API バージョンによって異なる場合があります。Facebook でソーシャル ID ログインをテストして、フェデレーションが意図したとおりに機能することを確認することをお勧めします。
   - Sign In with Apple (Apple でのサインイン) — 以前のセクションで作成された [Services ID] (サービス ID)、[Team ID] (チーム ID)、[Key ID] (キー ID) および [private key] プライベートキーを入力します。
7. 使用したい [Authorized scopes] (承認するスコープ) の名前を入力します。スコープは、アプリでアクセスするユーザー属性 (`name` や `email` など) を定義します。Facebook の場合は、コンマで区切る必要があります。Google および Login with Amazon の場合は、スペースで区切って指定します。Sign in with Apple の場合は、アクセスするスコープのチェックボックスをオンにします。

ソーシャル ID プロバイダー	スコープ例
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Apple でのサインイン	`email name`

アプリケーションユーザーは、これらの属性をアプリケーションに提供すること
に同意するよう求められます。ソーシャルプロバイダーのスコープの詳細について
は、Google、Facebook、Login with Amazon、および Sign in with Apple のドキュメントを参照
してください。

以下は、Sign in with Apple の場合にスコープが返らない可能性があるユーザーシナリオです。

- エンドユーザーが Apple のサインインページを離れるとエラーが発生する (Amazon Cognito の内部障害またはデベロッパーが記述したプログラムが原因である可能性があります)
- サービス ID 識別子がユーザープールや他の認証サービス全体で使用されている
- エンドユーザーが以前にサインインしてから、デベロッパーによってさらにスコープが追加されている (新しい情報は取得されません)
- デベロッパーによって削除されたユーザーが Apple ID プロファイルからアプリを削除せずに再度サインインしている

8. IdP からユーザープールに属性をマッピングします。詳細は、「ユーザープール用 ID プロバイダー属性マッピングの特定」を参照してください。
9. [作成] を選択します。
10. [App client integration] (アプリケーションクライアント統合) タブから、[App clients] (アプリケーションクライアント) のリストより 1 つ選択して、[Edit hosted UI settings] (ホストされた UI 設定の編集) をクリックします。[Identity providers] (ID プロバイダー) で、新しいソーシャル IdP をアプリクライアントに追加します。
11. [変更を保存] を選択します。

ステップ 3: ソーシャル IdP の設定をテストする

前の 2 つのセクションの要素を使用してログイン URL を作成できます。この URL を使用してソーシャル IdP の設定をテストします。

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

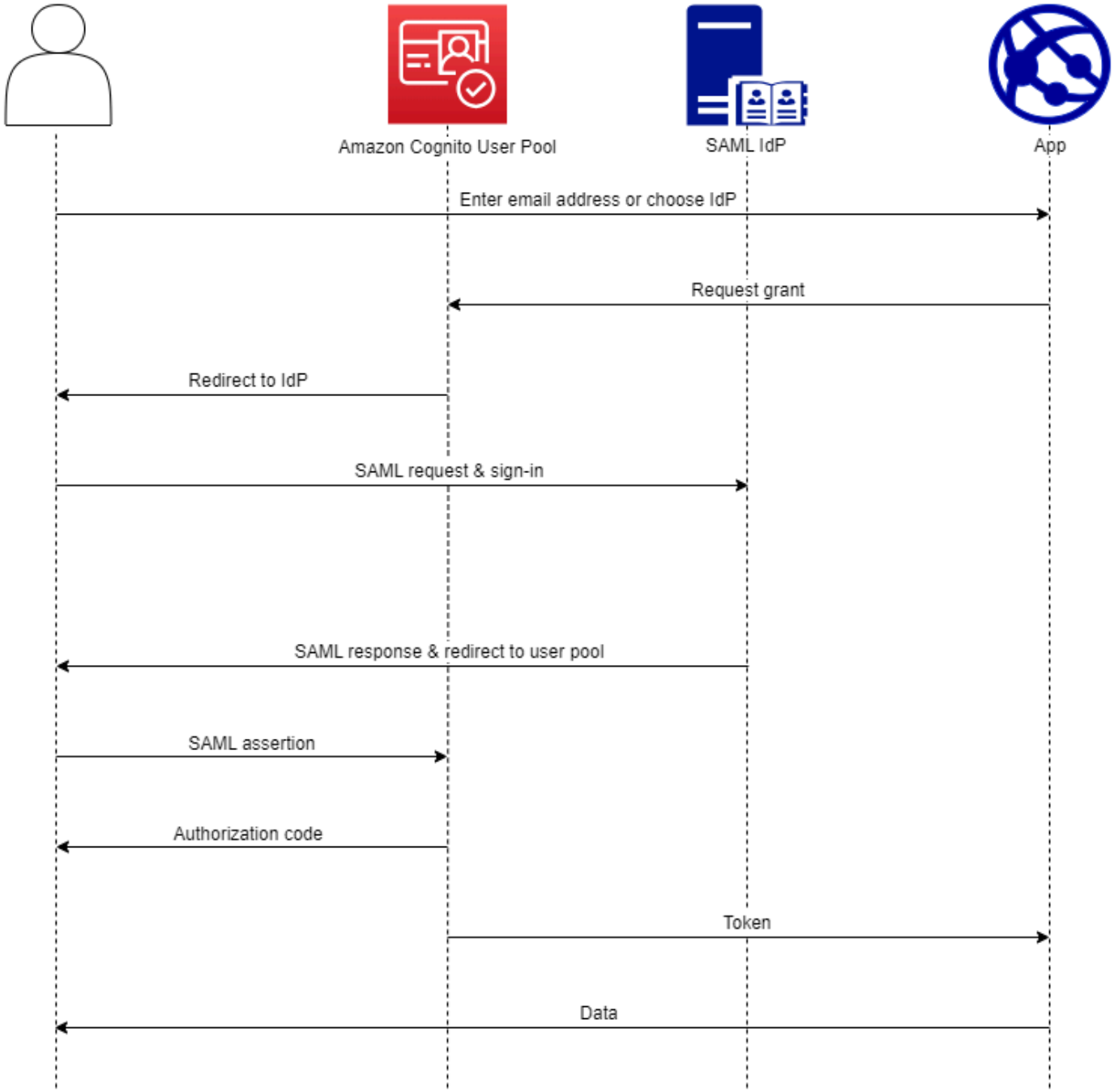
ドメインは、コンソールにあるユーザープールの [ドメイン名] ページで確認できます。クライアント ID は [アプリクライアントの設定] ページにあります。redirect_uri パラメータのコールバック URL を使用します。これは、ユーザーが認証に成功した後でリダイレクトされるページの URL です。

> **Note**
>
> Amazon Cognito は 5 分以内に完了しない認証リクエストをキャンセルし、ホストされた UI にユーザーをリダイレクトします。ページには、`Something went wrong` というエラーメッセージが表示されます。

ユーザープールでの SAML ID プロバイダーの使用

Microsoft Active Directory フェデレーションサービス (ADFS) や Shibboleth などの SAML ID プロバイダー (IdP) を介してウェブおよびモバイルアプリのユーザーにサインインさせることができます。https://msdn.microsoft.com/en-us/library/bb897402.aspxSAML 2.0 標準をサポートする SAML IdP を選択します。

ホストされた UI とフェデレーションエンドポイントを使用すると、Amazon Cognito はローカルおよびサードパーティーの IdP ユーザーを認証し、JSON ウェブトークン (JWTs。Amazon Cognito が発行するトークンを使用すると、複数の ID ソースをすべてのアプリでユニバーサル OpenID Connect (OIDC) 標準に統合できます。Amazon Cognito は、サードパーティープロバイダーからの SAML アサーションをその SSO 標準に処理できます。SAML IdP は AWS Management Console、、、 AWS CLIまたは Amazon Cognito ユーザープール API を使用して作成および管理できます。で最初の SAML IdP を作成するには AWS Management Console、「」を参照してくださいユーザープールでの SAML ID プロバイダーの追加と管理。



Note

サードパーティーの IdP を介したサインインによるフェデレーションは、Amazon Cognito ユーザープールの機能です。Amazon Cognito フェデレーティッド ID と呼ばれる Amazon Cognito ID プールは、フェデレーションの実装であり、各 ID プールで個別に設定する必要

があります。ユーザープールは、ID プールへのサードパーティー IdP にすることができます。詳細については、「Amazon Cognito アイデンティティプール」を参照してください。

IdP 設定のクイックリファレンス

リクエストを受け入れ、ユーザープールにレスポンスを送信するように SAML IdP を設定する必要があります。SAML IdP のドキュメントには、SAML 2.0 IdP の証明書利用者またはアプリケーションとしてユーザープールを追加する方法に関する情報が含まれています。次のドキュメントでは、SP エンティティ ID とアサーションコンシューマーサービス (ACS) URL に対して指定する必要がある値を示します。

ユーザープールの SAML 値のクイックリファレンス

SP エンティティ ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS URL

```
https://Your user pool domain/saml2/idpresponse
```

ID プロバイダーをサポートするようにユーザープールを設定する必要があります。外部 SAML IdP を追加する大まかな手順は次のとおりです。

1. IdP から SAML メタデータをダウンロードするか、メタデータエンドポイントへの URL を取得します。サードパーティーの SAML ID プロバイダーの設定 を参照してください。
2. ユーザープールに新しい IdP を追加します。SAML メタデータをアップロードするか、メタデータ URL を指定します。ユーザープールでの SAML ID プロバイダーの追加と管理 を参照してください。
3. アプリクライアントに IdP を割り当てます。「ユーザープールアプリクライアント」を参照

トピック

- Amazon Cognito ユーザープール IdPs の SAML について知っておくべきこと
- SAML ユーザー名の大文字と小文字の区別
- ユーザープールでの SAML ID プロバイダーの追加と管理

- Amazon Cognito ユーザープールでの SAML セッション開始
- SP 開始 SAML サインインの使用
- IdP が開始する SAML サインインの使用
- SAML サインアウトフロー
- SAML 署名と暗号化
- SAML ID プロバイダーの名前と識別子
- サードパーティーの SAML ID プロバイダーの設定

Amazon Cognito ユーザープール IdPs の SAML について知っておくべきこと

Amazon Cognito は SAML アサーションを自動的に処理します

Amazon Cognito ユーザープールは、POST バインドエンドポイントで SAML 2.0 フェデレーションをサポートします。これにより、ユーザープールがユーザーエージェント経由で IdP から直接 SAML レスポンスを受信するため、アプリケーションで SAML アサーションレスポンスを受信して解析する必要がなくなります。ユーザープールはアプリケーションのためのサービスプロバイダー (SP) として機能します。Amazon Cognito は、SAML V2.0 技術概要 のセクション 5.1.2 および 5.1.4 で説明されているように、SP 開始および IdP 開始のシングルサインオン (SSO) をサポートしています。

有効な IdP 署名証明書を提供する

ユーザープールで SAML IdP を設定するときに、SAML プロバイダーメタデータの署名証明書の有効期限が切れないようにしてください。

ユーザープールが複数の署名証明書をサポート

SAML IdP の SAML メタデータに複数の署名証明書が含まれている場合、サインイン時に、SAML アサーションが SAML メタデータ内のいずれかの証明書と一致すると、ユーザープールは SAML アサーションが有効であると判断します。各署名証明書の長さは 4,096 文字以下にする必要があります。

リレーステートパラメータを維持する

Amazon Cognito と SAML IdP は、セッション情報を `relayState` パラメータで管理します。

1. Amazon Cognito は、80 バイトを超える `relayState` 値をサポートします。SAML 仕様では、`relayState` の値は「長さが 80 バイトを超えてはならない」と規定されていますが、現在の業界の慣行はこの動作から外れていることがよくあります。その結果、80 バイトを超える `relayState` 値を拒否すると、多くの標準的な SAML プロバイダーの統合は壊れます。

2. `relayState` トークンは、Amazon Cognito によって維持される状態情報への不透明な参照です。Amazon Cognito は `relayState` パラメータの内容を保証しません。その内容を解析してアプリケーションでその結果に依存することは避けてください。詳細については、「SAML 2.0 の仕様」を参照してください。

ACS エンドポイントを特定する

SAML ID プロバイダーは、アサーションコンシューマーエンドポイントを設定することを要求します。IdP は、SAML アサーションを使用して、このエンドポイントにユーザーをリダイレクトします。SAML ID プロバイダーの SAML 2.0 POST バインド用に、ユーザープールドメインで次のエンドポイントを設定します。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://auth.example.com/saml2/idpresponse
```

ユーザープールドメインの詳細については、「ユーザープールのドメインを設定する」を参照してください。

アサーションが再生されない

Amazon Cognito `saml2/idpresponse` エンドポイントに対して SAML アサーションを繰り返したり、再生したりすることはできません。SAML アサーションを作成すると、そのアサーション ID は以前の IdP レスポンスの ID と重複します。

ユーザープール ID は SP エンティティ ID です

IdP には`urn`、オーディエンス URI または SP エンティティ ID とも呼ばれるサービスプロバイダー (SP) のユーザープール ID を指定する必要があります。ユーザープールのオーディエンス URI の形式は次のとおりです。

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ユーザープール ID は、Amazon Cognito コンソールのユーザープールの概要にあります。Amazon Cognito

必要な属性をすべてマッピングする

ユーザープールで必須として設定した属性の値を入力するように SAML IdP を設定します。例えば、`email` はユーザープールの一般的な必須属性です。ユーザーがサインインする前に、ユー

ザープール属性 email にマッピングするクレームを SAML IdP アサーションに含める必要があります。属性のマッピングの詳細については、「ユーザープールの ID プロバイダー属性マッピングを指定する」を参照してください。

アサーション形式には特定の要件があります

SAML IdP は、SAML アサーションに次のクレームを含める必要があります。

1. NameID クレーム。Amazon Cognito は、 によって SAML アサーションを送信先ユーザーに関連付けますNameID。NameID 変更した場合、Amazon Cognito はアサーションを新しいユーザー用と見なします。IdP 設定NameIDで に設定する属性には永続的な値が必要です。

```
<saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
  carlos
</saml2:NameID>
```

2. ユーザープールの SP エンティティ ID をレスポンスのターゲットとして設定する値 Audience を含む AudienceRestriction クレーム。

```
<saml:AudienceRestriction>
  <saml:Audience> urn:amazon:cognito:sp:us-east-1_EXAMPLE
</saml:AudienceRestriction>
```

3. SP 開始のシングルサインオンの場合、元の SAML リクエスト ID InResponseToの値を持つResponse要素。

```
<saml2p:Response Destination="https://mydomain.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IdP が開始する SAML アサーションにInResponseTo値を含めることはできません。

4. ユーザープールsaml2/idpresponseエンドポイントRecipientの値を持つSubjectConfirmationData要素、および SP 開始 SAML の場合は、元の SAML リクエスト ID に一致するInResponseTo値。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP が開始するサインインリクエスト

認可エンドポイント がユーザーを IdP サインインページにリダイレクトすると、Amazon Cognito は HTTP GET リクエストの URL パラメータに SAML リクエストを含めます。SAML リクエストには、ACS エンドポイントを含むユーザープールに関する情報が含まれています。オプションで、これらのリクエストに暗号化署名を適用できます。

リクエストに署名してレスポンスを暗号化する

SAML プロバイダーを持つすべてのユーザープールは、Amazon Cognito が SAML リクエストに割り当てるデジタル署名の非対称キーペアと署名証明書を生成します。暗号化された SAML レスポンスをサポートするように設定したすべての外部 SAML IdP により、Amazon Cognito はそのプロバイダーの新しいキーペアと暗号化証明書を生成します。パブリックキーを使用して証明書を表示およびダウンロードするには、Amazon Cognito コンソールのサインインエクスペリエンスタブで IdP を選択します。

ユーザープールからの SAML リクエストとの信頼を確立するには、ユーザープールの SAML 2.0 署名証明書のコピーを IdP に提供します。署名付きリクエストを受け入れるように IdP を設定しない場合、IdP はユーザープールが署名した SAML リクエストを無視することがあります。

1. Amazon Cognito は、ユーザーが IdP に渡す SAML リクエストにデジタル署名を適用します。ユーザープールはすべてのシングルログアウト (SLO) リクエストに署名し、SAML 外部 IdP のシングルサインオン (SSO) リクエストに署名するようにユーザープールを設定できます。証明書のコピーを提供すると、IdP はユーザーの SAML リクエストの整合性を検証できます。
2. SAML IdP は、暗号化証明書を使用して SAML レスポンスを暗号化できます。SAML 暗号化を使用して IdP を設定する場合、IdP は暗号化されたレスポンスのみを送信する必要があります。

英数字以外の文字をエンコードする

Amazon Cognito は、IdP が属性値として渡す #や ���などの 4 バイトの UTF-8 文字を受け入れません。文字を Base 64 にエンコードしてテキストとして渡し、アプリでデコードできます。

次の例では、属性のクレームは受け付けられません。

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
```

```
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">#</saml2:AttributeValue>
</saml2:Attribute>
```

上記の例とは対照的に、次の属性のクレームは受け付けられます。

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

メタデータエンドポイントには有効なトランスポートレイヤーセキュリティが必要です

HTTPS メタデータエンドポイント URL を使用して SAML IdP を作成中に `InvalidParameterException` (例えば「Error retrieving metadata from *<metadata endpoint>*」) が表示される場合、メタデータエンドポイントの SSL が正しくセットアップされていること、および有効な SSL 証明書が関連付けられていることを確認してください。証明書の検証の詳細については、「SSL/TLS 証明書とは」を参照してください。

IdP が開始する SAML を持つアプリケーションクライアントは、SAML でのみサインインできます

アプリクライアントで IdP 開始サインインをサポートする SAML 2.0 IdP のサポートをアクティブ化すると、そのアプリクライアント IdPs に追加できるのは他の SAML 2.0 のみです。この方法で設定されたアプリクライアントに、ユーザープールとすべての非 SAML 外部 ID プロバイダーにユーザーディレクトリを追加することはできません。

ログアウトレスポンスは POST バインディングを使用する必要があります

`/saml2/logout` エンドポイントは をHTTP POSTリクエストLogoutResponseとして受け入れます。ユーザープールは、バHTTP GETインディングによるログアウトレスポンスを受け入れません。

SAML ユーザー名の大文字と小文字の区別

フェデレーティッドユーザーがサインインしようとすると、SAML ID プロバイダー (IdP) はユーザーの SAML アサーションで Amazon Cognito `NameId`に一意の を渡します。Amazon Cognito は、SAML フェデレーティッドユーザーを `NameId` クレームによって識別します。ユーザープールの大文字と小文字を区別する設定に関係なく、Amazon Cognito は、SAML IdP から返されるフェデレーティッドユーザーが一意で大文字と小文字を区別する`NameId`クレームを渡すと認識しま

す。email のような属性を NameId にマッピングし、ユーザーが E メールアドレスを変更すると、アプリケーションにサインインできません。

変更されない値を持つ IdP 属性から SAML アサーションに NameId をマッピングします。

例えば、Carlos は、Carlos@example.com の NameId 値を渡した Active Directory フェデレーションサービス (ADFS) SAML アサーションからの大文字と小文字を区別しないユーザープールにユーザープロファイルを持っているとします。次回 Carlos がサインインしようとすると、ADFS IdP は carlos@example.com の NameId 値を渡します。NameId は大文字と小文字が完全に一致する必要があるため、サインインは成功しません。

NameID を変更した後、ユーザーがログインできない場合は、ユーザープールからそのユーザーのプロファイルを削除してください。Amazon Cognito は、次回サインインしたときに新しいユーザープロファイルを作成します。

トピック

- ユーザープールでの SAML ID プロバイダーの追加と管理
- Amazon Cognito ユーザープールでの SAML セッション開始
- SP 開始 SAML サインインの使用
- IdP が開始する SAML サインインの使用
- SAML サインアウトフロー
- SAML 署名と暗号化
- SAML ID プロバイダーの名前と識別子
- サードパーティーの SAML ID プロバイダーの設定

ユーザープールでの SAML ID プロバイダーの追加と管理

次の手順は、Amazon Cognito ユーザープールで SAML プロバイダーを作成、変更、削除する方法を示しています。

AWS Management Console

を使用して、SAML ID プロバイダー () AWS Management Console を作成および削除できます IdPs。

SAML IdP を作成する前に、サードパーティーの IdP から取得した SAML メタデータドキュメントが必要です。必要な SAML メタデータドキュメントの取得または生成方法については、「サードパーティーの SAML ID プロバイダーの設定」を参照してください。

ユーザープールに SAML 2.0 IdP を設定するには

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を探し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. [SAML] IdP を選択します。
6. プロバイダー名 を入力します。このフレンドリ名は、 `identity_provider`リクエストパラメータで に渡すことができます認可エンドポイント。
7. カンマで区切られた [Identifiers] (識別子) を入力します。識別子は Amazon Cognito に、ユーザーがサインインしたときに入力したメールアドレスを確認し、ドメインに対応するプロバイダーに誘導する必要があることを伝えます。
8. ユーザーがログアウトしたときに、Amazon Cognito が署名されたサインアウト要求をプロバイダーに送信するためには、[Add sign-out flow] (サインアウトフローの追加) を選択します。SAML 2.0 IdP を設定して、ホストされた UI を構成するときに作成される `https://mydomain.us-east-1.amazoncognito.com/saml2/logout` エンドポイントにサインアウト応答を送信する必要があります。`saml2/logout` エンドポイントでは、ポストバインディングを使用します。

   > **Note**
   >
   > このオプションが選択されていて、SAML IdP が署名付きログアウトリクエストを想定している場合は、ユーザープールの署名証明書を SAML IdP に提供する必要があります。
   > SAML IdP は署名されたログアウトリクエストを処理し、Amazon Cognito セッションからユーザーをサインアウトさせます。

9. IdP が開始する SAML サインイン設定を選択します。セキュリティのベストプラクティスとして、SP 開始 SAML アサーションのみを受け入れる を選択します。未承諾の SAML サインインセッションを安全に受け入れるように環境を準備している場合は、SP 開始および IdP 開始の SAML アサーションを受け入れる を選択します。詳細については、「Amazon Cognito ユーザープールでの SAML セッション開始」を参照してください。

10. [Metadata document source] (メタデータドキュメントソース) を選択します。IdP がパブリック URL で SAML メタデータを提供する場合は、[Metadata document URL] (メタデータドキュメント URL) を選択してそのパブリック URL を入力できます。それ以外の場合は、[Upload metadata document] (メタデータドキュメントをアップロード) を選択し、プロバイダーから以前ダウンロードしたメタデータファイルを選択します。

    > **Note**
    >
    > ファイルをアップロードするのではなく、プロバイダーにパブリックエンドポイントがある場合は、メタデータドキュメント URL を入力することをお勧めします。Amazon Cognito は、メタデータ URL からメタデータを自動的に更新します。通常、メタデータの更新は 6 時間ごとまたはメタデータの有効期限が切れる前のいずれか早いタイミングで発生します。

11. SAML プロバイダーとユーザープールの間で属性をマッピングし、SAML プロバイダー属性をユーザープールのユーザープロファイルにマッピングします。ユーザープールの必須属性を属性マップに含めます。

    例えば、[User pool attribute] (ユーザープール属性) `email` を選択する場合、IdP からの SAML アサーションに表示される SAML 属性名を入力します。IdP がサンプル SAML アサーションを提供している場合は、これらのサンプルアサーションを使用して名前を見つけやすいかもしれません。などの単純な名前 IdPs を使用するものもあれば`email`、次のような名前を使用するものもあります。

    ```
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
    ```

12. [作成] を選択します。

API/CLI

SAML ID プロバイダー (IdP) を作成して管理するには、以下のコマンドを使用します。

IdP を作成し、メタデータドキュメントをアップロードする

- AWS CLI: `aws cognito-idp create-identity-provider`

  メタデータファイルの例: `aws cognito-idp create-identity-provider --user-pool-id` *`us-east-1_EXAMPLE`* `--provider-name=`*`SAML_provider_1`* `--provider-type SAML --provider-details file:///details.json --attribute-`

mapping *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

ここでは details.json に以下が含まれます。

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> *<SAML ##### XML>* に文字 のインスタンスが含まれている場合は"、エスケープ文字\として を追加する必要があります\"。

メタデータ URL の例: aws cognito-idp create-identity-provider --user-pool-id *us-east-1_EXAMPLE* --provider-name=*SAML_provider_1* --provider-type SAML --provider-details MetadataURL=*https://myidp.example.com/sso/saml/metadata* --attribute-mapping *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

- AWS API: CreateIdentityProvider

IdP の新規メタデータドキュメントをアップロードする

- AWS CLI: aws cognito-idp update-identity-provider

  メタデータファイルの例: aws cognito-idp update-identity-provider --user-pool-id *us-east-1_EXAMPLE* --provider-name=*SAML_provider_1* --provider-details file:///details.json --attribute-mapping *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

  ここでは details.json に以下が含まれます。

```
"ProviderDetails": {
```

```
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> *<SAML ##### XML>* に文字 のインスタンスが含まれている場合は"、エスケープ文字\として を追加する必要があります\"。

メタデータ URL の例: `aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: UpdateIdentityProvider

固有の IdP に関する情報を取得するには

- AWS CLI: `aws cognito-idp describe-identity-provider`

  `aws cognito-idp describe-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1`
- AWS API: DescribeIdentityProvider

すべての に関する情報を一覧表示するには IdPs

- AWS CLI: `aws cognito-idp list-identity-providers`

  例: `aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --max-results 3`
- AWS API: ListIdentityProviders

IdP を削除する

- AWS CLI: `aws cognito-idp delete-identity-provider`

  ```
  aws cognito-idp delete-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1
  ```

- AWS API: DeleteIdentityProvider

ユーザープールを証明書利用者として追加するために SAML IdP をセットアップする

- ユーザープールのサービスプロバイダー URN は `urn:amazon:cognito:sp:us-east-1_EXAMPLE` です。Amazon Cognito には、SAML レスポンスでこの URN に一致するオーディエンス制限値が必要です。IdP から SP への応答メッセージに次の POST バインディングエンドポイントを使用するように IdP を設定します。

  ```
  https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
  ```

- SAML IdP は、SAML アサーションでユーザープールに必要な属性NameIDと を入力する必要があります。 NameIDは、ユーザープール内の SAML フェデレーティッドユーザーを一意に識別するために使用されます。IdP は、各ユーザーの SAML 名 ID を一貫した大文字と小文字を区別する形式で渡す必要があります。ユーザー名 ID の値にバリエーションがあると、新しいユーザープロファイルが作成されます。

SAML 2.0 IDP にデジタル署名用証明書を提供するには

- IdP が SAML ログアウトリクエストの検証に使用できるパブリックキーのコピーを Amazon Cognito からダウンロードするには、ユーザープールのサインインエクスペリエンスタブを選択し、IdP を選択し、署名証明書を表示 で、.crt としてダウンロード を選択します。

Amazon Cognito コンソールを使用して、ユーザープールで設定した SAML プロバイダーを削除できます。

SAML プロバイダーを削除する

1. Amazon Cognito コンソールにサインインします。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。

3. サインインエクスペリエンスタブを選択し、フェデレーティッド ID プロバイダーのサインインを見つけます。
4. IdPs 削除する SAML の横にあるラジオボタンを選択します。
5. [Delete identity provider] (ID プロバイダーの削除) のプロンプトが表示されたら、SAML プロバイダー名を入力して削除を確認し、[Delete] (削除) を選択します。

Amazon Cognito ユーザープールでの SAML セッション開始

Amazon Cognito は、サービスプロバイダーが開始する (SP が開始する) シングルサインオン (SSO) と IdP が開始する SSO をサポートしています。セキュリティのベストプラクティスとして、ユーザープールに SP 開始 SSO を実装します。SAML V2.0 技術概要のセクション 5.1.2 では、SP で開始される SSO について説明します。Amazon Cognito は、アプリケーションの ID プロバイダー (IdP) です。アプリケーションは、認証されたユーザーのトークンを取得するサービスプロバイダー (SP) です。ただし、サードパーティー IdP を使用してユーザーを認証する場合、Amazon Cognito は SP です。SAML 2.0 ユーザーが SP 開始フローで認証するときは、まず Amazon Cognito にリクエストを行い、認証のために IdP にリダイレクトする必要があります。

一部のエンタープライズユースケースでは、内部アプリケーションへのアクセスは Enterprise IdP がホストするダッシュボードのブックマークから始まります。ユーザーがブックマークを選択すると、IdP は SAML 応答を生成して SP に送信し、アプリケーションでユーザーを認証します。

IdP が開始する SSO をサポートするように、ユーザープールで SAML IdP を設定できます。IdP 開始認証をサポートしている場合、Amazon Cognito は SAML リクエストで認証を開始しないため、Amazon Cognito は受信した SAML レスポンスを要求したことを検証できません。SP 開始 SSO では、Amazon Cognito は元のリクエストに対して SAML レスポンスを検証する状態パラメータを設定します。SP 主導のサインインを使用すると、クロスサイトリクエスト偽造 (CSRF) を防ぐこともできます。

ユーザーがユーザープールでホストされた UI とやり取りしたくない環境で SP 開始 SAML を構築する方法の例については、「」を参照してくださいシナリオ例: エンタープライズダッシュボードで Amazon Cognito アプリケーションをブックマークする。

トピック

- シナリオ例: エンタープライズダッシュボードで Amazon Cognito アプリケーションをブックマークする

シナリオ例: エンタープライズダッシュボードで Amazon Cognito アプリケーションをブックマークする

ウェブアプリケーションへの Amazon Cognito ユーザープール SSO アクセスを提供するブックマークを SAML または OIDC IdP ダッシュボードに作成できます。Amazon Cognito には、ユーザーがホストされた UI でサインインする必要のない方法でリンクできます。これを行うには、次の形式で Amazon Cognito ユーザープール認可エンドポイントの にリダイレクトするサインインブックマークをポータルに追加します。

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&identity_provider=MySAMLIdP&client_id=1example23456789&redire
www.example.com
```

> **Note**
>
> 認証エンドポイントへのリクエストで、`identity_provider` パラメータの代わりに `idp_identifier` パラメータを使用することもできます。IdP 識別子は、ユーザープールに ID プロバイダーを作成するときに設定できる代替名または E メールドメインです。SAML ID プロバイダーの名前と識別子 を参照してください。

リクエストで適切なパラメータを使用すると、`/authorize` では、Amazon Cognito は SP が開始したサインインフローをサイレントに開始し、IdP でサインインするようにユーザーをリダイレクトします。

開始するには、ユーザープールに SAML IdP を追加します。サインインに SAML IdP を使用し、承認されたコールバック URL としてアプリの URL を持つアプリクライアントを作成します。アプリクライアントの詳細については、「ユーザープールアプリクライアント」を参照してください。

この認証されたアクセスをポータルにデプロイする前に、ホストされた UI からアプリケーションへの SP 開始サインインをテストします。Amazon Cognito で SAML IdP を設定する方法の詳細については、「サードパーティーの SAML ID プロバイダーの設定」を参照してください。

次の図表は、IDP 開始 SSO をエミュレートする認証フローを示しています。ユーザーは、会社ポータルのリンクから Amazon Cognito で認証できます。

要件を満たしたら、 `identity_provider`または `idp_identifier`パラメータ認可エンドポイントを含むブックマークを に作成します。ユーザー認証は次のように進行します。

1. ユーザーは SSO IdP ダッシュボードにサインインします。ユーザーがアクセスを許可されているエンタープライズアプリケーションには、このダッシュボードが表示されます。
2. ユーザーが Amazon Cognito で認証するアプリケーションへのリンクを選択します。多くの SSO ポータルでは、カスタムアプリリンクを追加できます。SSO ポータルでパブリック URL へのリンクを作成するために使用できる機能はすべて機能します。
3. SSO ポータル内のカスタムアプリリンクは、ユーザーをユーザープール 認可エンドポイント に誘導します。リンクには、`response_type`、`client_id`、`redirect_uri` および `identity_provider` のパラメータが含まれます。`identity_provider` パラメータは、ユーザープールで IdP に付けた名前です。`identity_provider` パラメータの代わりに `idp_identifier` パラメータを使用することもできます。ユーザーは、 `idp_identifier`または `identity_provider`パラメータを含むリンクからフェデレーションエンドポイントにアクセスします。このユーザーはサインインページをバイパスし、IdP で認証するために直接ナビゲートします。SAML の命名の詳細については IdPs、「」を参照してくださいSAML ID プロバイダーの名前と識別子。

   URL の例

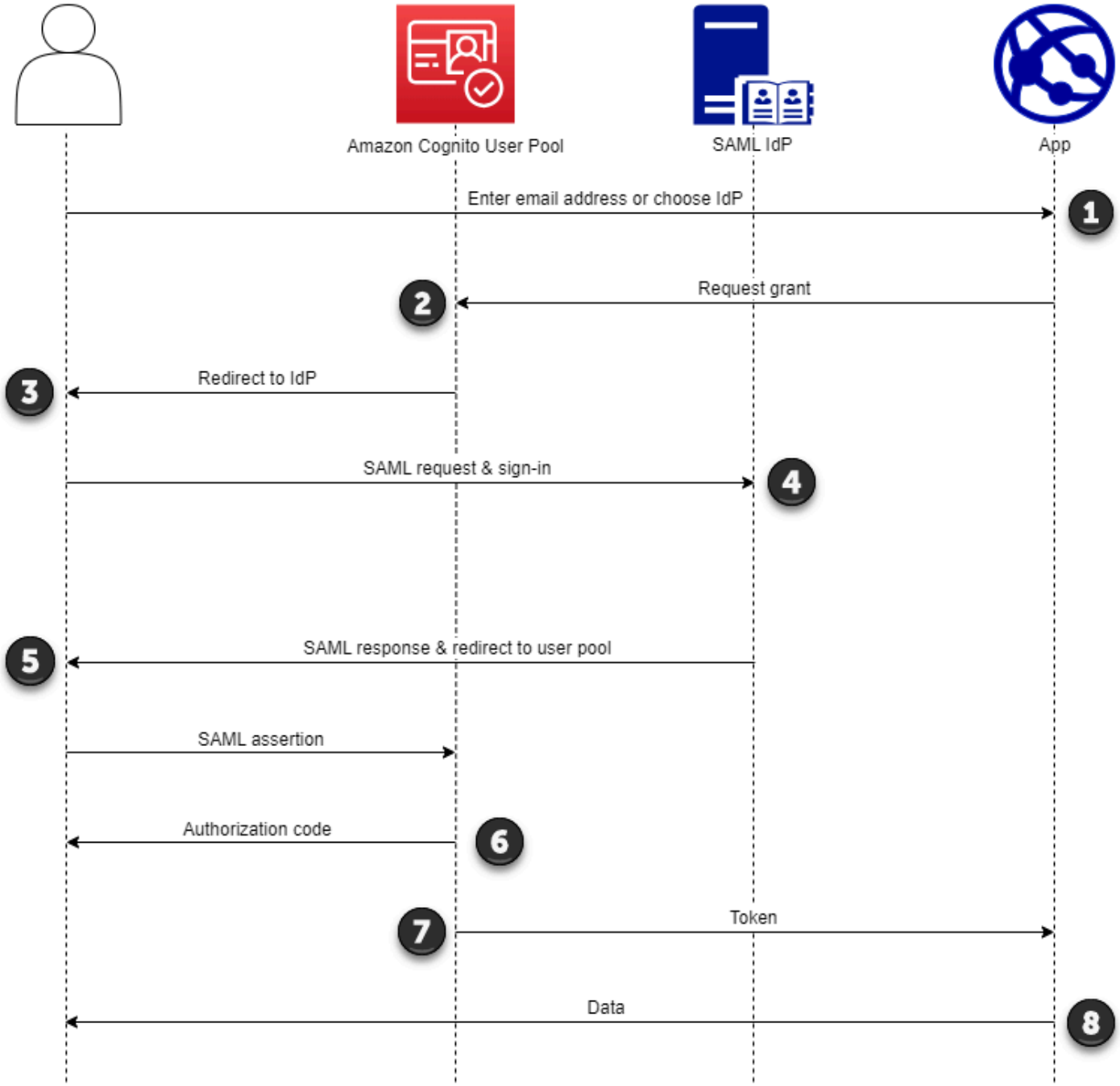
   ```
   https://mydomain.us-east-1.amazoncognito.com/authorize?
   response_type=code&
   identity_provider=MySAMLIdP&
   client_id=1example23456789&
   redirect_uri=https://www.example.com
   ```

4. Amazon Cognito は、SAML リクエストを使用してユーザーセッションを IdP にリダイレクトします。
5. ユーザーがダッシュボードにサインインしたときに、IdP からセッション Cookie を受信した可能性があります。IdP はこの cookie を使用してユーザーをそのまま検証し、SAML レスポンスで Amazon Cognito `idpresponse` エンドポイントにリダイレクトします。アクティブなセッションが存在しない場合、IdP は SAML レスポンスを投稿する前にユーザーを再認証します。
6. Amazon Cognito は SAML レスポンスを検証し、SAML アサーションに基づいてユーザープロファイルを作成または更新します。
7. Amazon Cognito は、認証コードを使用してユーザーを内部アプリケーションにリダイレクトします。内部アプリ URL をアプリクライアントの承認されたリダイレクト URL として設定しました。
8. アプリは認証コードを Amazon Cognito トークンと交換します。詳細については、「トークンエンドポイント」を参照してください。

SP 開始 SAML サインインの使用

ベストプラクティスとして、ユーザープールに service-provider-initiated (SP 開始) サインインを実装します。Amazon Cognito はユーザーのセッションを開始し、IdP にリダイレクトします。この方法では、サインインリクエストを提示するユーザーを最も細かく制御できます。特定の条件下で IdP 開始サインインを許可することもできます。詳細については、「Amazon Cognito ユーザープールでの SAML セッション開始」を参照してください。

以下のプロセスは、ユーザーが SAML プロバイダーを介してユーザープールにサインインする方法を示しています。



1. ユーザーはサインインページで E メールアドレスを入力します。ユーザーの IdP へのリダイレクトを確認するには、カスタムアプリケーションで E メールアドレスを収集するか、ウェブビューでホストされた UI を呼び出します。ホストされた UI を設定して、 のリストを表示する IdPs か、E メールアドレスのみの入力を求めることができます。
2. アプリはユーザープールリダイレクトエンドポイントを呼び出し、アプリに対応するクライアント ID とユーザーに対応する IdP ID を持つセッションをリクエストします。

3. Amazon Cognito は、 `AuthnRequest`要素でオプションで署名された SAML リクエストを使用して、ユーザーを IdP にリダイレクトします。 ???
4. IdP は、インタラクティブにユーザーを認証するか、ブラウザ Cookie で記憶されたセッションを使用してユーザーを認証します。
5. IdP は、POST ペイロード内のオプションで暗号化された SAML アサーションを使用して、ユーザーをユーザープールの SAML レスポンスエンドポイントにリダイレクトします。

   > **Note**
   >
   > Amazon Cognito は、5 分以内にレスポンスを受信しないセッションをキャンセルし、ユーザーをホストされた UI にリダイレクトします。ユーザーがこの結果を経験すると、`Something went wrong`エラーメッセージを受け取ります。

6. SAML アサーションを検証し、レスポンスのクレームからユーザー属性をマッピングすると、Amazon Cognito はユーザープールでユーザーのプロファイルを内部的に作成または更新します。通常、ユーザープールはユーザーのブラウザセッションに認証コードを返します。
7. ユーザーは認証コードをアプリケーションに提示し、アプリケーションはコードを JSON ウェブトークン (JWTs。
8. アプリはユーザーの ID トークンを認証として受け入れて処理し、アクセストークンを使用してリソースへの承認されたリクエストを生成し、更新トークンを保存します。

ユーザーが認証され、認証コードの付与を受け取ると、ユーザープールは ID、アクセス、および更新トークンを返します。ID トークンは、OIDC ベースの ID 管理用の認証オブジェクトです。アクセストークンは、OAuth 2.0 スコープを持つ認証オブジェクトです。更新トークンは、ユーザーの現在のトークンの有効期限が切れたときに新しい ID トークンとアクセストークンを生成するオブジェクトです。ユーザープールアプリクライアントでユーザーのトークンの期間を設定できます。

更新トークンの期間を選択することもできます。ユーザーの更新トークンの有効期限が切れたら、再度サインインする必要があります。SAML IdP を介して認証された場合、ユーザーのセッション期間は、IdP とのセッションの有効期限ではなく、トークンの有効期限によって設定されます。アプリは、各ユーザーの更新トークンを保存し、有効期限が切れたらセッションを更新する必要があります。ホストされた UI は、1 時間有効なブラウザ Cookie でユーザーセッションを維持します。

IdP が開始する SAML サインインの使用

IdP が開始する SAML 2.0 サインイン用に ID プロバイダーを設定すると、 でセッションを開始することなく、ユーザープールドメインの`saml2/idpresponse`エンドポイントに SAML アサーショ

ンを提示できます認可エンドポイント。この設定を持つユーザープールは、リクエストされたアプリケーションクライアントがサポートするユーザープールの外部 ID プロバイダーからの IdP 開始 SAML アサーションを受け入れます。次の手順では、IdP が開始する SAML 2.0 プロバイダーを使用して設定およびサインインする全体的なプロセスについて説明します。

1. ユーザープールとアプリケーションクライアントを作成または指定します。
2. ユーザープールに SAML 2.0 IdP を作成します。
3. IdP 開始をサポートするように IdP を設定します。IdP 主導の SAML では、他の SSO プロバイダーには適用されないセキュリティ上の考慮事項が導入されています。このため、IdP が開始するサインインで SAML プロバイダーを使用するアプリクライアントには IdPs、ユーザープール自体を含む SAML 以外の を追加することはできません。
4. IdP が開始する SAML プロバイダーをユーザープール内のアプリケーションクライアントに関連付けます。
5. SAML IdP のサインインページにユーザーを誘導し、SAML アサーションを取得します。
6. SAML アサーションを使用してユーザープール`saml2/idpresponse`エンドポイントにユーザーを誘導します。
7. JSON ウェブトークン (JWTs。

ユーザープールで未承諾の SAML アサーションを受け入れるには、アプリのセキュリティへの影響を考慮する必要があります。リクエストスプーフィングと CSRF の試行は、IdP が開始するリクエストを受け入れるときに行われる可能性があります。ユーザープールは IdP 開始のサインインセッションを検証できませんが、Amazon Cognito はリクエストパラメータと SAML アサーションを検証します。

さらに、SAML アサーションに`InResponseTo`クレームが含まれておらず、過去 6 分以内に発行された必要があります。

IdP 開始 SAML を使用してリクエストを に送信する必要があります`/saml2/idpresponse`。SP 開始およびホストされた UI 認証リクエストの場合、リクエストされたアプリケーションクライアント、スコープ、リダイレクト URI、およびその他の詳細を`HTTP GET`リクエストのクエリ文字列パラメータとして識別するパラメータを指定する必要があります。ただし、IdP が開始する SAML アサーションの場合、リクエストの詳細は`HTTP POST`リクエストの本文で`RelayState`パラメータとしてフォーマットする必要があります。リクエスト本文には、`SAMLResponse`パラメータとして SAML アサーションも含める必要があります。

以下は、IdP が開始する SAML プロバイダーに対するリクエストの例です。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

AWS Management Console

IdP が開始する SAML に IdP を設定するには

1. ユーザープール 、アプリケーションクライアント 、および SAML ID プロバイダーを作成します。
2. 関連付けられている場合は、すべてのソーシャル ID プロバイダーと OIDC ID プロバイダーをアプリクライアントから関連付け解除します。
3. ユーザープールのサインインエクスペリエンスタブに移動します。
4. フェデレーティッド ID プロバイダーのサインイン で、SAML プロバイダーを編集または追加します。
5. IdP が開始する SAML サインイン で、SP が開始する と IdP が開始する SAML アサーションを受け入れる を選択します。
6. [変更を保存] を選択します。

API/CLI

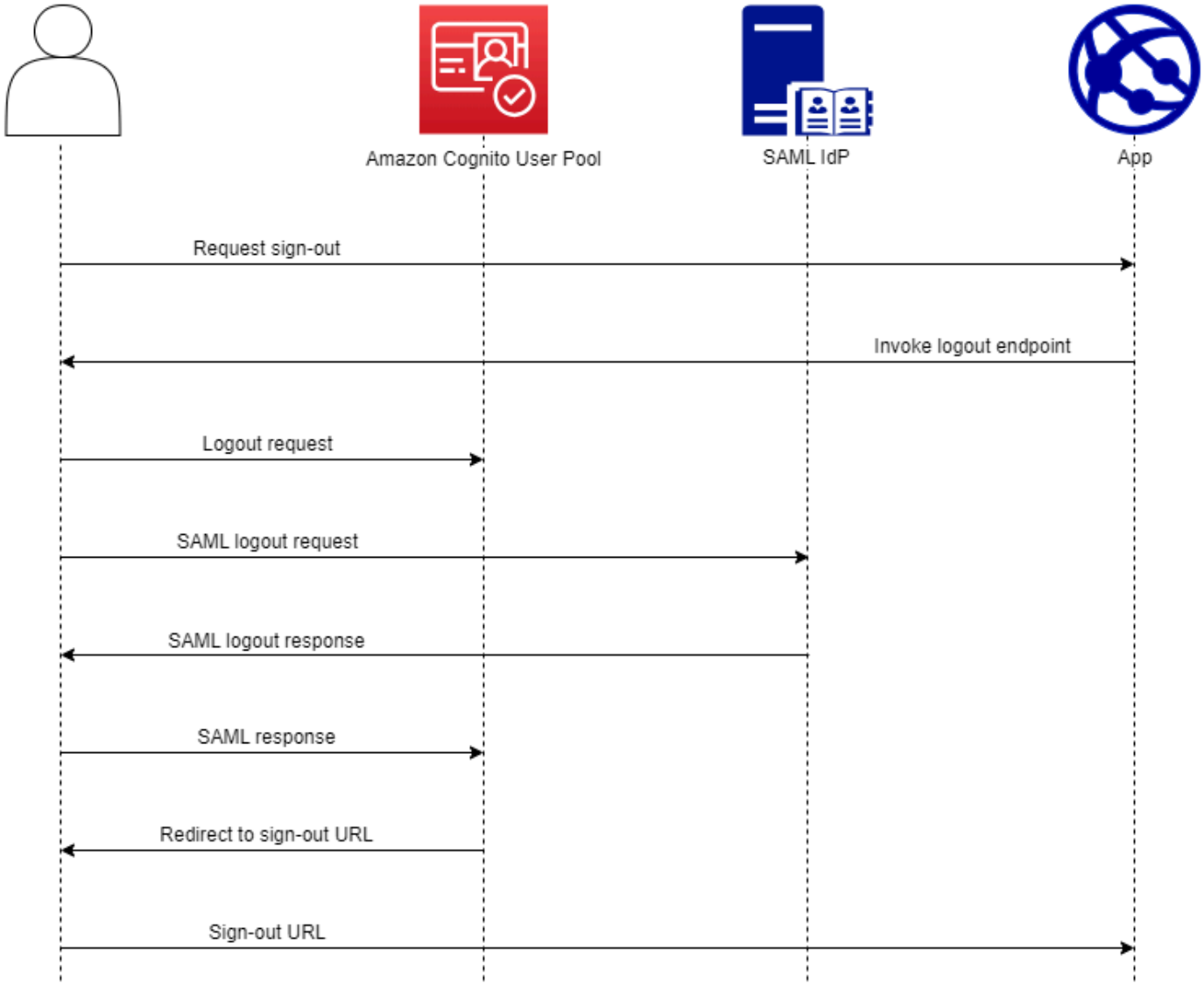
IdP が開始する SAML に IdP を設定するには

CreateIdentityProvider または UpdateIdentityProvider API リクエストの `IDPInit`パラメータを使用して IdP 開始 SAML を設定します。以下は、IdP が開始する SAML をサポートする IdP `ProviderDetails`の例です。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML サインアウトフロー

Amazon Cognito は SAML 2.0 シングルログアウト をサポートしています。サインアウトフロー をサポートするように SAML IdP を設定すると、Amazon Cognito は署名付き SAML ログアウトリクエストを使用してユーザーを IdP にリダイレクトします。Amazon Cognito は、IdP メタデータの `SingleLogoutService` URL からリダイレクト場所を決定します。Amazon Cognito は、ユーザープール署名証明書を使用してサインアウトリクエストに署名します。



SAML セッションを持つユーザーをユーザープール/logoutエンドポイントに誘導すると、Amazon Cognito は IdP メタデータで指定された SLO エンドポイントに次のリクエストで SAML ユーザーをリダイレクトします。

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

その後、ユーザーは IdP LogoutResponseから を使用してsaml2/logoutエンドポイントに戻ります。IdP は HTTP POSTリクエストLogoutResponseで を送信する必要があります。Amazon Cognito は、最初のサインアウトリクエストからリダイレクト先にリダイレクトします。

SAML プロバイダーは、複数の LogoutResponse を含む AuthnStatement を送信する場合があります。このタイプのレスポンスsessionIndexAuthnStatementの最初の の は、最初にユーザーを認証した sessionIndex SAML レスポンスの と一致する必要があります。sessionIndex が他の にある場合AuthnStatement、Amazon Cognito はセッションを認識しず、ユーザーはサインアウトされません。

AWS Management Console

SAML サインアウトを設定するには

1. ユーザープール 、アプリケーションクライアント 、および SAML IdP を作成します。
2. SAML ID プロバイダーを作成または編集するとき、ID プロバイダー情報 で、「サインアウトフローを追加」というタイトルのチェックボックスをオンにします。
3. ユーザープールの「サインインエクスペリエンス」タブの「フェデレーティッド ID プロバイダーのサインイン」で、IdP を選択し、署名証明書 を見つけます。
4. .crt としてダウンロード を選択します。
5. SAML シングルログアウトとリクエスト署名をサポートするように SAML プロバイダーを設定し、ユーザープール署名証明書をアップロードします。IdP はユーザープールドメイン/saml2/logoutの にリダイレクトする必要があります。

API/CLI

SAML サインアウトを設定するには

CreateIdentityProvider または UpdateIdentityProvider API リクエストの IDPSignoutパラメータを使用して単一のログアウトを設定します。以下は、SAML シングルログアウトをサポートする IdP ProviderDetailsの例です。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
```

```
}
```

SAML 署名と暗号化

Amazon Cognito は、サインインとサインアウトの署名付き SAML リクエストと暗号化された SAML レスポンスをサポートしています。ユーザープールの SAML オペレーション中のすべての暗号化オペレーションは、Amazon Cognito が生成する user-pool-provided キーを使用して署名と暗号文を生成する必要があります。現在、外部キーを使用してリクエストに署名したり、暗号化されたアサーションを受け入れるようにユーザープールを設定することはできません。

> **Note**
>
> ユーザープール証明書は 10 年間有効です。Amazon Cognito は 1 年に 1 回、ユーザープールの新しい署名証明書と暗号化証明書を生成します。Amazon Cognito は、署名証明書をリクエストするときに最新の証明書を返し、最新の署名証明書を使用してリクエストに署名します。IdP は、有効期限が切れていないユーザープール暗号化証明書を使用して SAML アサーションを暗号化できます。以前の証明書は、その期間を通して引き続き有効です。ベストプラクティスとして、プロバイダー設定の証明書を毎年更新します。

トピック

- IdP からの暗号化された SAML レスポンスの受け入れ
- SAML リクエストの署名

IdP からの暗号化された SAML レスポンスの受け入れ

Amazon Cognito と IdP は、ユーザーがサインインおよびサインアウトするときに SAML レスポンスで機密性を確立できます。Amazon Cognito は、パブリック/プライベート RSA キーペアと証明書を、ユーザープールで設定した各外部 SAML プロバイダーに割り当てます。ユーザープール SAML プロバイダーのレスポンス暗号化を有効にする場合は、暗号化された SAML レスポンスをサポートする IdP に証明書をアップロードする必要があります。IdP が指定されたキーですべての SAML アサーションの暗号化を開始する前に、SAML IdP へのユーザープール接続が機能しません。

以下は、暗号化された SAML サインインのフローの概要です。

1. ユーザーがサインインを開始し、SAML IdP を選択します。

2. ユーザープールは、SAML サインインリクエストを使用してユーザーを SAML IdP に認可エンドポイントリダイレクトします。ユーザープールは、必要に応じて、IdP による整合性の検証を可能にする署名でこのリクエストに付随できます。SAML リクエストに署名する場合は、ユーザープールが署名証明書のパブリックキーで署名したリクエストを受け入れるように IdP を設定する必要があります。
3. SAML IdP はユーザーにサインインし、SAML レスポンスを生成します。IdP はパブリックキーを使用してレスポンスを暗号化し、ユーザーをユーザープール`/saml2/idpresponse`エンドポイントにリダイレクトします。IdP は、SAML 2.0 仕様で定義されているようにレスポンスを暗号化する必要があります。詳細については、「OASIS Security `Element <EncryptedAssertion>` Assertion Markup Language (SAML) V2.0 のアサーションとプロトコル」の「」を参照してください。 V2
4. ユーザープールは、プライベートキーを使用して SAML レスポンスの暗号文を復号し、ユーザーにサインインします。

> **Important**
>
> ユーザープールで SAML IdP のレスポンス暗号化を有効にする場合、IdP はプロバイダーに固有のパブリックキーを使用してすべてのレスポンスを暗号化する必要があります。Amazon Cognito は、暗号化をサポートするために設定した SAML 外部 IdP からの暗号化されていない SAML レスポンスを受け入れません。

ユーザープール内の外部 SAML IdP はレスポンスの暗号化をサポートでき、各 IdP は独自のキーペアを受け取ります。

AWS Management Console

SAML レスポンスの暗号化を設定するには

1. ユーザープール 、アプリケーションクライアント 、および SAML IdP を作成します。
2. SAML ID プロバイダーを作成または編集するとき、リクエストに署名してレスポンスを暗号化する で、このプロバイダーからの暗号化された SAML アサーションを要求する というタイトルのチェックボックスをオンにします。
3. ユーザープールの「サインインエクスペリエンス」タブの「フェデレーティッド ID プロバイダーのサインイン」で、SAML IdP を選択し、「暗号化証明書の表示」を選択します。

4. .crt としてダウンロードを選択し、ダウンロードしたファイルを SAML IdP に提供します。証明書の キーを使用して SAML レスポンスを暗号化するように SAML IdP を設定します。

API/CLI

SAML レスポンスの暗号化を設定するには

CreateIdentityProvider または UpdateIdentityProvider API リクエストの EncryptedResponsesパラメータを使用してレスポンスの暗号化を設定します。以下は、リクエスト署名をサポートする IdP ProviderDetailsの例です。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML リクエストの署名

IdP への SAML 2.0 リクエストの整合性を証明できることは、Amazon Cognito SP が開始する SAML サインインのセキュリティ上の利点です。ドメインを持つ各ユーザープールは、ユーザープール X.509 署名証明書を受け取ります。この証明書のパブリックキーを使用すると、ユーザープールは、ユーザーが SAML IdP を選択したときにユーザープールが生成するサインアウトリクエストに暗号化署名を適用します。オプションで、SAML サインインリクエスト に署名するようにアプリケーションクライアントを設定できます。SAML リクエストに署名すると、IdP はリクエストの XML メタデータの署名が、指定したユーザープール証明書のパブリックキーと一致することを確認できます。

AWS Management Console

SAML リクエスト署名を設定するには

1. ユーザープール 、アプリケーションクライアント 、および SAML IdP を作成します。
2. SAML ID プロバイダーを作成または編集するときは、リクエストの署名とレスポンスの暗号化 で、このプロバイダーへの SAML リクエストの署名 というタイトルのチェックボックスをオンにします。

3. ユーザープールのサインインエクスペリエンスタブから、フェデレーティッド ID プロバイダーのサインイン で、署名証明書の表示 を選択します。
4. .crt としてダウンロードを選択し、ダウンロードしたファイルを SAML IdP に提供します。受信 SAML リクエストの署名を検証するように SAML IdP を設定します。

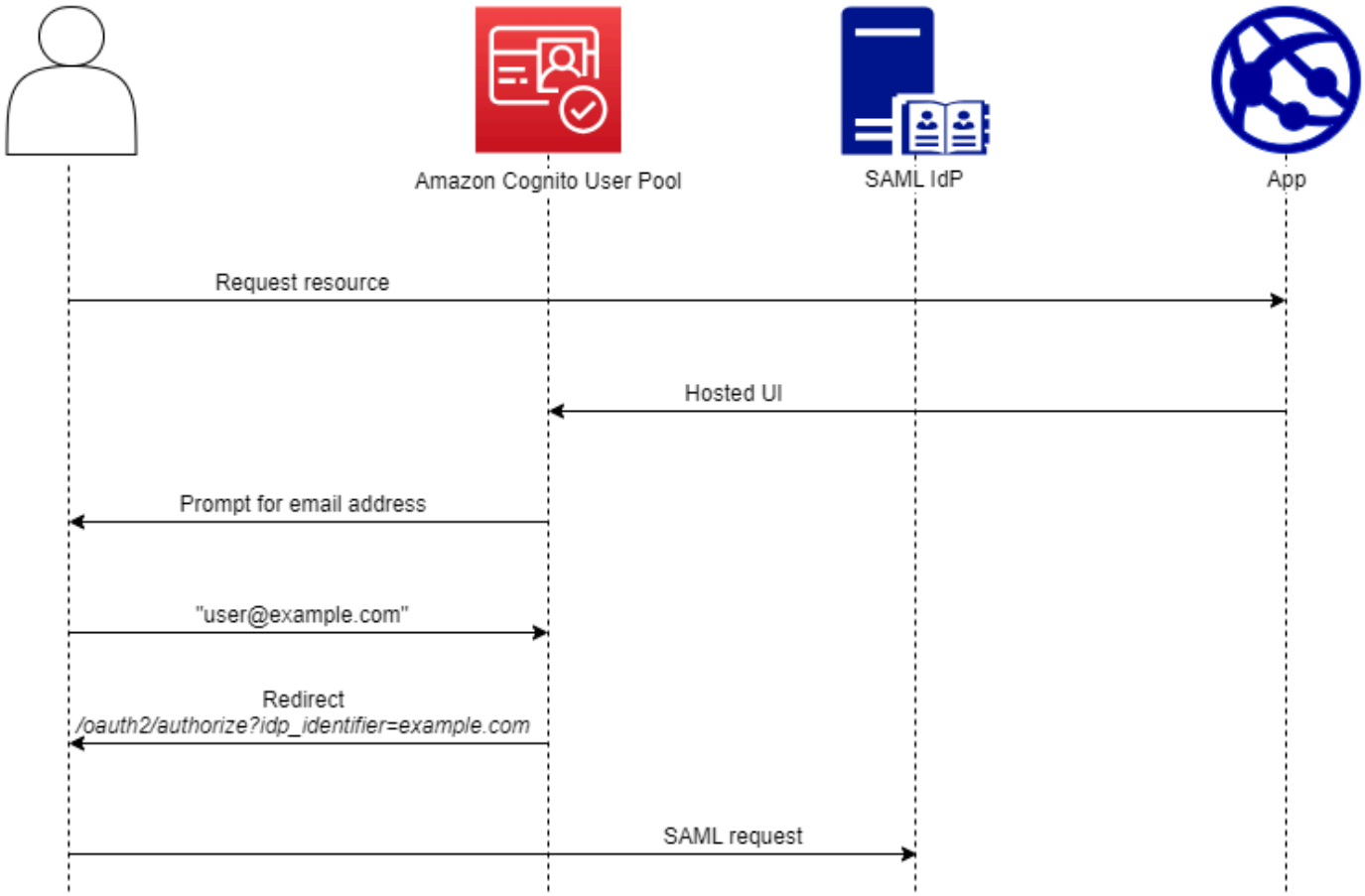
API/CLI

SAML リクエスト署名を設定するには

CreateIdentityProvider または UpdateIdentityProvider API リクエストの `RequestSigningAlgorithm`パラメータを使用してリクエスト署名を設定します。以下は、リクエスト署名をサポートする IdP `ProviderDetails`の例です。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML ID プロバイダーの名前と識別子



SAML ID プロバイダー (IdPs) に名前を付け、IdP 識別子を割り当てると、そのプロバイダーへの SP 開始のサインインおよびサインアウトリクエストのフローを自動化できます。プロバイダー名に対する文字列制約の詳細については、「」の `ProviderName`プロパティを参照してくださいCreateIdentityProvider。

SAML プロバイダーには、最大 50 個の識別子を選択することもできます。識別子は、ユーザープール内の IdP のわかりやすい名前であり、ユーザープール内で一意である必要があります。SAML 識別子がユーザーの E メールドメインと一致する場合、Amazon Cognito がホストする UI は各ユーザーの E メールアドレスをリクエストし、E メールアドレス内のドメインを評価し、ドメインに対応する IdP にリダイレクトします。同じ組織が複数のドメインを所有できるため、1 つの IdP に複数の識別子を含めることができます。

E メールドメイン識別子を使用するか使用しないかにかかわらず、マルチテナントアプリケーションの識別子を使用してユーザーを正しい IdP にリダイレクトできます。ホストされた UI を完全にバイパスする場合は、ユーザーに提示するリンクをカスタマイズして、ユーザーが を介して IdP 認可エ

ンドポイントに直接リダイレクトするようにできます。ユーザーに 識別子でサインインし、IdP にリダイレクトするには、最初の認証リクエストのidp_identifier=*myidp.example.com*リクエストパラメータに 形式の識別子を含めます。

ユーザーを IdP に渡すもう 1 つの方法は、パラメータに IdP の名前identity_providerを次の URL 形式で入力することです。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

ユーザーが SAML IdP でサインインすると、IdP はHTTP POST本文の SAML レスポンスを使用して/saml2/idpresponseエンドポイントにリダイレクトします。Amazon Cognito は SAML アサーションを処理し、レスポンスのクレームが期待を満たしている場合は、アプリクライアントのコールバック URL にリダイレクトします。ユーザーがこの方法で認証を完了すると、IdP とアプリのみのウェブページとやり取りします。

ドメイン形式の IdP 識別子を使用すると、Amazon Cognito がホストする UI はサインイン時に E メールアドレスをリクエストし、E メールドメインが IdP 識別子と一致すると、ユーザーは IdP のサインインページにリダイレクトされます。例えば、2 つの異なる会社の従業員によるサインインを必要とするアプリケーションを構築します。最初の会社である AnyCompany A は、exampleA.comと を所有していますexampleA.co.uk。2 番目の会社である AnyCompany B が を所有していますexampleB.com。この例では、次のように、会社ごとに 1 IdPsつずつ、2 つの をセットアップしています。

- IdP A では、識別子 exampleA.com および exampleA.co.uk を定義します。
- IdP B では、識別子 exampleB.com を定義します。

アプリで、アプリクライアントのホストされた UI を呼び出して、各ユーザーに E メールアドレスの入力を求めます。Amazon Cognito は、E メールアドレスからドメインを取得し、ドメインをドメイン識別子と IdP に関連付け、idp_identifierリクエストパラメータ認可エンドポイントを含むへのリクエストを使用してユーザーを正しい IdP にリダイレクトします。例えば、ユーザーが と入力した場合bob@exampleA.co.uk、ユーザーが操作する次のページは の IdP サインインページですhttps://auth.exampleA.co.uk/sso/saml。

同じロジックを個別に実装することもできます。アプリでは、ユーザー入力を収集し、独自のロジックに従って正しい IdP に関連付けるカスタムフォームを構築できます。アプリケーションテナント

ごとにカスタムアプリケーションポータルを生成できます。各アプリケーションポータルは、リクエストパラメータでテナントの識別子を使用して認証エンドポイントにリンクします。

ホストされた UI で E メールアドレスを収集してドメインを解析するには、アプリクライアントに割り当てた各 SAML IdP に少なくとも 1 つの識別子を割り当てます。デフォルトでは、ホストされた UI サインイン画面には、アプリクライアントに割り当て IdPs た各 のボタンが表示されます。ただし、識別子を正常に割り当てた場合、ホストされた UI サインインページは次の図のようになります。

ホストされた UI でドメインを解析するには、IdP 識別子としてドメインを使用する必要があります。アプリクライアントの各 SAML に任意のタイプの識別子を割り当てる IdPs と、そのアプリのホストされた UI には IdP 選択ボタンが表示されなくなります。E メール解析またはカスタムロジックを使用してリダイレクトを生成する場合は、SAML の IdP 識別子を追加します。サイレントリダイレクトを生成し、ホストされた UI に のリストを表示させたい場合は IdPs、識別子を割り当てず、`identity_provider` リクエストパラメータを認証リクエストで使用します。

- アプリクライアントに SAML IdP を 1 つだけ割り当てると、ホストされた UI サインインページにその IdP でサインインするためのボタンが表示されます。
- アプリクライアントに対してアクティブ化したすべての SAML IdP に識別子を割り当てると、ホストされた UI サインインページに E メールアドレスの入力プロンプトが表示されます。
- 複数の があり IdPs 、それらすべてに識別子を割り当てない場合、ホストされた UI サインインページには、割り当てられた各 IdP でサインインするためのボタンが表示されます。
- に識別子を割り当て IdPs 、ホストされた UI に IdP ボタンの選択を表示する場合は、識別子のない新しい IdP をアプリケーションクライアントに追加するか、新しいアプリケーションクライアントを作成します。既存の IdP を削除して、識別子なしで再度追加することもできます。新しい IdP を作成すると、SAML ユーザーが新しいユーザープロファイルを作成します。このアクティブなユーザーの重複は、IdP 設定を変更した月に請求に影響する可能性があります。

IdP セットアップの詳細については、「ユーザープールの ID プロバイダーの設定」を参照してください。

サードパーティーの SAML ID プロバイダーの設定

Amazon Cognito ユーザープールのフェデレーションで動作するようにサードパーティー SAML 2.0 ID プロバイダー (IdP) ソリューションを設定するには、次のアサーションコンシューマーサービス (ACS) URL にリダイレクトするように SAML IdP を設定する必要があります。`https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse`ユーザープー

ルに Amazon Cognito ドメインがある場合、Amazon Cognito コンソールのユーザープールの [App integration] (アプリの統合) タブでユーザープールドメインのパスを見つけることができます。

一部の SAML IdPs ではurn、オーディエンス URI または SP エンティティ ID とも呼ばれる を 形式で指定する必要がありますurn:amazon:cognito:sp:*us-east-1_EXAMPLE*。ユーザープール ID は、Amazon Cognito コンソールのユーザープールの概要にあります。

また、ユーザープールで必須属性として指定した属性の値を指定するように SAML IdP を設定する必要があります。通常、 emailはユーザープールに必要な属性です。この場合、SAML IdP は SAML アサーションで何らかの形式のemailクレームを提供する必要があり、クレームをそのプロバイダーの 属性にマッピングする必要があります。

サードパーティーの SAML 2.0 IdP ソリューションに関する次の設定情報は、Amazon Cognito ユーザープールとのフェデレーションの設定を開始するのに適しています。最新情報については、プロバイダーのドキュメントを直接参照してください。

SAML リクエストに署名するには、ユーザープール署名証明書によって署名されたリクエストを信頼するように IdP を設定する必要があります。暗号化された SAML レスポンスを受け入れるには、ユーザープールへのすべての SAML レスポンスを暗号化するように IdP を設定する必要があります。プロバイダーには、これらの機能の設定に関するドキュメントがあります。Microsoft の例については、「Microsoft Entra SAML トークン暗号化の設定」を参照してください。

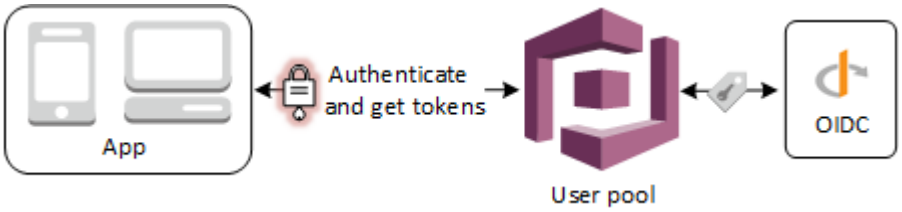
> **Note**
>
> Amazon Cognito では、ID プロバイダーメタデータドキュメントのみが必要です。プロバイダーが SAML 2.0 との AWS アカウント フェデレーションの設定情報を提供している場合があります。この情報は Amazon Cognito の統合には関係ありません。

ソリューション	詳細情報
Microsoft Active Directory フェデレーションサービス (AD FS)	フェデレーションメタデータエクスプローラー
Okta	SAML アプリケーション統合の IdP メタデータと SAML 署名証明書をダウンロードする方法
Auth0	Auth0 を SAML ID プロバイダーとして設定する

ソリューション	詳細情報
Ping アイデンティティ (PingFederate)	からの SAML メタデータのエクスポート PingFederate
JumpCloud	SAML 設定ノート
SecureAuth	SAML アプリケーション統合

ユーザープールでの OIDC ID プロバイダーの使用

OpenID Connect (OIDC) ID プロバイダー (IdPs) のアカウントを既に持っているユーザーは、サインアップステップをスキップして、既存のアカウントを使用してアプリケーションにサインインできます。組み込みの Hosted Web UI では、Amazon Cognito がすべての認証済みユーザーに関するトークンの処理と管理を提供します。このように、バックエンドシステムは 1 セットのユーザープールトークンで標準化できます。



> **Note**
>
> サードパーティー (フェデレーション) 経由のサインインは、Amazon Cognito のユーザープールで使用できます。この機能は、Amazon Cognito ID プール (フェデレーティッド ID) 経由のフェデレーションとは無関係です。

OIDC IdP は、 AWS Management Console、、またはユーザープール API メソッド を使用して AWS CLI、 のユーザープールに追加できますCreateIdentityProvider。

トピック

- 前提条件
- ステップ 1: OIDC IdP に登録する
- ステップ 2: ユーザープールに OIDC IdP を追加する
- ステップ 3: OIDC IdP の設定をテストする

- OIDC ユーザープール IdP 認証フロー

前提条件

開始するには、以下が必要です。

- アプリケーションクライアントとユーザープールドメインを使用するユーザープール。詳細については、「ユーザープールの作成」を参照してください。
- 次の設定を持つ OIDC IdP。
  - `client_secret_post` クライアント認証をサポートします。Amazon Cognito は、IdP の OIDC ディスカバリエンドポイントでの `token_endpoint_auth_methods_supported` クレームをチェックしません。Amazon Cognito は、`client_secret_basic` クライアント認証をサポートしていません。クライアント認証の詳細については、OpenID Connect ドキュメントの「クライアント認証」を参照してください。
  - `openid_configuration`、`userInfo`、および `jwks_uri` などの OIDC エンドポイントにのみ HTTPS を使用します。
  - OIDC エンドポイントには TCP ポート 80 および 443 のみを使用します。
  - HMAC-SHA、ECDSA または RSA アルゴリズムで ID トークンにのみ署名します。
  - キー ID `kid` クレームを `jwks_uri` で発行し、そのトークンに `kid` クレームをフックみます。

ステップ 1: OIDC IdP に登録する

Amazon Cognito で OIDC IdP を作成する前に、アプリケーションを OIDC IdP に登録して、クライアント ID とクライアントシークレットを取得する必要があります。

OIDC IdP に登録する

1. OIDC IdP のデベロッパーアカウントを作成します。

   OIDC へのリンク IdPs

OIDC IdP	インストール方法	OIDC 検出 URL
Salesforce	Salesforce ID プロバイダーのインストール	https://login.salesforce.com

OIDC IdP	インストール方法	OIDC 検出 URL
Ping Identity	Ping Identity ID プロバイダーのインストール	https://*Ping ########*:9031/idp/userinfo.openid 例: `https://pf.company.com:9031/idp/userinfo.openid`
Okta	Okta ID プロバイダーのインストール	https://*Okta ######*.oktapreview.com または `https://`*`Your Okta subdomain`*`.okta.com`
Microsoft Azure Active Directory (Azure AD)	Microsoft Azure AD ID プロバイダーのインストール	`https://login.microsoftonline.com/`*`{tenant}`*`/v2.0`
Google	Google ID プロバイダーのインストール	https://accounts.google.com **Note** Amazon Cognito では、Google を統合されたソーシャルサインイン IdP として提供します。統合された IdP を使用することをお勧めします。「ユーザープールでのソーシャル ID プロバイダーの使用」を参照してください。

2. OIDC IdP を使用して、`/oauth2/idpresponse` エンドポイントを持つユーザープールドメインを登録します。そうすることで、OIDC IdP が後ほど、ユーザーの認証時に Amazon Cognito からこれを受け入れることが確実になります。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

3. コールバック URL を Amazon Cognito ユーザープールに登録します。これは、認証が成功した後で Amazon Cognito がユーザーをリダイレクトするページの URL です。

```
https://www.example.com
```

4. スコープを選択します。スコープ openid が必要です。email および email_verified クレームにアクセスを付与するには、email スコープが必要です。
5. OIDC IdP は、クライアント ID とクライアントシークレットを提供します。これらは、ユーザープールで OIDC IdP を設定するときに使用します。

例: Salesforce を OIDC IdP としてユーザープールで使用する

OIDC 互換 IdP (Salesforce など) とユーザープールの間で信頼性を確立するときに OIDC IdP を使用します。

1. Salesforce 開発者ウェブサイトでアカウントを作成します。
2. 前のステップで設定した開発者アカウントを使用してサインインします。
3. Salesforce ページで、次のいずれかの操作を行います。
   - Lightning Experience を使用している場合は、歯車アイコンを選択してから、[Setup Home] (ホームの設定) を選択します。
   - Salesforce Classic を使用しており、ユーザーインターフェイスのヘッダーに [Setup (設定)] が表示されている場合は、選択します。
   - Salesforce Classic を使用しており、ヘッダーに [Setup (設定)] が表示されていない場合は、上部のナビゲーションバーで名前を選択し、ドロップダウンリストより [Setup (設定)] を選択します。
4. 左のナビゲーションバーで、[Company Settings (組織の設定)] を選択します。
5. ナビゲーションバーで、[Domain] (ドメイン) を選択してドメインを入力し、[Create] (作成) を入力します。
6. 左のナビゲーションバーで、[Platform Tools] (プラットフォームツール) に移動し、[Apps] (アプリケーション) を選択します。
7. [アプリケーションマネージャ] を選択します。
8. a. [New connected app] (新規接続アプリケーション) を選択します。

b. 必須フィールドに入力します。

[Start URL] (開始 URL) で、Salesforce IdP でサインインするユーザープールドメインの `/authorize` エンドポイントの URL を入力します。ユーザーが接続アプリケーションにアクセスすると、Salesforce はこの URL に誘導してサインインを完了します。次に、Salesforce はユーザーをアプリクライアントに関連付けたコールバック URL にリダイレクトします。

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://
www.example.com&identity_provider=CorpSalesforce
```

c. OAuth 設定を有効にし、コールバック URL にユーザープールドメインの `/oauth2/idpresponse` エンドポイントの URL を入力します。これは、Amazon Cognito が OAuth トークンと交換する認証コードを Salesforce が発行する URL です。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

9. スコープを選択します。スコープ openid を含める必要があります。email および email_verified クレーム にアクセスを付与するには、email スコープが必要です。スコープはスペースで区切ります。
10. [作成] を選択します。

Salesforce では、クライアント ID は [コンシューマーキー]、クライアントシークレットは [コンシューマーシークレット] と呼ばれます。クライアント ID とクライアントシークレットを書き留めます。これらは次のセクションで使用します。

ステップ 2: ユーザープールに OIDC IdP を追加する

このセクションでは、OIDC IdP からの OIDC ベースの認証リクエストを処理するようにユーザープールを設定します。

OIDC IdP を追加する (Amazon Cognito コンソール)

OIDC IdP を追加する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ナビゲーションメニューから [User Pools] (ユーザープール) を選択します。

3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Federated sign-in] (フェデレーションサインイン) を検索し、[Add an identity provider] (ID プロバイダーの追加) を選択します。
5. [OpenID Connect] IdP を選択します。
6. [Provider name] (プロバイダー名) に一意の名前を入力します。
7. プロバイダーから受け取ったクライアント ID を、[Client ID] (クライアント ID) へ入力します。
8. プロバイダーから受け取ったクライアントシークレットを、[Client secret] (クライアントシークレット) に入力します。
9. このプロバイダーの [Authorized scopes] (承認済みスコープ) を入力します。スコープは、アプリケーションがプロバイダーにリクエストするユーザー属性のグループ (`name` および `email` など) を定義します。OAuth 2.0 仕様に従い、スコープはスペースで区切る必要があります。

   ユーザーはこれらの属性をアプリケーションに提供することに同意を求められます。
10. [Attribute request method] (属性リクエストメソッド) を選択して、プロバイダーが操作する [userInfo] エンドポイントからユーザーの詳細をフェッチするために必要な HTTP メソッド (GET または POST) を Amazon Cognito に提供します。
11. [Setup method] (セットアップ方法) を選択して、OpenID Connect エンドポイントを、[Auto fill through issuer URL] (発行者 URL による自動入力) または [Manual input] (手動入力) で取得します。[Auto fill through issuer URL] (発行者 URL による自動入力) は、Amazon Cognito が `authorization`、`token`、`userInfo`、および `jwks_uri` エンドポイントの URL を取得できるパブリック `.well-known/openid-configuration` エンドポイントを、プロバイダーが持っている場合に使用します。
12. 発行者の URL、または IdP からの `authorization`、`token`、`userInfo`、および `jwks_uri` エンドポイントの URL を入力します。

> **Note**
>
> URL は、`https://` で始める必要があり、末尾にスラッシュ (/) を使用することはできません。この URL では、ポート番号 443 および 80 のみを使用できます。例えば、Salesforce では次の URL を使用します。
> `https://login.salesforce.com`
> 自動入力を選択した場合、検出ドキュメントは `authorization_endpoint`、`token_endpoint`、`userinfo_endpoint`、および

`jwks_uri` の値に HTTPS を使用する必要があります。そうしない場合は、ログインが失敗します。

13. デフォルトで、OIDC クレームの [sub] (サブ) はユーザープール属性の [Username] (ユーザーネーム) にマッピングされます。他の OIDC クレームをユーザープール属性にマッピングできます。OIDC クレームを入力し、対応するユーザープール属性をドロップダウンリストから選択します。例えば、通常、クレームの [email] はユーザープール属性の [E メール] にマッピングされます。
14. IdP からユーザープールに属性をマッピングします。詳細は、「ユーザープール用 ID プロバイダー属性マッピングの特定」を参照してください。
15. [作成] を選択します。
16. [App client integration] (アプリケーションクライアント統合) タブから、[App clients] (アプリケーションクライアント) のリストより 1 つ選択して、[Edit hosted UI settings] (ホストされた UI 設定の編集) をクリックします。[Identity providers] (ID プロバイダー) で、新しい OIDC IdP をアプリケーションクライアントに追加します。
17. [変更を保存] を選択します。

OIDC IdP を追加するには (AWS CLI)

- CreateIdentityProvider API メソッドのパラメータの説明を参照してください。

```
aws cognito-idp create-identity-provider
--user-pool-id string
--provider-name string
--provider-type OIDC
--provider-details map


--attribute-mapping string
--idp-identifiers (list)
--cli-input-json string
--generate-cli-skeleton string
```

このプロバイダー詳細のマップを使用します。

```
{
  "client_id": "string",
  "client_secret": "string",
  "authorize_scopes": "string",
  "attributes_request_method": "string",
  "oidc_issuer": "string",

  "authorize_url": "string",
  "token_url": "string",
  "attributes_url": "string",
  "jwks_uri": "string"
}
```

ステップ 3: OIDC IdP の設定をテストする

前の 2 つのセクションの要素を使用して認証 URL を作成し、OIDC IdP の設定をテストできます。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

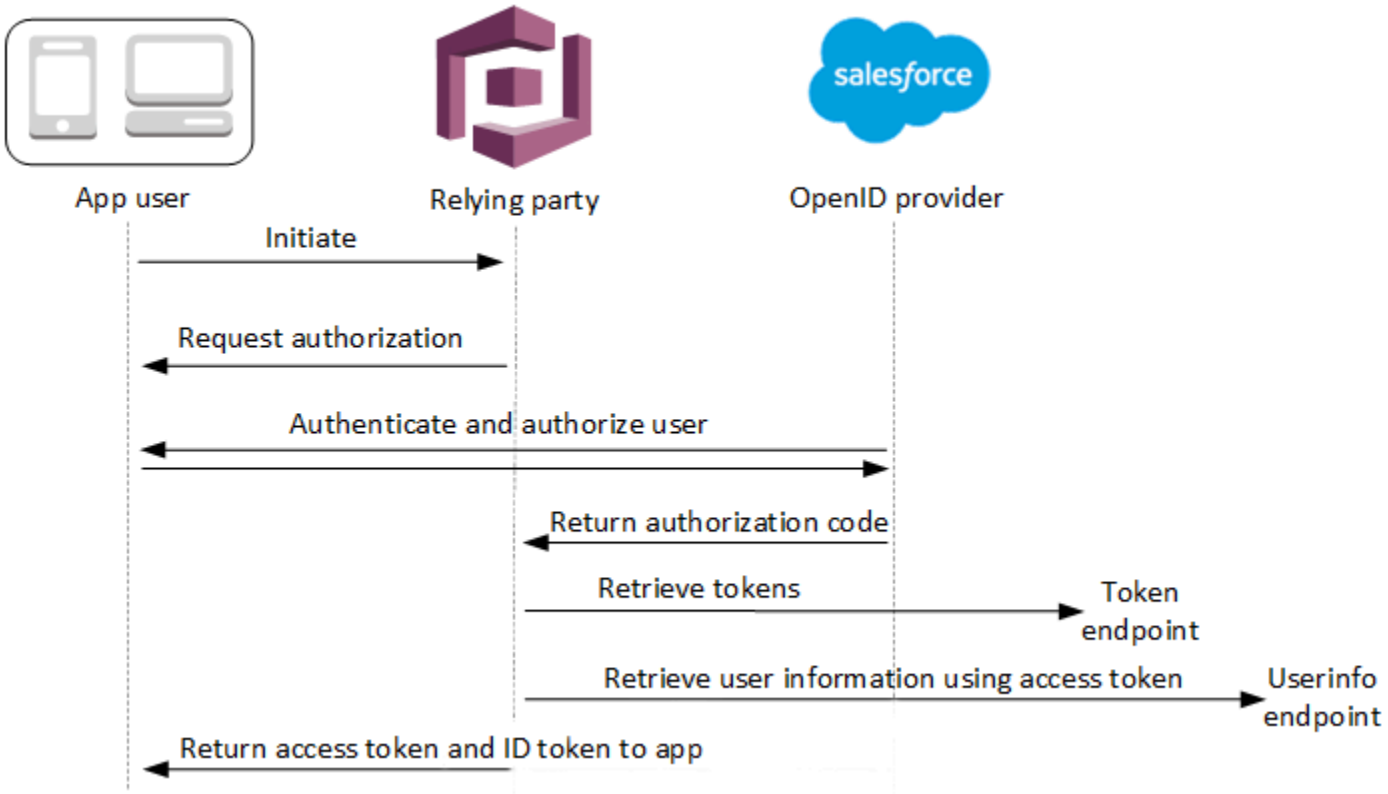
ドメインは、コンソールにあるユーザープールの [ドメイン名] ページで確認できます。client_id は [全般設定] ページにあります。redirect_uri パラメータのコールバック URL を使用します。これは、ユーザーが認証に成功した後でリダイレクトされるページの URL です。

OIDC ユーザープール IdP 認証フロー

ユーザーが OIDC IdP を使用してアプリケーションにサインインすると、次の認証フローを通過します。

1. ユーザーが Amazon Cognito の組み込みサインインページにアクセスすると、Salesforce などの OIDC IdP 経由でサインインするオプションが提示されます。
2. ユーザーは、OIDC IdP の `authorization` エンドポイントにリダイレクトされます。
3. ユーザーが認証されると、OIDC IdP が認証コードを使用して Amazon Cognito にリダイレクトします。
4. Amazon Cognito が OIDC IdP の認証コードをアクセストークンと交換します。

5. Amazon Cognito がユーザープールのユーザーアカウントを作成または更新します。
6. Amazon Cognito がアプリケーションのベアラートークンを発行します。これには ID トークン、アクセストークン、および更新トークンが含まれる場合があります。



> **Note**
>
> Amazon Cognito は 5 分以内に完了しない認証リクエストをキャンセルし、ホストされた UI にユーザーをリダイレクトします。ページには、`Something went wrong` というエラーメッセージが表示されます。

OIDC は OAuth 2.0 上の ID レイヤーで、 が OIDC クライアントアプリケーション (関係当事者) IdPs に発行する JSON 形式の (JWT) ID トークンを指定します。Amazon Cognito の OIDC Relying Party としての追加に関する情報は、OIDC IdP のドキュメントを参照してください。

ユーザーが認証コード付与を使用して認証すると、ユーザープールは ID、アクセス、更新トークンを返します。ID トークンは ID 管理用の OIDC 標準トークンです。アクセストークンは OAuth 2.0 標

準トークンです。ユーザープールのアプリケーションクライアントがサポートできる付与タイプの詳細については、「認可エンドポイント」を参照してください。

ユーザープールが OIDC プロバイダーからのクレームを処理する方法

ユーザーがサードパーティーの OIDC プロバイダーへのサインインを完了すると、Amazon Cognito がホストする UI は IdP から認証コードを取得します。ユーザープールは、アクセストークンと ID トークンの認証コードを IdP の `token` エンドポイントと交換します。ユーザープールはこれらのトークンをユーザーやアプリに渡すのではなく、これらを使って独自のトークンのクレームで提示するデータを使用してユーザープロファイルを構築します。

Amazon Cognito はアクセストークンを個別に検証しません。代わりに、`userInfo`プロバイダーのエンドポイントにユーザー属性情報をリクエストし、トークンが有効でない場合はリクエストが拒否されることを想定しています。

Amazon Cognito は、以下のチェックを行ってプロバイダー ID トークンを検証します。

1. プロバイダーが RSA、HMAC、Elliptic Curve というセットのアルゴリズムを使用してトークンに署名したことをチェックします。
2. プロバイダーが非対称署名アルゴリズムを使用してトークンに署名した場合は、トークンの `kid` クレームの署名キー ID がプロバイダーの `jwks_uri` エンドポイントに表示されていることを確認します。
3. ID トークンの署名を、プロバイダーのメタデータに基づいて想定される署名と比較します。
4. `iss` クレームを IdP に設定された OIDC 発行者と比較します。
5. `aud` クレームが IdP で設定されているクライアント ID と一致するか、または `aud` クレームに複数の値がある場合は設定されたクライアント ID が含まれているかを比較します。
6. `exp` クレームのタイムスタンプが現在の時刻より前でないことをチェックします。

ユーザープールは ID トークンを検証し、プロバイダーアクセストークンを使用してプロバイダーの `userInfo` エンドポイントへのリクエストを試みます。アクセストークンのスコープによって読み取りが許可されたユーザープロファイル情報がすべて取得されます。次にユーザープールはユーザープールの要求に従って設定したユーザー属性を検索します。プロバイダー設定で、必要な属性の属性マッピングを作成する必要があります。ユーザープールはプロバイダー ID トークンと `userInfo` レスポンスをチェックします。ユーザープールは、マッピングルールに一致するすべてのクレームをユーザープールのユーザープロファイルのユーザー属性に書き込みます。ユーザープールは、マッピングルールに一致するが必須ではなく、プロバイダーのクレームにも含まれていない属性を無視します。

ユーザープールの ID プロバイダー属性マッピングを指定する

AWS Management Console、、または AWS CLI API を使用して、ユーザープールの ID プロバイダー (IdP) の属性マッピングを指定できます。

マッピングについて知っておくべきこと

ユーザー属性マッピングの設定を開始する前に、以下の重要な詳細を確認してください。

- フェデレーションユーザーがアプリケーションにサインインする際、マッピングは、ユーザープールが必要とする各ユーザープール属性に存在している必要があります。例えば、サインアップするためにユーザープールで `email` 属性が必要となる場合には、この属性を IdP から該当するものにマッピングします。
- デフォルトで、マップされた E メールアドレスは未検証です。ワンタイムコードを使用してマップされた E メールアドレスを検証することはできません。その代わりに IdP からの属性をマップして、検証ステータスを取得します。例えば、Google とほとんどの OIDC プロバイダーには `email_verified` 属性が含まれています。
- ID プロバイダー (IdP) のトークンをユーザープールのカスタム属性にマッピングできます。ソーシャルプロバイダーはアクセストークンを提示し、OIDC プロバイダーはアクセストークンと ID トークンを提示します。トークンをマッピングするには、最大長さ 2,048 文字のカスタム属性を追加し、アプリクライアントにその属性への書き込みアクセス許可を付与し、IdP からの `access_token` または `id_token` をカスタム属性にマッピングします。
- マッピングされた各ユーザープールの属性では、値の最大長 さである 2048 文字が、Amazon Cognito が IdP から取得する値に対して十分な長さである必要があります。そうではない場合は、ユーザーがアプリケーションにサインインするときに Amazon Cognito がエラーを報告します。Amazon Cognito は、トークンの長さが 2,048 文字を超える場合、IdP トークンのカスタム属性へのマッピングをサポートしていません。
- Amazon Cognito は、次の表に示すように、フェデレーティッド IdP が渡す特定のクレームからフェデレーティッドユーザーのプロファイルの `username` 属性を取得します。Amazon Cognito は、この属性値の前に IdP の名前、例えば を付加しますMyOIDCIdP_[sub]。フェデレーティッドユーザーに外部ユーザーディレクトリの属性と完全に一致する属性を持たせたい場合は、その属性を のような Amazon Cognito サインイン属性にマッピングしますpreferred_username。

ID プロバイダー	**username** ソース属性
Facebook	`id`

ID プロバイダー	**username** ソース属性
Google	`sub`
Login with Amazon	`user_id`
Apple でのサインイン	`sub`
SAML プロバイダー	`NameID`
OpenID Connect (OIDC) プロバイダー	`sub`

- Amazon Cognito は、ユーザーがアプリケーションにサインインするときにマップされたユーザープール属性を更新できる必要があります。ユーザーが IdP 経由でサインインすると、Amazon Cognito は IdP からの最新情報でマップされた属性を更新します。Amazon Cognito は、マップされた属性の現行の値が最新情報とすでに一致している場合でも、各属性を更新します。Amazon Cognito が属性を更新できることを確実にするため、以下の要件をチェックしてください。
  - IdP からマッピングするすべてのユーザープールのカスタム属性は、変更可能である必要があります。変更可能なカスタム属性はいつでも更新できます。対照的に、ユーザーの変更不可なカスタム属性の値を設定できるのは、ユーザープロファイルを最初に作成するときのみです。Amazon Cognito コンソールで変更可能なカスタム属性を作成するには、[Sign-up experience] (サインアップエクスペリエンス) タブの [Add custom attributes] (カスタム属性の追加) を選択時に、追加する属性の [Mutable] (変更可能) チェックボックスを有効にします。または、 CreateUserPool API オペレーションを使用してユーザープールを作成する場合は、これらの各属性の `Mutable`パラメータを に設定することもできます`true`。IdP がマッピングされたイミュータブル属性の値を送信すると、Amazon Cognito はエラーを返し、サインインは失敗します。
  - アプリケーションのアプリクライアントの設定では、マッピングされた属性を書き込み可能にする必要があります。書き込み可能な属性は、Amazon Cognito コンソールの [App clients] (アプリクライアント) ページで設定できます。また、`CreateUserPoolClient` API を使用してアプリクライアントを作成する場合、これらの属性を `WriteAttributes` 配列に追加できます。IdP がマッピングされた書き込み不可能な属性の値を送信する場合、Amazon Cognito は属性値を設定せず、認証に進みます。
- IdP 属性に複数の値が含まれている場合、Amazon Cognito はすべての値をカンマで区切られた単一の文字列にフラット化し、URL は英数字以外の文字 (.「」、「」、-*「」、「」文字を除く) _を含む値を形式エンコードします。値は、アプリでの使用前にデコードおよび解析しておく必要があります。

ユーザープールの ID プロバイダー属性マッピングを指定する (AWS Management Console)

を使用して AWS Management Console 、ユーザープールの IdP の属性マッピングを指定できます。

> **Note**
>
> Amazon Cognito は、着信トークンに着信クレームが存在する場合に限り、そのクレームをユーザープール属性にマップします。以前にマッピングされたクレームが着信トークンに存在しなくなった場合、そのクレームは削除または変更されません。削除されたクレームのマッピングがアプリケーションで必要な場合は、事前認証 Lambda トリガーを使用して認証中にカスタム属性を削除し、それらの属性を着信トークンから再入力できます。

ソーシャル IdP 属性マッピングを指定するには

1. Amazon Cognito コンソールにサインインします。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [Sign-in experience] (サインインエクスペリエンス) タブを選択し、[Federated sign-in] (フェデレーションサインイン) を検索します。
4. [Add an identity provider] (ID プロバイダーの追加) を選択するか、設定した [Facebook]、[Google]、[Amazon]、もしくは [Apple] の IdP を選択します。[Attribute mapping] (属性マッピング) を検索し、 [Edit] (編集) を選択します。

   ソーシャル IdP の追加の詳細については、「ユーザープールでのソーシャル ID プロバイダーの使用」を参照してください。
5. マッピングする必要のある各属性について、以下のステップを完了してください。

   a. [User pool attribute] (ユーザープール属性) 列から属性を選択します。これは、ユーザープール内のユーザープロファイルに割り当てられる属性です。カスタム属性は、標準属性の後にリストされます。

   b. [***<provider>*** attribute] (<プロバイダー> 属性) 列から属性を選択します。これは、プロバイダーディレクトリから渡される属性になります。ソーシャルプロバイダーの既知の属性は、ドロップダウンリストに表示されます。

c. IdP と Amazon Cognito の間に追加の属性をマッピングするには、[Add another attribute] (その他の属性を追加) を選択します。

6. [変更を保存] を選択します。

SAML プロバイダー属性マッピングを指定する

1. Amazon Cognito コンソールにサインインします。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [Sign-in experience] (サインインエクスペリエンス) タブを選択し、[Federated sign-in] (フェデレーションサインイン) を検索します。
4. [Add an identity provider] (ID プロバイダーの追加) を選択するか、設定した SAML IdP を選択します。[Attribute mapping] (属性マッピング) を探し、[Edit] (編集) を選択します。SAML IdP の詳細については、「ユーザープールでの SAML ID プロバイダーの使用」を参照してください。
5. マッピングする必要のある各属性について、以下のステップを完了してください。

   a. [User pool attribute] (ユーザープール属性) 列から属性を選択します。これは、ユーザープール内のユーザープロファイルに割り当てられる属性です。カスタム属性は、標準属性の後にリストされます。

   b. [SAML attribute] (SAML 属性) 列から属性を選択します。これは、プロバイダーディレクトリから渡される属性になります。

      IdP は、参考として SAML アサーションのサンプルを提供する場合があります。などの単純な名前 IdPs を使用するものもあればemail、次のような URL 形式の属性名を使用するものもあります。

      ```
      http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
      ```

   c. IdP と Amazon Cognito の間に追加の属性をマッピングするには、[Add another attribute] (その他の属性を追加) を選択します。

6. [変更を保存] を選択します。

ユーザープール (AWS CLI および AWS API) の ID プロバイダー属性マッピングの指定

以下のコマンドを使用して、ユーザープールの IdP 属性マッピングを指定します。

プロバイダーの作成時に属性マッピングを指定する

- AWS CLI: `aws cognito-idp create-identity-provider`

  メタデータファイルの例: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  ここでは `details.json` に以下が含まれます。

  ```
  {
      "MetadataFile": "<SAML metadata XML>"
  }
  ```

  > **Note**
  >
  > *<SAML ##### XML>* に引用符が (") 含まれる場合は、エスケープ (\") する必要があります。

  メタデータ URL の例 :

  ```
  aws cognito-idp create-identity-provider \
  --user-pool-id us-east-1_EXAMPLE \
  --provider-name=SAML_provider_1 \
  --provider-type SAML \
  --provider-details MetadataURL=https://myidp.example.com/saml/metadata \
  --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
  emailaddress
  ```

- AWS API: CreateIdentityProvider

既存の IdP の属性マッピングを指定するには

- AWS CLI: `aws cognito-idp update-identity-provider`

  例: `aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API: UpdateIdentityProvider

特定の IdP の属性マッピングに関する情報を取得するには

- AWS CLI: `aws cognito-idp describe-identity-provider`

  例: `aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`
- AWS API: DescribeIdentityProvider

フェデレーションユーザーを既存のユーザープロファイルにリンクする

多くの場合、同じユーザーには、ユーザープールに接続した複数の ID プロバイダー (IdPs) を持つプロファイルがあります。Amazon Cognito では、ユーザーが出現するたびにディレクトリ内の同じユーザープロファイルにリンクできます。このようにして、複数の IdP ユーザーを持つ 1 人のユーザーが、アプリで一貫したエクスペリエンスを持つことができます。AdminLinkProviderForUser は、フェデレーティッドディレクトリ内のユーザーの一意の ID をユーザープール内のユーザーとして認識するように Amazon Cognito に指示します。ユーザープール内のユーザーは、ユーザープロファイルに関連付けられているフェデレーション ID が 0 個以上ある場合、課金の目的で 1 人の月間アクティブユーザー (MAU) としてカウントされます。

フェデレーションユーザーが初めてユーザープールにサインインすると、Amazon Cognito はその ID にリンクされたローカルプロファイルを探します。リンクされたプロファイルが存在しない場合、ユーザープールは新しいプロファイルを作成します。ローカルプロファイルを作成し、計画された事前ステージングタスクまたは で、AdminLinkProviderForUserAPI リクエストで、最初のサインイン前であればいつでもフェデレーティッドユーザーにリンクできますサインアップ前の Lambda トリガー。ユーザーがサインインし、Amazon Cognito がリンクされたローカルプロファイルを検出すると、ユーザープールはユーザーのクレームを読み取り、IdP のマッピングルールと比較します。次に、ユーザープールは、ログイン時にマッピングされたクレームを反映して、リンクされたローカルプロファイルを更新します。このようにして、アクセスクレームでローカルプロファイルを設

定し、その ID クレームをプロバイダー up-to-date に保持できます。Amazon Cognito により、フェデレーションユーザーとリンクされたプロファイルがマッチングされると、フェデレーションユーザーは常にそのプロファイルにサインインします。次に、ユーザーの他のプロバイダー ID を同じプロファイルにリンクして、ユーザーのアプリエクスペリエンスを一貫させることができます。以前にサインインしたフェデレーションユーザーをリンクするには、まず既存のプロファイルを削除する必要があります。既存のプロファイルは、*`[Provider name]`*`_identifier` という形式で識別できます。例えば `LoginWithAmazon_amzn1.account.AFAEXAMPLE` です。作成してサードパーティーのユーザー ID にリンクしたユーザーには、作成したユーザー名と、リンクされた ID の詳細を含む`identities`属性があります。

> **Important**
>
> `AdminLinkProviderForUser` は、外部フェデレーティッド ID を持つユーザーがユーザープール内の既存のユーザーとしてサインインできるようにするため、アプリケーション所有者によって信頼されている外部属性 IdPs とプロバイダー属性でのみ使用することが重要です。

例えば、複数のお客様と共有するアプリを使用するマネージドサービスプロバイダー (MSP) の場合です。お客様はそれぞれ、Active Directory Federation Services (ADFS) を使用してアプリにサインインします。IT 管理者 Carlos は、各お客様のドメインにアカウントを持っています。IdP に関係なく、Carlos がサインインするたびにアプリ管理者として認識されるようにします。

ADFS IdPs は、Carlos の SAML アサーションの`email`クレーム`msp_carlos@example.com`に Carlos の E メールアドレスを Amazon Cognito に提示します。ユーザー名 `Carlos` を使用してユーザープールにユーザーを作成します。次の AWS Command Line Interface (AWS CLI) コマンドは、IdPs ADFS1, ADFS2ADFS3リンクします。

> **Note**
>
> 特定の属性クレームに基づいてユーザーをリンクできます。この機能は OIDC と SAML に固有です IdPs。他のプロバイダータイプでは、固定ソース属性に基づいてリンクする必要があります。詳細については、「」を参照してくださいAdminLinkProviderForUser。ソーシャル IdP をユーザープロファイルにリンクするとき、`ProviderAttributeName` に `Cognito_Subject` を設定する必要があります。`ProviderAttributeValue` は、IdP のユーザーの一意の識別子でなければなりません。

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

ユーザープールのユーザープロファイル Carlos は、次の identities 属性を持つようになりました。

```
[{
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS1",
    "providerType": "SAML",
    "issuer": "http://auth.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS2",
    "providerType": "SAML",
    "issuer": "http://auth2.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS3",
    "providerType": "SAML",
    "issuer": "http://auth3.example.com",
    "primary": false,
    "dateCreated": 111111111111111
```

```
}]
```

フェデレーションユーザーのリンクについて知っておくべきこと

- 各ユーザープロファイルに最大 5 人のフェデレーションユーザーをリンクできます。
- フェデレーションユーザーは、既存のフェデレーションユーザープロファイルまたはローカルユーザーにリンクできます。
- プロバイダーを のユーザープロファイルにリンクすることはできません AWS Management Console。
- ユーザーの ID トークンには、`identities` クレームに関連するすべてのプロバイダーが含まれます。
- AdminSetUserPassword API リクエストで自動的に作成されたフェデレーティッドユーザープロファイルのパスワードを設定できます。その後、そのユーザーのステータスは `EXTERNAL_PROVIDER` から `CONFIRMED` に変わります。この状態のユーザーは、フェデレーションユーザーとしてサインインし、リンクされたローカルユーザーのように API で認証フローを開始できます。また、 や などのトークン認証された API リクエストのパスワードChangePasswordと属性を変更することもできますUpdateUserAttributes。セキュリティのベストプラクティスとして、またユーザーを外部 IdP と同期した状態を維持するため、フェデレーションユーザープロファイルにパスワードを設定しないでください。代わりに、`AdminLinkProviderForUser` でユーザーをローカルプロファイルにリンクしてください。
- Amazon Cognito は、ユーザーが IdP からサインインするときに、リンクされたローカルユーザープロファイルにユーザー属性を入力します。Amazon Cognito は OIDC IdP からの ID トークン内のアイデンティティ要求を処理し、OAuth 2.0 プロバイダーと OIDC プロバイダー両方の `userInfo` エンドポイントもチェックします。Amazon Cognito では、ID トークン内の情報は `userInfo` からの情報よりも優先されます。

プロファイルにリンクした外部ユーザーアカウントをユーザーが使用しなくなった場合は、ユーザーアカウントとユーザープールユーザーとの関連付けを解除できます。ユーザーをリンクしたときに、ユーザーの属性名、属性値、プロバイダー名をリクエストに入力しました。ユーザーが不要になったプロファイルを削除するには、同等のパラメータを使用して AdminDisableProviderForUser API リクエストを行います。

SDK のその他のコマンド構文と例AdminLinkProviderForUserについては、「」を参照してください。 AWS SDKs

Lambda トリガーを使用したユーザープールワークフローのカスタマイズ

Amazon Cognito は AWS Lambda 関数を使用してユーザープールの認証動作を変更します。初回サインアップ前、認証完了後、およびその間のいくつかの段階で Lambda 関数を自動的に呼び出すようにユーザープールを設定できます。関数は、認証フローのデフォルト動作を変更したり、ユーザープールやその他の AWS リソースを変更するための API リクエストを行ったり、外部システムと通信したりできます。Lambda 関数のコードはユーザー独自のものです。Amazon Cognito はイベントデータを関数に送信し、関数がデータを処理するのを待ちます。ほとんどの場合、セッションへの変更を示すレスポンスイベントが返されます。

リクエストイベントとレスポンスイベントのシステム内では、独自の認証チャレンジの導入、ユーザープールと別の ID ストア間でのユーザーの移行、メッセージのカスタマイズ、JSON ウェブトークン (JWT) の変更を行うことができます。

Lambda トリガーは、ユーザーがユーザープールでアクションを開始した後に Amazon Cognito がユーザーに配信するレスポンスをカスタマイズできます。例えば、他の方法では成功するはずのユーザーによるサインインを禁止できます。また、AWS 環境、外部 API、データベース、または ID ストアに対してランタイムオペレーションを実行することもできます。例えば、ユーザー移行のトリガーでは、外部アクションと Amazon Cognito の変更を組み合わせることができます。つまり、外部ディレクトリでユーザー情報を検索し、その外部情報に基づいて新しいユーザーに属性を設定できます。

Lambda トリガーをユーザープールに割り当てると、Amazon Cognito はデフォルトのフローを中断して関数から情報をリクエストします。Amazon Cognito は JSON イベントを生成し、それを関数に渡します。イベントには、ユーザーアカウントの作成、サインイン、パスワードのリセット、または属性の更新を求めるユーザーのリクエストに関する情報が含まれます。これで、関数がアクションを実行、またはイベントを変更せずに送り返すことができます。

次の表は、Lambda トリガーを使用してユーザープールオペレーションをカスタマイズする方法をまとめたものです。

ユーザープールフロー	オペレーション	説明
カスタム認証フロー	認証チャレンジの定義	カスタム認証フローでの次のチャレンジを決定する
	認証チャレンジの作成	カスタム認証フローでのチャレンジを作成する

ユーザープールフロー	オペレーション	説明
	認証チャレンジレスポンスの確認	カスタム認証フローでのレスポンスが正しいかどうかを判断する
認証イベント	the section called “認証前の Lambda トリガー”	サインインリクエストを承認または拒否するカスタム検証
	the section called “認証後の Lambda トリガー”	カスタム分析用のイベントをログに記録する
	the section called “トークン生成前の Lambda トリガー”	トークンの主張を強化または制限する
サインアップ	the section called “サインアップ前の Lambda トリガー”	サインアップリクエストを承認または拒否するカスタム検証を実行する
	the section called “確認後の Lambda トリガー”	カスタム分析用のカスタムウェルカムメッセージまたはイベントログ記録を作成する
	the section called “ユーザー移行の Lambda トリガー”	既存のユーザーディレクトリからユーザープールにユーザーを移行する
メッセージ	the section called “カスタムメッセージの Lambda トリガー”	メッセージの高度なカスタマイズとローカライズを実行する
トークンの作成	the section called “トークン生成前の Lambda トリガー”	ID トークンの属性を追加または削除する
E メールと SMS のサードパーティープロバイダー	the section called “カスタム送信者の Lambda トリガー”	サードパーティープロバイダーを使用して SMS メッセージと E メールメッセージを送信する

トピック

重要な考慮事項

Lambda 関数用のユーザープールを準備するときは、以下の点を考慮します。

- Amazon Cognito が Lambda トリガーに送信するイベントは、新しい機能に伴って変わる可能性があります。JSON 階層内のレスポンス要素とリクエスト要素の位置が変更されたり、要素名が追加されたりする場合があります。Lambda 関数では、このガイドで説明している入力要素のキーと値のペアを受け取ることが期待できますが、入力検証がより厳密になると、関数が失敗する可能性があります。
- Amazon Cognito が一部のトリガーに送信する複数のバージョンのイベントから 1 つを選択できます。バージョンによっては、Amazon Cognito 料金の変更を受け入れることが必要になる場合があります。料金の詳細については、「Amazon Cognito の料金」を参照してください。トークン生成前の Lambda トリガー のアクセストークンをカスタマイズするには、アドバンストセキュリティ機能を使用してユーザープールを設定し、イベントバージョン 2 を使用するように Lambda トリガー設定を更新する必要があります。

- カスタム送信者の Lambda トリガーを除き、Amazon Cognito は Lambda 関数を同期的に呼び出します。Amazon Cognito が Lambda 関数を呼び出したら、5 秒以内に応答する必要があります。失敗し、呼び出しを再試行できる場合、Amazon Cognito は呼び出しを再試行します。試行が 3 回失敗すると、関数はタイムアウトします。この 5 秒のタイムアウト値を変更することはできません。詳細については、「AWS Lambda Lambda デベロッパーガイド」の「Lambda プログラミングモデル」を参照してください。

  Amazon Cognito は、HTTP ステータスコード 500～599 の呼び出しエラーを返す関数呼び出しを再試行しません。これらのコードは、Lambda が関数を起動できない原因となる設定の問題を示しています。詳細については、「AWS Lambda でのエラー処理と自動再試行」を参照してください。

- Lambda トリガー設定で関数バージョンを宣言することはできません。Amazon Cognito ユーザープールは、デフォルトで最新バージョンの関数を呼び出します。ただし、CreateUserPool または UpdateUserPool API リクエストにより、関数バージョンをエイリアスに関連付け、トリガーの `LambdaArn` をエイリアス ARN に設定できます。このオプションは、AWS Management Console では使用できません。エイリアスの詳細については、「AWS Lambda デベロッパーガイド」の「Lambda 関数のエイリアス」を参照してください。

- Lambda トリガーを削除する場合は、ユーザープール内の対応するトリガーを更新する必要があります。例えば、認証後トリガーを削除した場合は、ユーザープール内の対応する認証後トリガーを [なし] に設定する必要があります。

- Lambda 関数がリクエストとレスポンスのパラメータを Amazon Cognito に返さないか、エラーを返す場合、認証イベントは成功しません。関数にエラーを返して、ユーザーのサインアップ、認証、トークン生成、または Lambda トリガーを呼び出す認証フローのその他の段階を防ぐことができます。

  Amazon Cognito がホストする UI は、Lambda トリガーが生成するエラーをサインインプロンプトの上にエラーテキストとして返します。Amazon Cognito ユーザープール API は、トリガーエラーを *`[trigger]`* `failed with error` *`[error text from response]`* 形式で返します。ベストプラクティスとして、Lambda 関数では、ユーザーに表示させるエラーのみを生成します。`print()` のような出力方法を使用して、機密情報やデバッグ情報を CloudWatch ログに記録します。例については、「サインアップ前の例:ユーザー名が 5 文字未満の場合にサインアップを拒否する」を参照してください。

- ユーザープールのトリガーとして Lambda 関数を別の AWS アカウント の関数に追加できます。AWS CloudFormation または AWS CLI の CreateUserPool および UpdateUserPool API オペレーション、またはそれに相当するオペレーションにクロスアカウントトリガーを追加する必要があります。AWS Management Console にクロスアカウント機能を追加することはできません。

- Amazon Cognito コンソールに Lambda トリガーを追加すると、Amazon Cognito はユーザープールが関数を呼び出すことを許可するリソースベースのポリシーを、関数に追加します。クロスアカウント関数を含む Lambda トリガーを Amazon Cognito コンソールの外部で作成する場合は、Lambda 関数のリソースベースポリシーにアクセス許可を追加する必要があります。追加したアクセス権限では、Amazon Cognito はユーザープールの代わりに関数を呼び出すことを許可する必要があります。Lambda コンソールから権限を追加、または Lambda AddPermission API オペレーションを使用することができます。

  Lambda リソースベースのポリシーの例

  次の Lambda リソースベースのポリシーの例では、Lambda 関数を呼び出す Amazon Cognito の限定的な機能が付与されます。Amazon Cognito は、`aws:SourceArn` の条件でのユーザープールと `aws:SourceAccount` 条件でのアカウントの両方に代わって関数を呼び出す場合にのみ、関数を呼び出すことができます。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

ユーザープールの Lambda トリガーの追加

コンソールを使用してユーザープールの Lambda トリガーを追加する

1. Lambda コンソールを使用して Lambda 関数を作成します。Lambda 関数の詳細については、AWS Lambda デベロッパーガイドを参照してください。
2. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [] (ユーザープールのプロパティ) タブを選択し、[Lambda triggers] (Lambda トリガー) を検索します。
5. [Add a Lambda trigger] (Lambda トリガーの追加) を選択します。
6. カスタマイズする認証のステージに基づいて、Lambda トリガーカテゴリを選択します。
7. [Assign Lambda function] (Lambda 関数の割り当て) を選択して、AWS リージョン ユーザープールと同じ関数を選択します。

   > **Note**
   >
   > AWS Identity and Access Management (IAM) 認証情報に Lambda 関数を更新する権限がある場合、Amazon Cognito は Lambda リソースベースのポリシーを追加します。このポリシーを使用すると、Amazon Cognito は選択した関数を呼び出すことができます。サインインした認証情報に十分な IAM アクセス権限がない場合は、リソースベースのポリシーを個別に更新する必要があります。詳細については、「the section called “重要な考慮事項”」を参照してください。

8. [Save changes] (変更の保存) をクリックします。
9. Lambda コンソールで CloudWatch を使用して、Lambda 関数をログすることができます。詳細については、「Accessing CloudWatch Logs for Lambda」を参照してください。

ユーザープールの Lambda トリガーイベント

Amazon Cognito は Lambda 関数にイベント情報を渡します。Lambda 関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。以下のイベントは、Lambda トリガーの一般的なパラメータを示しています。

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
                "string": "string",
                ....
            }
        },
    "response": {}
}
```

ユーザープールの Lambda トリガーの一般的なパラメータ

バージョン

Lambda 関数のバージョン番号。

triggerSource

Lambda 関数をトリガーしたイベントの名前。各 triggerSource の説明については、「Lambda トリガーのユーザープールの機能オペレーションへの接続」を参照してください。

region

AWSRegion インスタンスとしての AWS リージョン。

userPoolId

ユーザープールの ID。

userName

現在のユーザーのユーザー名。

callerContext

リクエストとコード環境に関するメタデータ。これには、[awsSdkVersion] と [clientId] フィールドが含まれています。

awsSdkVersion

リクエストを生成した AWS SDK のバージョン。

clientId

ユーザープールアプリクライアントの ID。

request

ユーザーの API リクエストの詳細。以下のフィールドと、トリガーに固有のリクエストパラメータが含まれます。例えば、Amazon Cognito が事前認証トリガーに送信するイベントには `userNotFound` パラメータも含まれます。ユーザーが未登録のユーザー名でサインインしようとしたときに、このパラメータの値を処理してカスタムアクションを実行できます。

userAttributes

ユーザー属性の名前と値の 1 つ以上のキーと値のペア。例: `"email": "john@example.com"`。

レスポンス

このパラメータには、元のリクエストの情報は含まれていません。Lambda 関数はイベント全体を Amazon Cognito に返し、返されるパラメータは `response` に追加する必要があります。関数に含めることができる返されるパラメータを確認するには、使用するトリガーのドキュメントを参照してください。

Lambda トリガーへの API オペレーションの接続

以下のセクションでは、Amazon Cognito がユーザープールのアクティビティから呼び出す Lambda トリガーについて説明します。

アプリが Amazon Cognito ユーザープールの API、ホストされた UI、またはユーザープールエンドポイント を介してユーザーをサインインさせると、Amazon Cognito はセッションコンテキストに基づいて Lambda 関数を呼び出します。Amazon Cognito ユーザープール API およびユーザープールの詳細については、「Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用」を参照してください。次のセクションの表では、Amazon Cognito が関数を呼び出す原因とな

るイベントと、Amazon Cognito がリクエストに含める `triggerSource` 文字列について説明します。

トピック

- Amazon Cognito API の Lambda トリガー
- ホストされた UI の Amazon Cognito ローカルユーザーの Lambda トリガー
- フェデレーティッドユーザーの Lambda トリガー

Amazon Cognito API の Lambda トリガー

次の表は、アプリがローカルユーザーを作成、サインイン、または更新するときに Amazon Cognito が呼び出すことができる Lambda トリガーのソース文字列を示しています。

Amazon Cognito API のローカルユーザートリガーソース

API オペレーション	Lambda トリガー	トリガーソース
AdminCreateUser	サインアップ前	`PreSignUp_AdminCreateUser`
	トークン生成前	`TokenGeneration_NewPasswordChallenge`
	カスタムメッセージ	`CustomMessage_AdminCreateUser`
	カスタム E メール送信者	`CustomEmailSender_AdminCreateUser`
	カスタム SMS 送信者	`CustomSMSSender_AdminCreateUser`
SignUp	サインアップ前	`PreSignUp_SignUp`
	カスタムメッセージ	`CustomMessage_SignUp`
	カスタム E メール送信者	`CustomEmailSender_SignUp`

API オペレーション	Lambda トリガー	トリガーソース
	カスタム SMS 送信者	CustomSMSSender_SignUp
ConfirmSignUp AdminConfirmSignUp	確認後	PostConfirmation_ConfirmSignUp
InitiateAuth AdminInitiateAuth	認証前	PreAuthentication_Authentication
	認証チャレンジの定義	DefineAuthChallenge_Authentication
	認証チャレンジの作成	CreateAuthChallenge_Authentication
	トークン生成前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	ユーザー移行	UserMigration_Authentication
	カスタムメッセージ	CustomMessage_Authentication
	カスタム E メール送信者	CustomEmailSender_AccountTakeOverNotification
	カスタム SMS 送信者	CustomSMSSender_Authentication

API オペレーション	Lambda トリガー	トリガーソース
ForgotPassword	ユーザー移行	UserMigration_ForgotPassword
	カスタムメッセージ	CustomMessage_ForgotPassword
	カスタム E メール送信者	CustomEmailSender_ForgotPassword
	カスタム SMS 送信者	CustomSMSSender_ForgotPassword
ConfirmForgotPassword	確認後	PostConfirmation_ConfirmForgotPassword
UpdateUserAttributes AdminUpdateUserAttributes	カスタムメッセージ	CustomMessage_UpdateUserAttribute
	カスタム E メール送信者	CustomEmailSender_UpdateUserAttribute
	カスタム SMS 送信者	CustomSMSSender_UpdateUserAttribute
VerifyUserAttributes	カスタムメッセージ	CustomMessage_VerifyUserAttribute
	カスタム E メール送信者	CustomEmailSender_VerifyUserAttribute
	カスタム SMS 送信者	CustomSMSSender_VerifyUserAttribute

ホストされた UI の Amazon Cognito ローカルユーザーの Lambda トリガー

次の表は、ローカルユーザーがホストされた UI でユーザープールにサインインするときに Amazon Cognito が呼び出すことができる Lambda トリガーのソース文字列を示しています。

ホストされた UI のローカルユーザートリガーソース

ホストされた UI URI	Lambda トリガー	トリガーソース
/signup	サインアップ前	PreSignUp_SignUp
	カスタムメッセージ	CustomMessage_SignUp
	カスタム E メール送信者	CustomEmailSender_SignUp
	カスタム SMS 送信者	CustomSMSSender_SignUp
/confirmuser	確認後	PostConfirmation_ConfirmSignUp
/login	認証前	PreAuthentication_Authentication
	認証チャレンジの定義	DefineAuthChallenge_Authentication
	認証チャレンジの作成	CreateAuthChallenge_Authentication
	トークン生成前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens

ホストされた UI URI	Lambda トリガー	トリガーソース
	ユーザー移行	UserMigration_Authentication
	カスタムメッセージ	CustomMessage_Authentication
	カスタム E メール送信者	CustomEmailSender_AccountTakeOverNotification
	カスタム SMS 送信者	CustomSMSSender_Authentication
/forgotpassword	ユーザー移行	UserMigration_ForgotPassword
	カスタムメッセージ	CustomMessage_ForgotPassword
	カスタム E メール送信者	CustomEmailSender_ForgotPassword
	カスタム SMS 送信者	CustomSMSSender_ForgotPassword
/confirmforgotpassword	確認後	PostConfirmation_ConfirmForgotPassword

フェデレーティッドユーザーの Lambda トリガー

次の Lambda トリガーを使用して、フェデレーションプロバイダーでサインインするユーザーのユーザープールのワークフローをカスタマイズできます。

> **Note**
>
> フェデレーションユーザーは Amazon Cognito でホストされている UI を使用してサインインでき、また、ID プロバイダーのサインインページにサイレントリダイレクトするリクエストを 認可エンドポイント に 生成することもできます。Amazon Cognito ユーザープール API を使用してフェデレーションユーザーをサインインすることはできません。

フェデレーティッドユーザートリガーのソース

サインインイベント	Lambda トリガー	トリガーソース
初回サインイン	サインアップ前	`PreSignUp_ExternalProvider`
	確認後	`PostConfirmation_ConfirmSignUp`
	トークン生成前	`TokenGeneration_HostedAuth`
その後のサインイン	認証前	`PreAuthentication_Authentication`
	認証後	`PostAuthentication_Authentication`
	トークン生成前	`TokenGeneration_HostedAuth`

フェデレーティッドサインインは、ユーザープールの カスタム認証チャレンジの Lambda トリガー、ユーザー移行の Lambda トリガー、カスタムメッセージの Lambda トリガー、または カスタム送信者の Lambda トリガー を呼び出しません。

Lambda トリガーのユーザープールの機能オペレーションへの接続

各 Lambda トリガーは、ユーザープールで機能的な役割を果たします。例えば、トリガーを使用してサインアップフローを変更したり、カスタム認証チャレンジを追加したりできます。Amazon Cognito が Lambda 関数に送信するイベントには、その機能ロールを構成する複数のアクションのう

ちの 1 つが反映されます。例えば、Amazon Cognito は、ユーザーがサインアップしたとき、およびユーザーを作成したときに、事前サインアップトリガーを呼び出します。同じ機能的役割を持つ、これらのさまざまなケースには、それぞれ独自の `triggerSource` 値があります。Lambda 関数は、呼び出したオペレーションに基づいて受信イベントを異なる方法で処理できます。

また Amazon Cognito は、イベントがトリガーソースに対応するときに、割り当てられたすべての関数を呼び出します。例えば、ユーザー移行トリガーと事前認証トリガーを割り当てたユーザープールにユーザーがサインインすると、両方がアクティブになります。

サインアップ、確認、およびサインイン (認証) トリガー

トリガー	triggerSource 値	イベント
サインアップ前	`PreSignUp_SignUp`	サインアップ前。
サインアップ前	`PreSignUp_AdminCreateUser`	管理者が新しいユーザーを作成するときのサインアップ前
サインアップ前	`PreSignUp_ExternalProvider`	外部 ID プロバイダーのサインアップ前。
確認後	`PostConfirmation_ConfirmSignUp`	サインアップの確認後。
確認後	`PostConfirmation_ConfirmForgotPassword`	パスワードを忘れた場合の確認後。
認証前	`PreAuthentication_Authentication`	認証前。
認証後	`PostAuthentication_Authentication`	認証後。

カスタム認証チャレンジトリガー

Trigger	triggerSource 値	イベント
認証チャレンジの定義	`DefineAuthChallenge_Authentication`	認証チャレンジの定義。
認証チャレンジの作成	`CreateAuthChallenge_Authentication`	認証チャレンジの作成。
認証チャレンジの検証。	`VerifyAuthChallengeResponse_Authentication`	認証チャレンジレスポンスの確認。

トークン生成前トリガー

Trigger	triggerSource 値	イベント
トークン生成前	`TokenGeneration_HostedAuth`	Amazon Cognito は、のホストされた UI のサインインページからユーザーを認証します。
トークン生成前	`TokenGeneration_Authentication`	ユーザー認証フローが完了しました。
トークン生成前	`TokenGeneration_NewPasswordChallenge`	管理者がユーザーを作成します。ユーザーが一時パスワードを変更する必要があるときに、Amazon Cognito は、これを呼び出します。
トークン生成前	`TokenGeneration_AuthenticateDevice`	ユーザーデバイスの認証の終了。
トークン生成前	`TokenGeneration_RefreshTokens`	ユーザーは、ID およびアクセストークンを更新しようとします。

移行ユーザートリガー

Trigger	triggerSource 値	イベント
ユーザー移行	`UserMigration_Authentication`	サインイン時のユーザー移行。
ユーザー移行	`UserMigration_ForgotPassword`	パスワードを忘れた場合のフロー実行時のユーザー移行

カスタムメッセージトリガー

Trigger	triggerSource 値	イベント
カスタムメッセージ	`CustomMessage_SignUp`	ユーザーがユーザープールにサインアップしたときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_AdminCreateUser`	管理者としてユーザーを作成し、Amazon Cognito から一時パスワードが送信されるときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_ResendCode`	既存のユーザーが新しい確認コードをリクエストしたときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_ForgotPassword`	ユーザーがパスワードのリセットを要求したときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_UpdateUserAttribute`	ユーザーが E メールアドレスまたは電話番号を変更し、Amazon Cognito が検証コードを送信したときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_VerifyUserAttribute`	ユーザーが E メールアドレスまたは電話番号を追加し

Trigger	triggerSource 値	イベント
		、Amazon Cognito が検証コードを送信したときのカスタムメッセージ。
カスタムメッセージ	`CustomMessage_Authentication`	SMS MFA を設定したユーザーがサインインしたときのカスタムメッセージ。

サインアップ前の Lambda トリガー

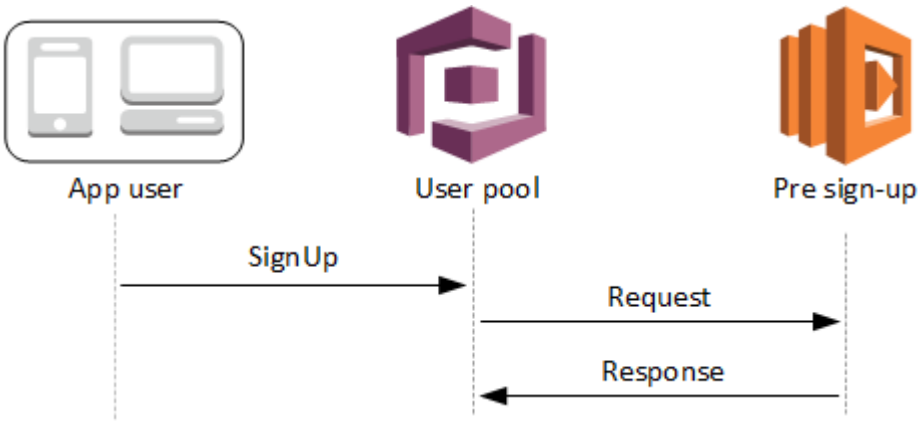
Amazon Cognito は、新しいユーザーをサインアップする直前に、サインアップ前の AWS Lambda 関数をアクティブにします。サインアッププロセスの一環として、この関数を使用してカスタム検証を実行し、検証の結果に基づいて、登録リクエストを受け入れるか拒否することができます。

トピック

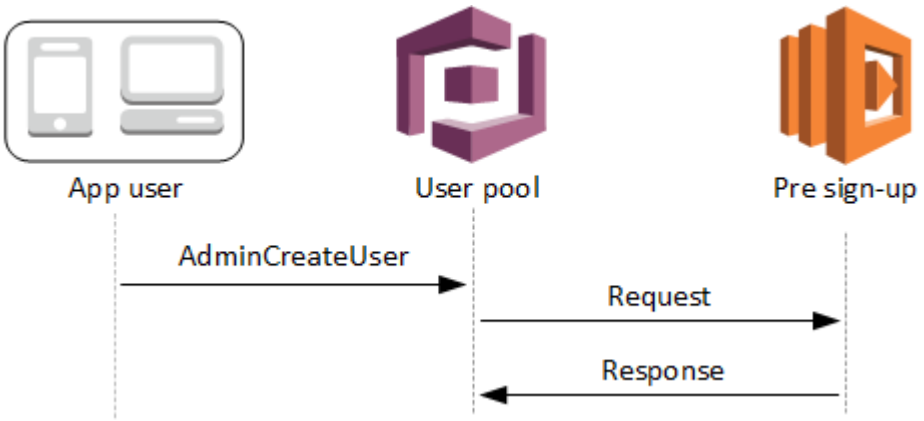
- サインアップ前の Lambda フロー
- サインアップ前の Lambda トリガーのパラメータ
- サインアップのチュートリアル
- サインアップ前の例: 登録済みドメインのユーザーを自動確認する
- サインアップ前の例: すべてのユーザーを自動確認して自動検証する
- サインアップ前の例:ユーザー名が 5 文字未満の場合にサインアップを拒否する

サインアップ前の Lambda フロー

クライアントのサインアップフロー



サーバーのサインアップフロー



リクエストには、クライアントからの検証データが含まれます。このデータは、ユーザープール SignUp と AdminCreateUser API メソッドに渡された`ValidationData`値から取得されます。

サインアップ前の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
```

```
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },

    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

サインアップ前のリクエストパラメータ

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。属性名はキーです。

validationData

新しいユーザーの作成リクエストでアプリケーションから Amazon Cognito に渡したユーザー属性データを示す 1 つ以上のキーと値のペアです。この情報を AdminCreateUserまたは SignUp API リクエストの ValidationData パラメータで Lambda 関数に送信します。

Amazon Cognito は、作成したユーザーの属性として ValidationData データを設定しません。ValidationData は、サインアップ前の Lambda トリガーのために指定する一時的なユーザー情報です。

clientMetadata

サインアップ前のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。このデータは、、AdminCreateUser、、ForgotPasswordおよび の ClientMetadata API アクションで パラメータを使用して Lambda AdminRespondToAuthChallenge関数に渡すことができますSignUp。

サインアップ前のレスポンスパラメータ

ユーザーを自動的に確認する場合は、レスポンスで autoConfirmUser を true に設定できます。autoVerifyEmail を true に設定してユーザーの E メールを自動検証できます。autoVerifyPhone を true に設定してユーザーの電話番号を自動検証できます。

> **Note**
>
> サインアップ前 Lambda 関数が AdminCreateUser API によってトリガーされる場合、レスポンスパラメータの autoVerifyPhone、autoVerifyEmail、および autoConfirmUser は Amazon Cognito によって無視されます。

autoConfirmUser

ユーザーを自動的に確認する場合は true に、それ以外の場合は false に設定します。

autoVerifyEmail

true に設定すると、サインアップしたユーザーの E メールアドレスを検証済みとして設定します。それ以外の場合は false です。autoVerifyEmail が true に設定されている場合、email 属性は有効な null 以外の値である必要があります。それ以外の場合はエラーが発生し、ユーザーがサインアップを完了できません。

email 属性がエイリアスとして選択されている場合、autoVerifyEmail が設定されていると、ユーザーの E メールアドレスのエイリアスが自動的に作成されます。その E メールアドレスのエイリアスが既に存在している場合は、エイリアスは新規ユーザーに移動され、以前のユーザーの E メールアドレスは未検証としてマークされます。詳細については、「ログイン属性のカスタマイズ」を参照してください。

autoVerifyPhone

true に設定すると、サインアップしたユーザーの電話番号を検証済みとして設定します。それ以外の場合は false です。autoVerifyPhone が true に設定されている場合、phone_number 属性は有効な null 以外の値である必要があります。それ以外の場合はエラーが発生し、ユーザーがサインアップを完了できません。

phone_number 属性がエイリアスとして選択されている場合、autoVerifyPhone が設定されていると、ユーザーの電話番号のエイリアスが自動的に作成されます。その電話番号のエイリアスが既に存在している場合、エイリアスは新規ユーザーに移動され、以前のユーザーの電話番号

は未検証としてマークされます。詳細については、「ログイン属性のカスタマイズ」を参照してください。

サインアップのチュートリアル

サインアップ前の Lambda 関数は、ユーザーのサインアップ前にトリガーされます。JavaScript、Android、iOS に関する以下の Amazon Cognito サインアップチュートリアルを参照してください。

プラットフォーム	チュートリアル
JavaScript ID SDK	でユーザーをサインアップする JavaScript
Android ID SDK	Android でのユーザーのサインアップ
iOS ID SDK	iOS でのユーザーのサインアップ

サインアップ前の例: 登録済みドメインのユーザーを自動確認する

サインアップ前の Lambda トリガーを使用して、ユーザープールにサインアップする新しいユーザーを検証するためのカスタムロジックを追加できます。これは、新しいユーザーをサインアップする方法を示すサンプル JavaScript プログラムです。これは認証の一環としてサインアップ前の Lambda トリガーを呼び出します。

JavaScript

```
var attributeList = [];
var dataEmail = {
  Name: "email",
  Value: "...", // your email here
};
var dataPhoneNumber = {
  Name: "phone_number",
  Value: "...", // your phone number here with +country code and no delimiters in
 front
};

var dataEmailDomain = {
  Name: "custom:domain",
```

```
  Value: "example.com",
};
var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataPhoneNumber
);
var attributeEmailDomain = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataEmailDomain
);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);
attributeList.push(attributeEmailDomain);

var cognitoUser;
userPool.signUp(
  "username",
  "password",
  attributeList,
  null,
  function (err, result) {
    if (err) {
      alert(err);
      return;
    }
    cognitoUser = result.user;
    console.log("user name is " + cognitoUser.getUsername());
  }
);
```

以下は、ユーザープールサインアップ前の Lambda トリガーを使用してサインアップ直前に呼び出されるサンプル Lambda トリガーです。これは、カスタム属性の [custom:domain] を使用して特定の E メールドメインからの新しいユーザーを自動的に確認します。このカスタムドメインに属していない新しいユーザーは、ユーザープールには追加されますが、自動的には確認されません。

Node.js

```
exports.handler = (event, context, callback) => {
  // Set the user pool autoConfirmUser flag after validating the email domain
  event.response.autoConfirmUser = false;

  // Split the email address so we can compare domains
```

```
  var address = event.request.userAttributes.email.split("@");

  // This example uses a custom attribute "custom:domain"
  if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
    if (event.request.userAttributes["custom:domain"] === address[1]) {
      event.response.autoConfirmUser = true;
    }
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
```

```
            "custom:domain": "example.com"
        }
    },
    "response": {}
}
```

サインアップ前の例: すべてのユーザーを自動確認して自動検証する

次の例では、すべてのユーザーを確認し、ユーザーの email 属性と phone_number 属性を検証済みとして設定します (属性が存在する場合)。また、エイリアシングが有効になっている場合は、自動検証が設定されていると、phone_number および email にエイリアスが作成されます。

> **Note**
>
> 同じ電話番号のエイリアスが既に存在している場合は、エイリアスは新規ユーザーに移動され、以前のユーザーの phone_number は未検証としてマークされます。E メールアドレスの場合も同様です。これを防ぐには、ユーザープール ListUsers API を使用して、新しいユーザーの電話番号または E メールアドレスをエイリアスとして既に使用している既存のユーザーがいるかどうかを確認できます。

Node.js

```
const handler = async (event) => {
  // Confirm the user
  event.response.autoConfirmUser = true;
  // Set the email as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }

  // Set the phone number as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("phone_number")) {
    event.response.autoVerifyPhone = true;
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    # Confirm the user
    event['response']['autoConfirmUser'] = True

    # Set the email as verified if it is in the request
    if 'email' in event['request']['userAttributes']:
        event['response']['autoVerifyEmail'] = True

    # Set the phone number as verified if it is in the request
    if 'phone_number' in event['request']['userAttributes']:
        event['response']['autoVerifyPhone'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

サインアップ前の例:ユーザー名が 5 文字未満の場合にサインアップを拒否する

次の例は、サインアップリクエストのユーザー名の長さを調べます。次の例は、ユーザーが 5 文字未満の名前を入力した場合、エラーを返します。

Node.js

```
exports.handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

確認後の Lambda トリガー

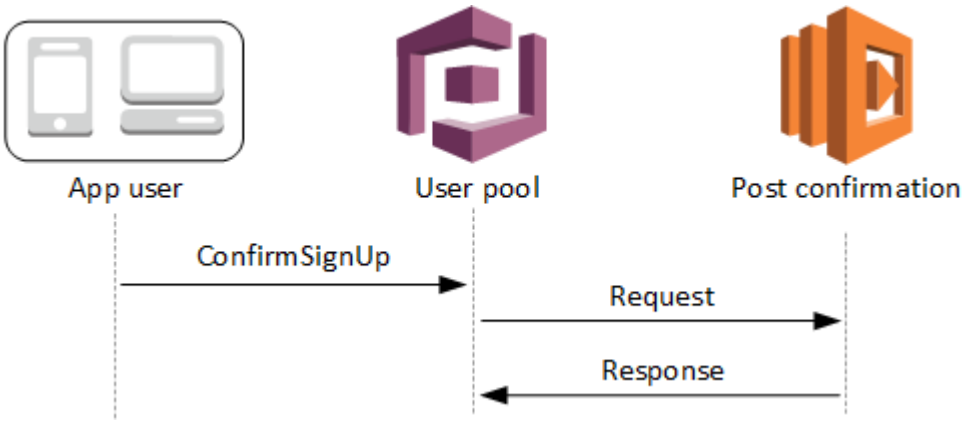
Amazon Cognito は、サインアップしたユーザーがユーザーアカウントを確認した後にこのトリガーを呼び出します。投稿確認 Lambda 関数では、カスタムメッセージを送信したり、カスタム API リクエストを追加したりできます。たとえば、外部システムにクエリを実行して、ユーザーに追加の属性を入力できます。Amazon Cognito は、管理者認証情報を使用して作成したユーザーではなく、ユーザープールにサインアップしたユーザーに対してのみこのトリガーを呼び出します。

リクエストには、確認されたユーザーの現在の属性が含まれています。

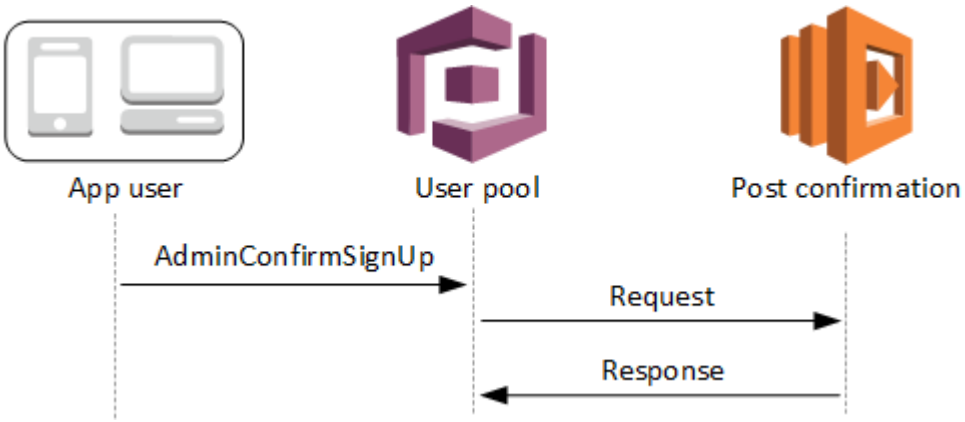
トピック

確認後の Lambda フロー

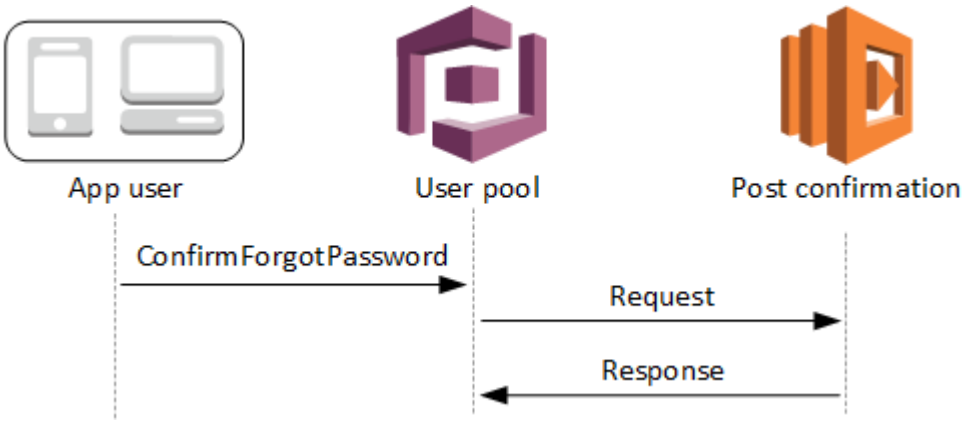
クライアントの確認サインアップフロー



サーバーの確認サインアップフロー



パスワードを忘れた場合の確認フロー



確認後の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
            "userAttributes": {
                "string": "string",
                . . .
            },
            "clientMetadata": {
             "string": "string",
              . . .
            }
        },
```

```
    "response": {}
}
```

確認後のリクエストパラメータ

userAttributes

ユーザー属性を表す 1 つ以上のキーバリューペア。

clientMetadata

確認後のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。このデータは、AdminConfirmSignUp、ConfirmForgotPassword、ConfirmSignUp、および SignUp の API アクションで ClientMetadata パラメータを使用することによって Lambda 関数に渡すことができます。

確認後のレスポンスパラメータ

レスポンスで返される追加の情報は想定されていません。

ユーザー確認チュートリアル

確認後の Lambda 関数は、Amazon Cognito が新しいユーザーを確認した直後にトリガーされます。JavaScript、Android、iOS に関する以下のユーザー確認チュートリアルを参照してください。

プラットフォーム	チュートリアル
JavaScript ID SDK	JavaScript でのユーザーの確認
Android ID SDK	Android でのユーザーの確認
iOS ID SDK	iOS でのユーザーの確認

確認後の例

この Lambda 関数の例では、Amazon SES を使用してユーザーに確認 E メールメッセージを送信します。詳細については、Amazon Simple Email Service デベロッパーガイドを参照してください。

Node.js

```
// Import required AWS SDK clients and commands for Node.js. Note that this requires
// the `@aws-sdk/client-ses` module to be either bundled with this code or included
// as a Lambda layer.
import { SES, SendEmailCommand } from "@aws-sdk/client-ses";
const ses = new SES();

const handler = async (event) => {
  if (event.request.userAttributes.email) {
    await sendTheEmail(
      event.request.userAttributes.email,
      `Congratulations ${event.userName}, you have been confirmed.`
    );
  }
  return event;
};

const sendTheEmail = async (to, body) => {
  const eParams = {
    Destination: {
      ToAddresses: [to],
    },
    Message: {
      Body: {
        Text: {
          Data: body,
        },
      },
      Subject: {
        Data: "Cognito Identity Provider registration completed",
      },
    },
    // Replace source_email with your SES validated email address
    Source: "<source_email>",
  };
  try {
    await ses.send(new SendEmailCommand(eParams));
  } catch (err) {
    console.log(err);
  }
};
```

```
export { handler };
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

認証前の Lambda トリガー

Amazon Cognito は、ユーザーがサインインしようとするときにこのトリガーを呼び出し、準備アクションを実行するカスタム認証を作成できるようにします。たとえば、認証リクエストを拒否したり、外部システムへのセッションデータを記録したりできます。

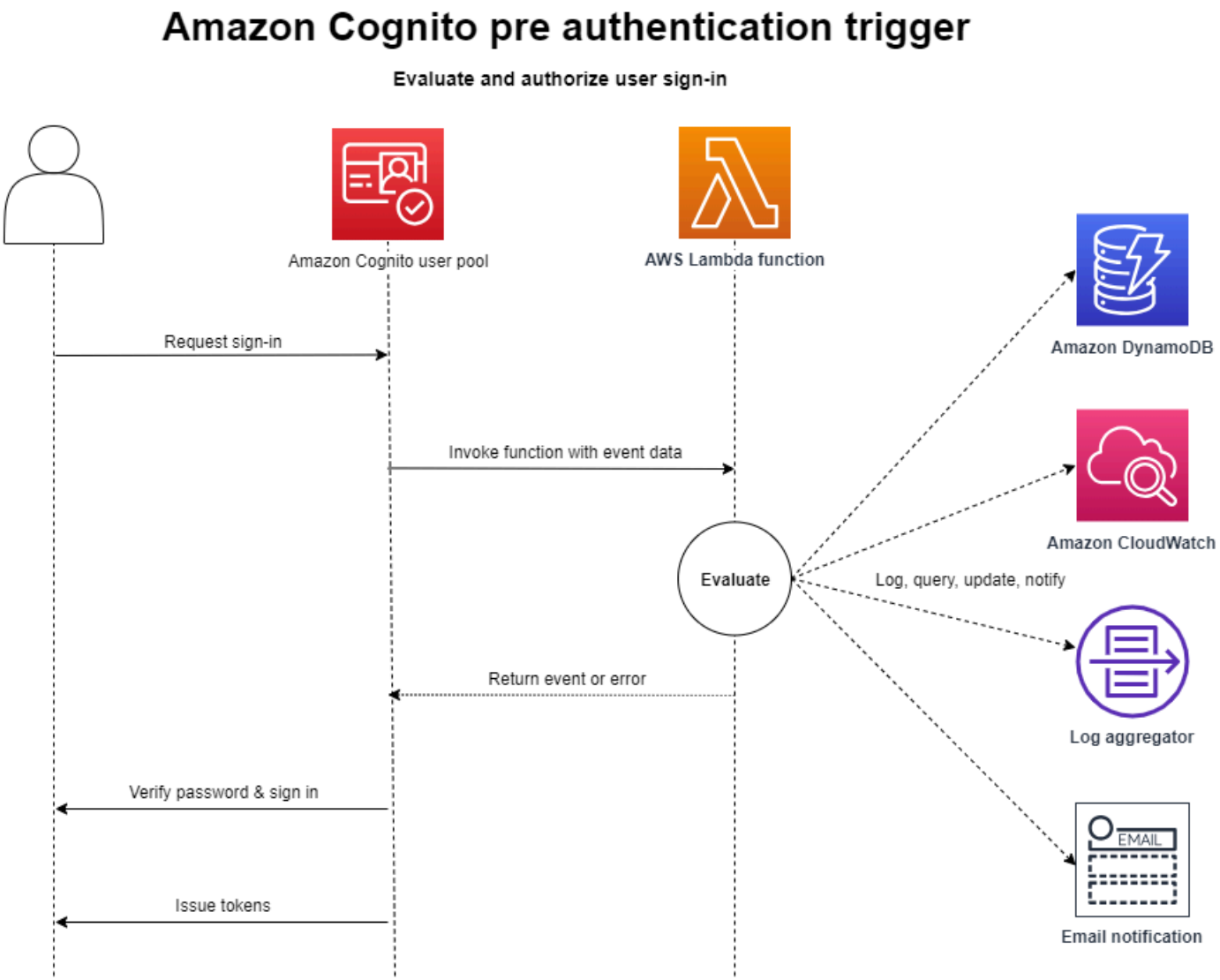
> **Note**
>
> ユーザーが存在しない場合や、ユーザープールに既にセッションがある場合、この Lambda トリガーはアクティブになりません。ユーザープールアプリクライアントの `PreventUserExistenceErrors` 設定が `ENABLED` になっている場合、Lambda トリガーはアクティブになります。

トピック

- 認証フローの概要
- 認証前の Lambda トリガーのパラメータ

- 事前認証の例

認証フローの概要



リクエストには、アプリケーションからユーザープールの `InitiateAuth` および `AdminInitiateAuth` API オペレーションに渡す `ClientMetadata` 値のクライアント検証データが含まれます。

詳細については、「ユーザープール認証フロー」を参照してください。

認証前の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {}
}
```

認証前のリクエストパラメータ

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。

userNotFound

ユーザープールクライアントの `PreventUserExistenceErrors` を `ENABLED` に設定すると、Amazon Cognito はこのブール値を入力します。

validationData

ユーザーのサインインリクエストに検証データを含む 1 つ以上のキーバリューペア。このデータを Lambda 関数に渡すには、InitiateAuth および AdminInitiateAuth API アクションで ClientMetadata パラメータを使用します。

認証前のレスポンスパラメータ

Amazon Cognito は、レスポンスに追加の戻り情報を期待していません。関数がエラーを返してサインイン試行を拒否したり、API オペレーションを使用してリソースをクエリしたり変更できます。

事前認証の例

このサンプル関数は、特定のアプリクライアントからユーザープールにサインインされるのを防ぎます。事前認証の Lambda 関数は、ユーザーが既存のセッションを持っている場合は呼び出されないため、この関数はブロックするアプリクライアント ID の新規セッションのみを防止します。

Node.js

```
const handler = async (event) => {
  if (
    event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked"
  ) {
    throw new Error("Cannot authenticate users from this user pool app client");
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be
 blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
}
```

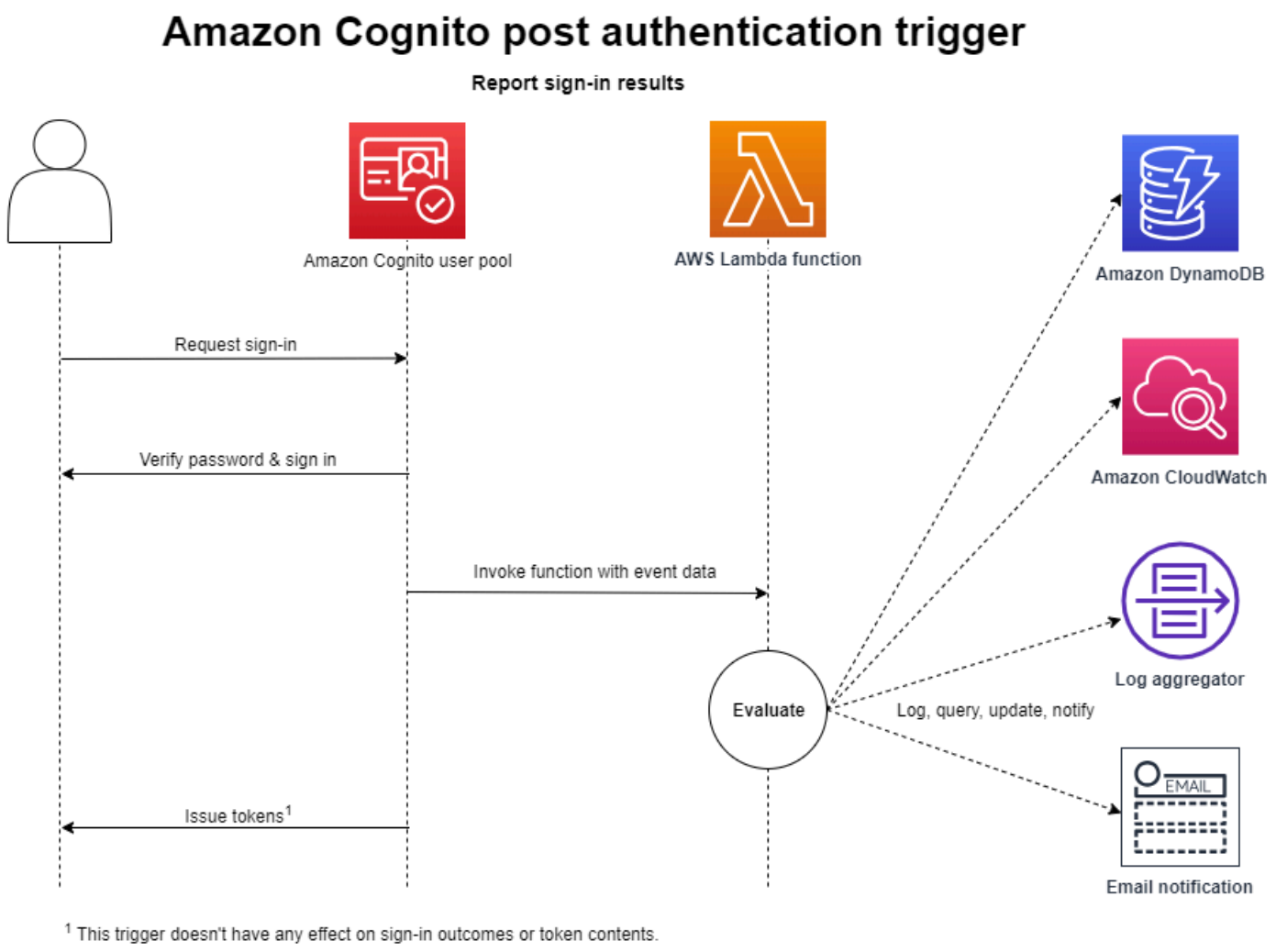
認証後の Lambda トリガー

Amazon Cognito はユーザーのサインイン後にこのトリガーを呼び出すため、Amazon Cognito がユーザーを認証した後にカスタムロジックを追加できます。

トピック

- 認証フローの概要
- 認証後の Lambda トリガーのパラメータ
- 認証チュートリアル
- 認証後の例

認証フローの概要



詳細については、「ユーザープール認証フロー」を参照してください。

認証後の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
```

```
            "string": "string",
            . . .
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            . . .
           }
       },
    "response": {}
}
```

認証後のリクエストパラメータ

newDeviceUsed

このフラグは、ユーザーが新しいデバイスにサインインしているかどうかを示します。Amazon Cognito は、ユーザープールの記憶済みデバイス値が `Always` または `User Opt-In` である場合にのみ、このフラグを設定します。

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。

clientMetadata

認証後のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。このデータを Lambda 関数に渡すには、AdminRespondToAuthChallenge および RespondToAuthChallenge API アクションで ClientMetadata パラメータを使用します。Amazon Cognito は、認証後関数に渡すリクエストの AdminInitiateAuth および InitiateAuth API オペレーションの ClientMetadata パラメータからのデータを含めません。

認証後のレスポンスパラメータ

Amazon Cognito は、レスポンスに追加の返品情報を期待しません。関数では、API オペレーションを使用してリソースをクエリして変更したり、イベントメタデータを外部システムに記録することができます。

認証チュートリアル

Amazon Cognito がユーザーにサインインした直後、認証後の Lambda 関数をアクティブ化します。JavaScript、Android、iOS に関する以下のサインインチュートリアルを参照してください。

プラットフォーム	チュートリアル
JavaScript ID SDK	JavaScript でのユーザーのサインイン
Android ID SDK	Android でのユーザーのサインイン
iOS ID SDK	iOS でのユーザーのサインイン

認証後の例

この認証後の Lambda 関数のサンプルは、正常に行われたサインインからのデータをCloudWatch Logs に送信します。

Node.js

```
const handler = async (event) => {
  // Send post authentication data to Amazon CloudWatch logs
  console.log("Authentication successful");
  console.log("Trigger function =", event.triggerSource);
  console.log("User pool = ", event.userPoolId);
  console.log("App client ID = ", event.callerContext.clientId);
  console.log("User ID = ", event.userName);

  return event;
};

export { handler }
```

Python

```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])
```

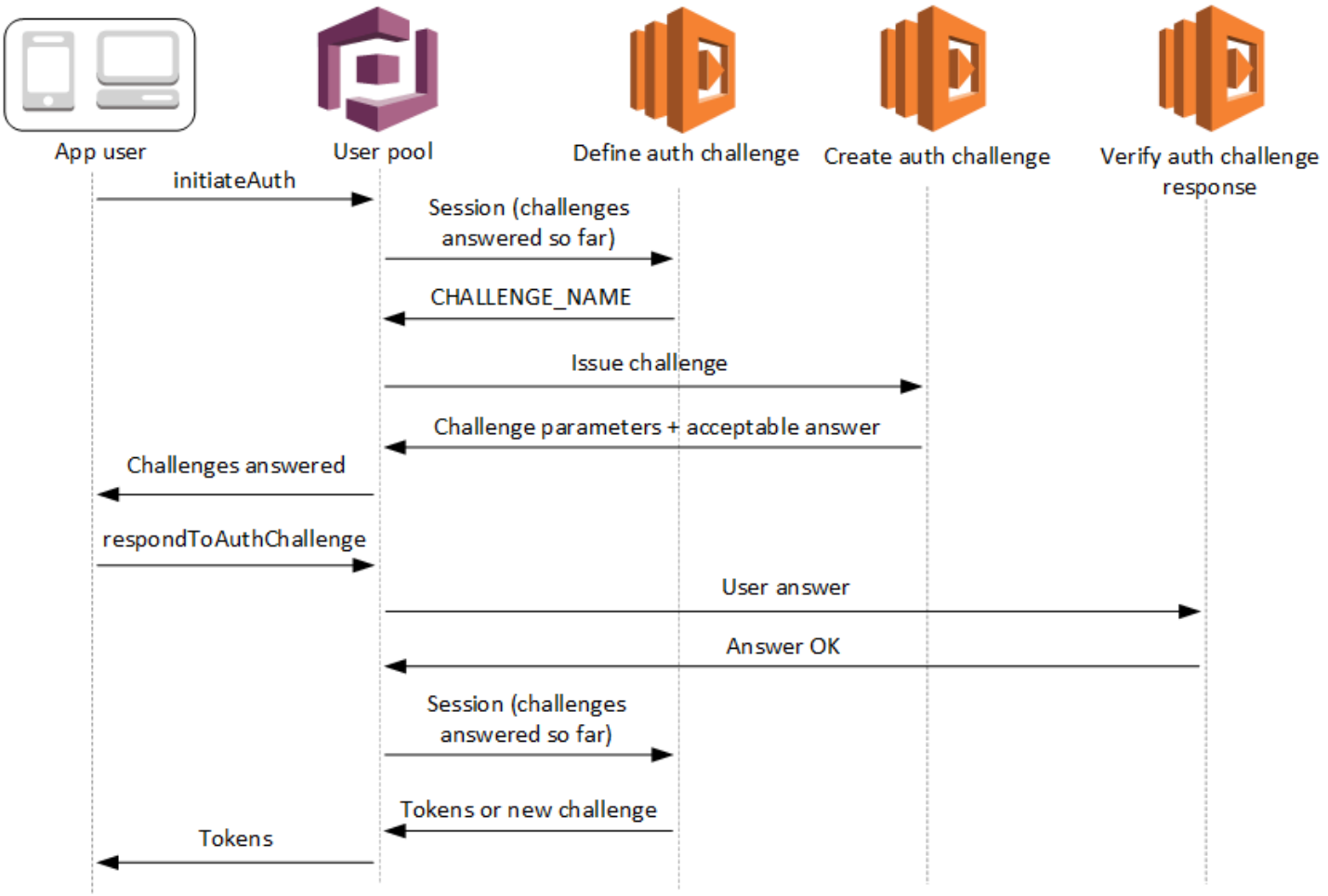
```
    # Return to Amazon Cognito
    return event
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
      "clientId": "12345"
  },
  "response": {}
}
```

カスタム認証チャレンジの Lambda トリガー



これらの Lambda トリガーは、ユーザープールのカスタム認証フローの一環として独自のチャレンジを発行し、検証します。

認証チャレンジの定義

Amazon Cognito は、このトリガーを呼び出してカスタム認証フローを開始します。

認証チャレンジの作成

Amazon Cognito は、このトリガーを [Define Auth Challenge] (認証チャレンジの定義後に呼び出して、カスタムチャレンジを作成します。

認証チャレンジレスポンスの検証

Amazon Cognito は、このトリガーが呼び出して、カスタムチャレンジに対するエンドユーザーからのレスポンスが有効であるかどうかを検証します。

これらのチャレンジ Lambda トリガーに新しいチャレンジタイプに組み込むことができます。例えば、CAPTCHA や動的なチャレンジ質問をチャレンジタイプに含めることができます。

ユーザープールの InitiateAuth と RespondToAuthChallenge の API メソッドを使用して、2 つの一般的なステップで認証を実行できます。

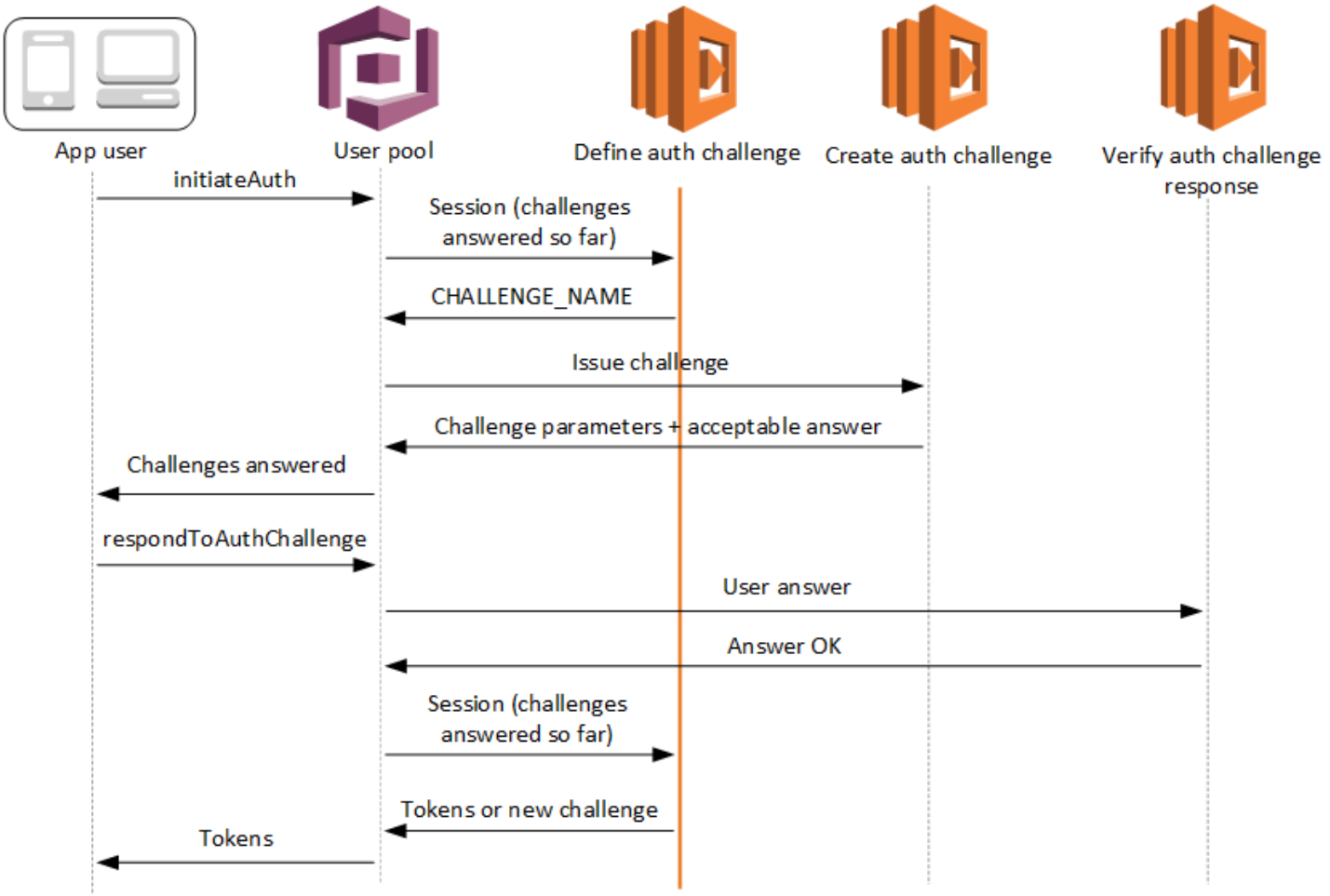
このフローでは、認証が失敗するか、トークンが発行されるまで、ユーザーは引き続きチャレンジに回答して認証を行います。これらの 2 つの API コールを繰り返して複数の異なるチャレンジを含めることができます。

> **Note**
>
> Amazon Cognito でホストされた UI は、カスタム認証チャレンジの Lambda トリガーを使用したカスタム認証フローをサポートしていません。

トピック

- 認証チャレンジの定義の Lambda トリガー
- 認証チャレンジの作成の Lambda トリガー
- 認証チャレンジレスポンスの検証の Lambda トリガー

認証チャレンジの定義の Lambda トリガー



認証チャレンジの定義

Amazon Cognito は、このトリガーを呼び出してカスタム認証フローを開始します。

この Lambda トリガーのリクエストには、`session` が含まれます。`session` パラメータは、現在の認証プロセスでユーザーに提示されたすべてのチャレンジが含まれる配列です。リクエストには、対応する結果も含まれます。`session` 配列は、チャレンジの詳細 (`ChallengeResult`) を時系列に保存します。チャレンジ `session[0]` は、ユーザーが最初に受け取るチャレンジを表します。

Amazon Cognito では、カスタムチャレンジを発行する前にユーザーパスワードを検証させることができます。リクエストレートクォータの認証カテゴリに関連付けられているすべての Lambda トリガーは、カスタムチャレンジフローで SRP 認証を行うと、実行されます。以下は、そのプロセスの概要です。

1. アプリは、AuthParameters マップを使用して InitiateAuth または AdminInitiateAuth を呼び出してサインインを開始します。パラメータには CHALLENGE_NAME: SRP_A,、SRP_A および USERNAME の値を含める必要があります。
2. Amazon Cognito は、challengeName: SRP_A と challengeResult: true を含む初期セッションで、認証チャレンジの定義 Lambda トリガーを呼び出します。
3. Lambda 関数は、これらの入力を受け取った後、challengeName: PASSWORD_VERIFIER、issueTokens: false、failAuthentication: false で応答します。
4. パスワードの検証が成功すると、Amazon Cognito challengeName: PASSWORD_VERIFIER と challengeResult: true が含まれる新しいセッションで Lambda 関数を再度呼び出します。
5. カスタムチャレンジを開始するために、Lambda 関数は challengeName: CUSTOM_CHALLENGE、issueTokens: false、および failAuthentication: false で応答します。パスワード検証でカスタム認証フローを開始したくない場合は、CHALLENGE_NAME: CUSTOM_CHALLENGE を含む AuthParameters マップでサインインを開始できます。
6. チャレンジループは、すべてのチャレンジが回答されるまで繰り返します。

トピック

- 認証チャレンジの定義の Lambda トリガーのパラメータ
- 認証チャレンジの定義の例

認証チャレンジの定義の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
```

```
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

認証チャレンジの定義のリクエストパラメータ

Amazon Cognito が Lambda 関数を呼び出すときに、次のパラメータを提供します。

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。

userNotFound

ユーザープールクライアントの `PreventUserExistenceErrors` が `ENABLED` に設定されている場合に、Amazon Cognito が入力するブール値です。`true` の値は、ユーザー ID (ユーザー名、メールアドレス、その他の詳細など) が既存のいずれのユーザーとも一致しなかったことを意味します。`PreventUserExistenceErrors` が `ENABLED` に設定されている場合、サービスは存在しないユーザーをアプリに通知しません。Lambda 関数が同じユーザーエクスペリエンスを維持し、レイテンシーを考慮することをお勧めします。この方法では、発信者はユーザーが存在する場合と存在しない場合で異なる動作を検出することができません。

セッション

`ChallengeResult` 要素の配列。それぞれに以下の要素が含まれます。

challengeName

次のチャレンジタイプのいずれかです:
`CUSTOM_CHALLENGE`、`SRP_A`、`PASSWORD_VERIFIER`、`SMS_MFA`,
`DEVICE_SRP_AUTH`、`DEVICE_PASSWORD_VERIFIER`、または `ADMIN_NO_SRP_AUTH`。

認証チャレンジの定義機能が、多要素認証を設定したユーザーに `PASSWORD_VERIFIER` チャレンジを発行すると、Amazon Cognito はそれに続いて `SMS_MFA` チャレンジを行います。関

数には、SMS_MFA チャレンジからの入力イベントの処理を含めます。定義した認証チャレンジ関数から SMS_MFA チャレンジを呼び出す必要はありません。

> **Important**
>
> ユーザーが正常に認証され、トークンが発行されるべきかを判断するときは、常に define auth challenge 関数で `challengeName` をチェックし、予想された値と一致することを確認します。

challengeResult

ユーザーが正常にチャレンジを完了した場合は `true` に、それ以外の場合は `false` に設定します。

challengeMetadata

カスタムチャレンジの名前を入力します。`challengeName` が `CUSTOM_CHALLENGE` である場合にのみ使用されます。

clientMetadata

認証チャレンジの定義のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。このデータを Lambda 関数に渡すには、AdminRespondToAuthChallenge および RespondToAuthChallenge API オペレーションで `ClientMetadata` パラメータを使用できます。define auth challenge 関数を呼び出すリクエストに、AdminInitiateAuth および InitiateAuth API オペレーションの ClientMetadata パラメータに渡されたデータは含まれません。

認証チャレンジの定義のレスポンスパラメータ

レスポンスで、認証プロセスの次のステージを返すことができます。

challengeName

次のチャレンジの名前を含む文字列。ユーザーに新しいチャレンジを提示する場合は、ここでチャレンジ名を指定します。

issueTokens

ユーザーが認証チャレンジを十分に完了したと判断した場合、`true` に設定します。ユーザーがチャレンジを十分に満たしていない場合は、`false` に設定します。

failAuthentication

現在の認証プロセスを終了する場合は、`true` に設定します。現在の認証プロセスを続行するには、`false` に設定します。

認証チャレンジの定義の例

この例では、認証用に一連のチャレンジを定義し、それらのチャレンジがすべて正常に完了した場合にのみトークンを発行します。

Node.js

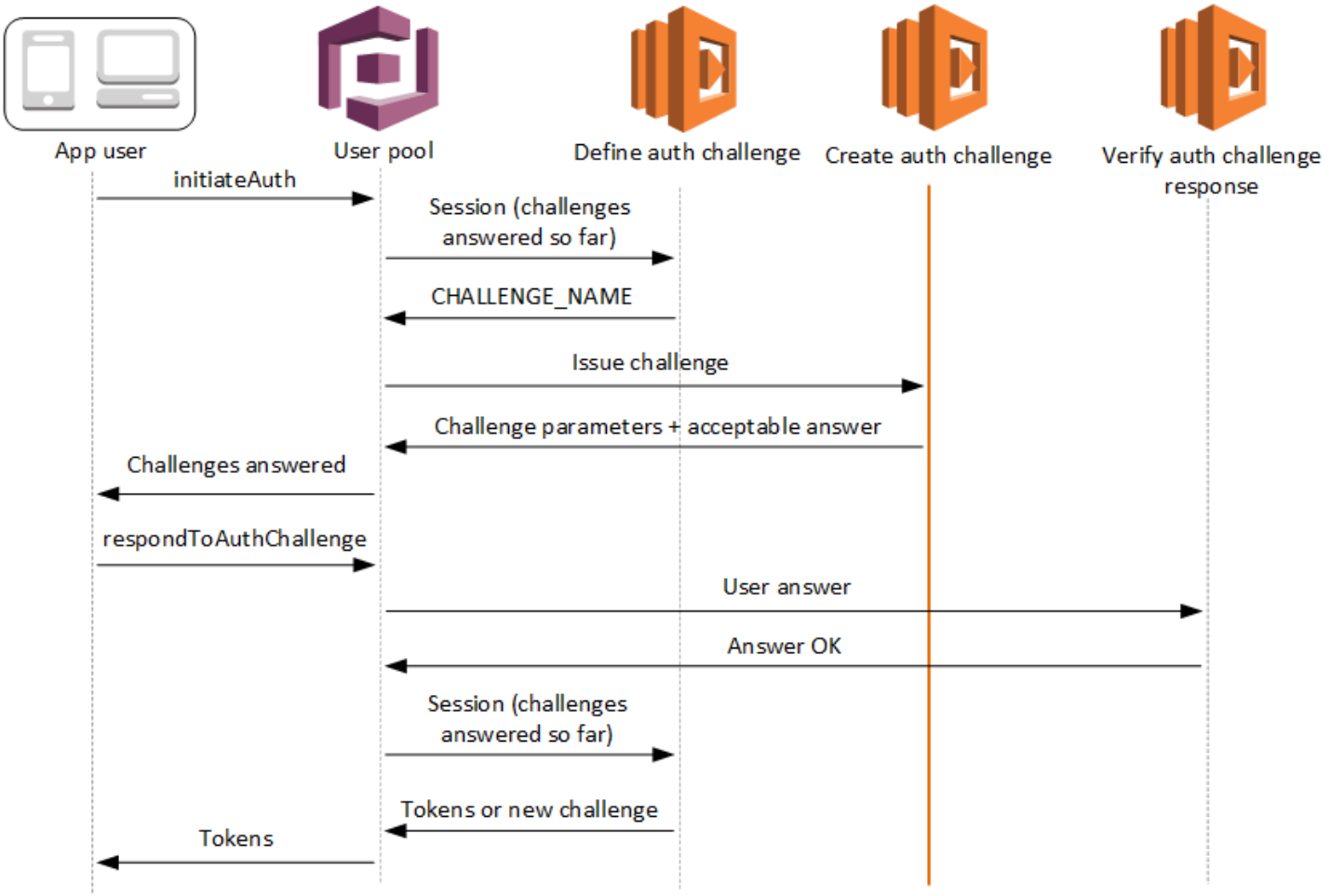
```
const handler = async (event) => {
  if (
    event.request.session.length == 1 &&
    event.request.session[0].challengeName == "SRP_A"
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
  } else if (
    event.request.session.length == 2 &&
    event.request.session[1].challengeName == "PASSWORD_VERIFIER" &&
    event.request.session[1].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 3 &&
    event.request.session[2].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[2].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 4 &&
    event.request.session[3].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[3].challengeResult == true
  ) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
```

```
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

  return event;
};

export { handler }
```

認証チャレンジの作成の Lambda トリガー



認証チャレンジの作成

認証チャレンジの定義トリガーの一部としてカスタムチャレンジが指定されている場合、Amazon Cognito は認証チャレンジの定義後にこのトリガーを呼び出します。これにより、カスタム認証フローを作成します。

この Lambda トリガーは、ユーザーに提示するチャレンジを作成するために呼び出されます。この Lambda トリガーのリクエストには challengeName と session が含まれます。challengeName は文字列であり、ユーザーに対する次のチャレンジの名前です。この属性の値は、認証チャレンジの定義の Lambda トリガーに設定されています。

チャレンジループは、すべてのチャレンジに応答するまで繰り返されます。

トピック

- 認証チャレンジの作成 Lambda トリガーパラメータ
- 認証チャレンジの作成の例

認証チャレンジの作成 Lambda トリガーパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
```

```
            "string": "string",
            . . .
        },
        "challengeMetadata": "string"
    }
}
```

認証チャレンジの作成のリクエストパラメータ

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。

userNotFound

このブール値は、ユーザープールクライアントで PreventUserExistenceErrors が ENABLED に設定されている場合に設定されます。

challengeName

新しいチャレンジの名前。

session

session 要素は ChallengeResult 要素の配列であり、それぞれに以下の要素が含まれます。

challengeName

チャレンジタイプ。"CUSTOM_CHALLENGE"、"PASSWORD_VERIFIER"、"SMS_MFA"、"DEVICE_SRP_AUTH"、"D のいずれかです。

challengeResult

ユーザーが正常にチャレンジを完了した場合は true に、それ以外の場合は false に設定します。

challengeMetadata

カスタムチャレンジの名前を入力します。challengeName が "CUSTOM_CHALLENGE" である場合にのみ使用されます。

clientMetadata

認証チャレンジの作成のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。このデータは、AdminRespondToAuthChallenge およ

び RespondToAuthChallenge API アクションで ClientMetadata パラメータを使用することによって Lambda 関数に渡すことができます。create auth challenge 関数を呼び出すリクエストに、AdminInitiateAuth および InitiateAuth API オペレーションで ClientMetadata パラメータに渡されたデータは含まれません。

認証チャレンジの作成のレスポンスパラメータ

publicChallengeParameters

クライアントアプリケーションでユーザーに提示されるチャレンジに使用する 1 つ以上のキー - 値ペア。このパラメータには、ユーザーにチャレンジを正確に提示するために必要なすべての情報を含める必要があります。

privateChallengeParameters

このパラメータは、認証チャレンジレスポンスの検証の Lambda トリガー以外では使用されません。このパラメータには、チャレンジに対するユーザーのレスポンスを検証するために必要な情報のすべてを含める必要があります。つまり、`publicChallengeParameters` には、ユーザーに示される質問が含まれ、`privateChallengeParameters` には、質問に対する有効な回答が含まれます。

challengeMetadata

カスタムチャレンジの名前 (カスタムチャレンジの場合)。

認証チャレンジの作成の例

CAPTCHA はユーザーに対するチャレンジとして作成されます。CAPTCHA イメージの URL が "`captchaUrl`" としてパブリックチャレンジパラメータに追加され、想定される回答がプライベートチャレンジパラメータに追加されます。

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
```

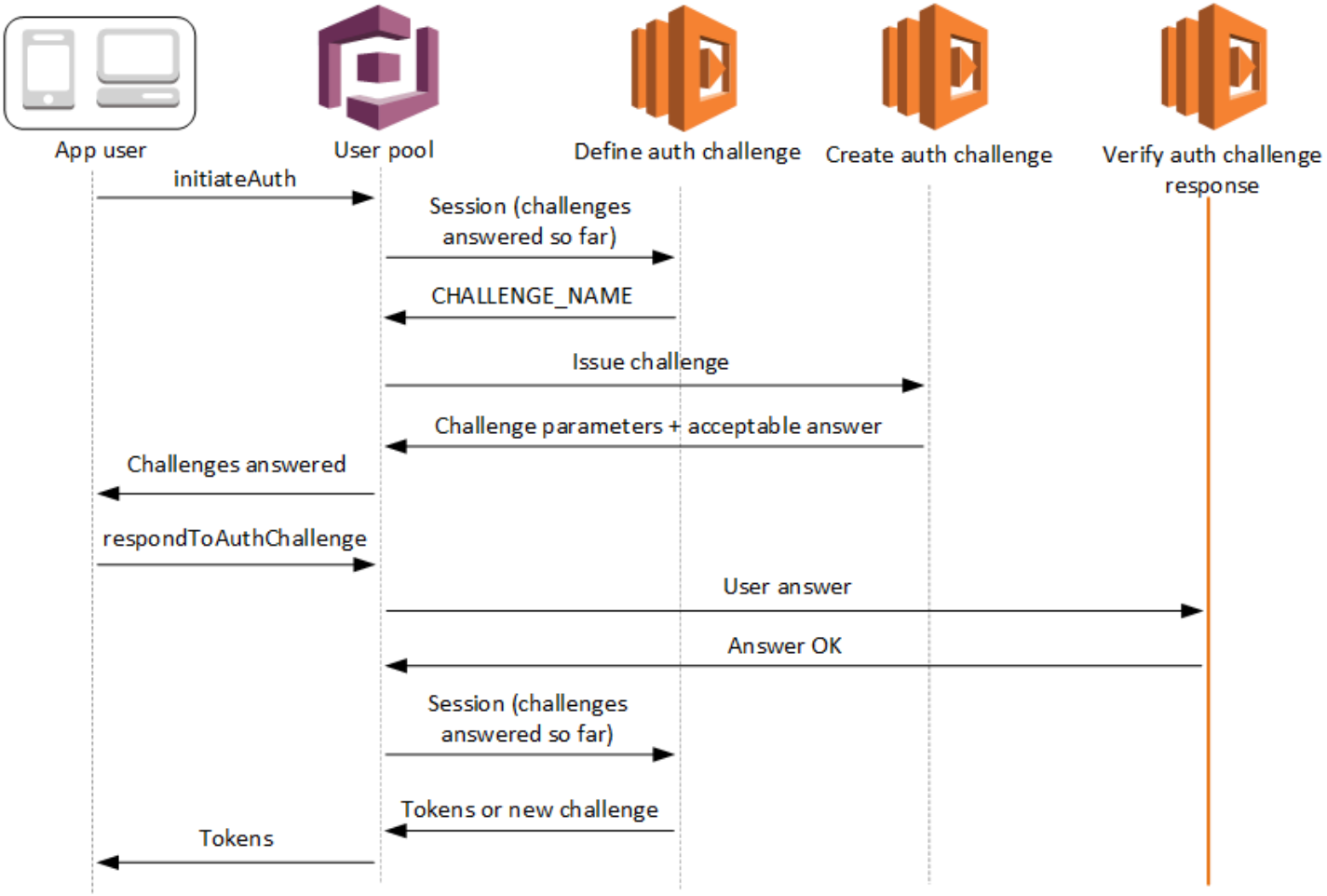
```
    event.response.privateChallengeParameters.answer = "5";
  }

  if (event.request.session.length === 3) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler }
```

認証チャレンジレスポンスの検証の Lambda トリガー



認証チャレンジレスポンスの検証

Amazon Cognito は、このトリガーが呼び出して、カスタム認証チャレンジに対するユーザーからのレスポンスが有効であるかどうかを検証します。これはユーザープールのカスタム認証フローの一環です。

このトリガーのリクエストには `privateChallengeParameters` および `challengeAnswer` パラメータが含まれます。`privateChallengeParameters` 値は、認証チャレンジの作成の Lambda トリガーによって返され、ユーザーからの期待されるレスポンスが含まれます。`challengeAnswer` パラメータには、チャレンジに対するユーザーのレスポンスが含まれます。

レスポンスには、`answerCorrect` 属性が含まれます。ユーザーがチャレンジを正常に完了すると、Amazon Cognito は属性値を `true` に設定します。ユーザーがチャレンジを正常に完了しなかった場合、Amazon Cognito は値を `false` に設定します。

チャレンジループは、すべてのチャレンジが回答されるまで繰り返します。

トピック

- 認証チャレンジの検証の Lambda トリガーのパラメータ
- 認証チャレンジレスポンスの確認の例

認証チャレンジの検証の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
```

```
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
    }
}
```

認証チャレンジの検証のリクエストパラメータ

userAttributes

このパラメータには、ユーザー属性を表す 1 つ以上の名前-値ペアが含まれます。

userNotFound

Amazon Cognito がユーザープールクライアントの `PreventUserExistenceErrors` を `ENABLED` に設定すると、Amazon Cognito はこのブール値を入力します。

privateChallengeParameters

このパラメータは、認証チャレンジの作成トリガーから取得されます。ユーザーがチャレンジに合格したかどうかを判断するために、Amazon Cognito はパラメータをユーザーの challengeAnswer パラメータと比較します。

このパラメータには、チャレンジに対するユーザーのレスポンスを検証するために必要な情報のすべてが含まれます。その情報には、Amazon Cognito がユーザーに提示する質問 (`publicChallengeParameters`) と、その質問に対する有効な回答 (`privateChallengeParameters`) が含まれます。認証チャレンジレスポンスの検証の Lambda トリガーのみがこのパラメータを使用します。

challengeAnswer

このパラメータ値は、チャレンジに対するユーザーの回答です。

clientMetadata

このパラメータには、認証チャレンジの検証のトリガーの Lambda 関数へのカスタム入力として提供できる 1 つまたは複数のキーバリューペアが含まれています。このデータを Lambda 関数に渡すには、AdminRespondToAuthChallenge および RespondToAuthChallenge API オペレーションで ClientMetadata パラメータを使用します。Amazon Cognito は、認証チャレンジの検証関数

に渡すリクエストの AdminInitiateAuth および InitiateAuth API オペレーションの ClientMetadata パラメータからのデータを含めません。

認証チャレンジの検証のレスポンスパラメータ

answerCorrect

ユーザーがチャレンジを正常に完了した場合、Amazon Cognito はこのパラメータを `true` に設定します。ユーザーがチャレンジを正常に完了しなかった場合、Amazon Cognito はパラメータを `false` に設定します。

認証チャレンジレスポンスの確認の例

この例では、チャレンジに対するユーザーのレスポンスが期待されるレスポンスに一致するかどうかを Lambda 関数がチェックします。ユーザーのレスポンスが期待されるレスポンスと一致した場合、Amazon Cognito は `answerCorrect` パラメータを `true` に設定します。

Node.js

```
const handler = async (event) => {
  if (
    event.request.privateChallengeParameters.answer ==
    event.request.challengeAnswer
  ) {
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  return event;
};

export { handler };
```

トークン生成前の Lambda トリガー

Amazon Cognito は、このトリガーをトークンの生成前に呼び出すため、ユーザープールトークンのクレームをカスタマイズできます。バージョン 1 または `V1_0` のトークン生成前トリガーイベントの基本機能を使用して、アイデンティティ (ID) トークンをカスタマイズできます。高度なセキュリ

ティ機能が有効なユーザープールでは、アクセストークンのカスタマイズにより、バージョン 2 または V2_0 のトリガーイベントを生成できます。

Amazon Cognito は、ID トークンに書き込むデータが含まれているリクエストとして V1_0 イベントを関数に送信します。V2_0 イベントは、Amazon Cognito が ID トークンとアクセストークンの両方に書き込むデータが含まれている 1 つのリクエストです。両方のトークンをカスタマイズするには、最新のトリガーバージョンを使用するように関数を更新し、同じレスポンスで両方のトークン用のデータを送信する必要があります。

この Lambda トリガーでは、Amazon Cognito がアプリケーションに発行する前の ID トークンとアクセストークンの一部のクレームを追加、削除、変更できます。この機能を使用するには、Amazon Cognito ユーザープールコンソールから Lambda 関数を関連付けるか、 AWS Command Line Interface (AWS CLI) でユーザープール LambdaConfig を更新します。

イベントバージョン

ユーザープールは、トークン生成前トリガーイベントのさまざまなバージョンを Lambda 関数に配信できます。V1_0 トリガーは、ID トークンを変更するためのパラメータを配信します。V2_0 トリガーは、以下のパラメータを配信します。

1. V1_0 トリガーの関数。
2. アクセストークンをカスタマイズする機能。
3. ID およびアクセストークンのクレーム値に複雑なデータ型を渡す機能 :
   - 文字列
   - 数
   - ブール値
   - 文字列、数値、ブール値、またはこれらの組み合わせの配列
   - JSON

> **Note**
>
> ID トークンでは、、、phone_number_verified、email_verifiedupdated_atおよびを除くクレームの値に複雑なオブジェクトを入力できますaddress。

ユーザープールはデフォルトでV1_0イベントを配信します。V2_0 イベントを送信するようにユーザープールを設定するには、Amazon Cognito コンソールでトリガーを設定するときに、基本機能

のトリガーイベントバージョン + アクセストークンのカスタマイズを選択します。 UpdateUserPool または CreateUserPool API リクエストLambdaVersionのLambdaConfigパラメータで の値を設定することもできます。V2_0 イベントによるアクセストークンのカスタマイズには、追加料金が適用されます。詳細については、「Amazon Cognito の料金」を参照してください。

除外されたクレームとスコープ

Amazon Cognito は、アクセストークンと ID トークンに対して追加、変更、または抑制できるクレームとスコープを制限します。Lambda 関数がこれらのクレームのいずれかに値を設定しようとすると、Amazon Cognito は元のクレーム値 (リクエストに存在する場合) を含むトークンを発行します。

共有クレーム

- acr
- amr
- at_hash
- auth_time
- azp
- exp
- iat
- iss
- jti
- nbf
- nonce
- origin_jti
- sub
- token_use

ID トークンのクレーム

- identities
- aud

- cognito:username

アクセストークンのクレーム

- username
- client_id
- scope

> **Note**
>
> アクセストークンのスコープは scopesToAdd および scopesToSuppress レスポンス値で変更できますが、scope クレームを直接変更することはできません。ユーザープールのリザーブドスコープ aws.cognito.signin.user.admin など、aws.cognito で始まるスコープを追加することはできません。

- device_key
- event_id
- version

以下のプレフィックスを含むクレームを追加またはオーバーライドすることはできませんが、これらをトークンで抑制または非表示にすることはできます。

- dev:
- cognito:

aud クレームをアクセストークンに追加することはできますが、その値は現在のセッションのアプリクライアント ID と一致する必要があります。リクエストイベントのクライアント ID は、event.callerContext.clientId から取得できます。

ID トークンのカスタマイズ

トークン生成前の Lambda トリガーを使用すると、ユーザープールのアイデンティティ (ID) トークンの内容をカスタマイズできます。ID トークンは、ウェブまたはモバイルアプリにサインインするためのユーザー属性を、信頼できる ID ソースから提供します。ID トークンの詳細については、「ID トークンの使用」を参照してください。

トークン生成前の Lambda トリガーでは、ID トークンを使用して以下のような操作を行います。

- ユーザーがアイデンティティプールにリクエストした IAM ロールをランタイムに変更します。
- 外部ソースからユーザー属性を追加します。
- 既存のユーザー属性値を追加または置換します。
- アプリに渡されるはずのユーザー属性が、ユーザーの許可されたスコープや、アプリクライアントに付与した属性の読み取りアクセス権により開示されることを抑制します。

アクセストークンのカスタマイズ

トークン生成前の Lambda トリガーを使用すると、ユーザープールのアクセストークンの内容をカスタマイズできます。アクセストークンは、Amazon Cognito トークン認可 API オペレーションやサードパーティ API などのアクセス保護されたリソースから情報を取得することをユーザーに許可します。Amazon Cognito では、クライアント認証情報付与を使用して (M2M) 認証用の machine-to-machineアクセストークンを生成できますが、M2M リクエストはトークン生成前のトリガー関数を呼び出しず、カスタマイズされたアクセストークンを発行できません。アクセストークンの詳細については、「アクセストークンの使用」を参照してください。

トークン生成前の Lambda トリガーでは、アクセストークンを使用して以下のような操作を行います。

- `scope` クレームでの OAuth 2.0 スコープの追加または抑制を行います 例えば、スコープ `aws.cognito.signin.user.admin` のみを割り当てる Amazon Cognito ユーザープール API 認証で生成されたアクセストークンにスコープを追加できます。
- ユーザープールグループのユーザーのメンバーシップを変更します。
- Amazon Cognito アクセストークンにまだ存在していないクレームを追加します。
- アプリに渡されるはずのクレームの開示を抑制します。

ユーザープールでのアクセスのカスタマイズをサポートするには、トリガーリクエストの更新バージョンを生成するようにユーザープールを設定する必要があります。ユーザープールを更新するには、次の手順に従います。

AWS Management Console

トークン生成前の Lambda トリガーでアクセストークンのカスタマイズをサポートするには

1. Amazon Cognito コンソールに移動し、[ユーザープール] を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。

3. まだ有効にしていない場合は、[アプリケーションの統合] タブから [高度なセキュリティ機能] を有効にします。
4. [] (ユーザープールのプロパティ) タブを選択し、[Lambda triggers] (Lambda トリガー) を検索します。
5. トークン生成前トリガーを追加または編集します。
6. [Lambda 関数を割り当てる] で Lambda 関数を選択します。
7. [基本機能 + アクセストークンのカスタマイズ] の [トリガーイベントバージョン] を選択します。この設定により、Amazon Cognito が関数に送信するリクエストパラメータが更新され、アクセストークンをカスタマイズするためのフィールドが含まれるようになります。

User pools API

トークン生成前の Lambda トリガーでアクセストークンのカスタマイズをサポートするには

CreateUserPool または UpdateUserPool API リクエストを生成します。デフォルト値に設定しないすべてのパラメータには、値を指定する必要があります。詳細については、「ユーザープール設定の更新」を参照してください。

リクエストの `LambdaVersion` パラメータに以下の内容を含めます。`LambdaVersion` の値が `V2_0` である場合、ユーザープールはアクセストークンをカスタマイズするためのパラメータを追加します。特定の関数バージョンを呼び出すには、関数バージョンを `LambdaArn` の値とする Lambda 関数 ARN を使用します。

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V2_0"
},
```

トピック

- トークン生成前の Lambda トリガーのソース
- トークン生成前の Lambda トリガーのパラメータ
- トークン生成前トリガーイベントバージョン 2 の例:クレーム、スコープ、グループの追加と非表示
- トークン生成前イベントバージョン 2 の例: 複雑なオブジェクトでクレームを追加する
- トークン生成前イベントバージョン 1 の例: 新しいクレームの追加と既存のクレームの非表示

- トークン生成前イベントバージョン 1 の例: ユーザーグループのメンバーシップの変更

トークン生成前の Lambda トリガーのソース

triggerSource 値	イベント
TokenGeneration_HostedAuth	認証時に Amazon Cognito のホストされた UI のサインインページから呼び出されます。
TokenGeneration_Authentication	ユーザー認証フローが完了した後に呼び出されます。
TokenGeneration_NewPasswordChallenge	管理者によってユーザーが作成された後に呼び出されます。このフローは、ユーザーが一時パスワードを変更する必要があるときに呼び出されます。
TokenGeneration_AuthenticateDevice	ユーザーデバイスの認証の終了時に呼び出されます。
TokenGeneration_RefreshTokens	ユーザーが ID およびアクセスのトークンを更新しようとしたときに呼び出されます。

トークン生成前の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。トークン生成前の Lambda トリガーをユーザープールに追加するときに、トリガーバージョンを選択できます。このバージョンにより、Amazon Cognito がアクセストークンをカスタマイズするための追加パラメータを含むリクエストを Lambda 関数に渡すかどうかが決まります。

Version 1

バージョン 1 トークンは、ID トークンでグループメンバーシップ、IAM ロール、および新しいクレームを設定できます。

```
{
    "request": {
```

```
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
            "claimsToSuppress": [
                "string",
                "string"
            ],
            "groupOverrideDetails": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Version 2

バージョン 2 リクエストイベントは、アクセストークンをカスタマイズするフィールドを追加します。また、レスポンスオブジェクトの複雑な`claimsToOverride`データ型に対するサポートも追加されています。Lambda 関数は、 の値で次のタイプのデータを返すことができます`claimsToOverride`。

- 文字列
- 数
- ブール値
- 文字列、数値、ブール値、またはこれらの組み合わせの配列
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
            "preferredRole": "string"
        },
        "clientMetadata": {
            "string": "string"
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
```

```
            }
        }
    }
}
```

トークン生成前のリクエストパラメータ

名前	説明	トリガーイベントの最小バージョン
userAttributes	ユーザープール内のユーザープロファイルの属性。	1
groupConfiguration	現在のグループ設定を含む入力オブジェクト。このオブジェクトには、`groupsToOverride` 、`iamRolesToOverride` 、および `preferredRole` が含まれています。	1
groupsToOverride	ユーザーがメンバーになっているユーザープールグループ。	1
iamRolesToオーバーライド	ユーザープールグループを AWS Identity and Access Management (IAM) ロールに関連付けることができます。この要素は、ユーザーがメンバーになっているグループのすべての IAM ロールのリストです。	1
preferredRole	ユーザープールグループには、優先順位を設定できます。この要素には、`groupsToOverride` 要素内で優先順位が最も高いグループの IAM ロールの名前が含まれます。	1
clientMetadata	トークン生成前のトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。 このデータを Lambda 関数に渡すには、 ClientMetadataAdminRespondToAuthChallengeおよび RespondToAuthChallenge API オペレーションで パラメータを使用します。Amazon Cognito は、トークン生成前関数に渡	1

名前	説明	トリガーイベントの最小バージョン
	すリクエストに AdminInitiateAuth および InitiateAuth API オペレーションの `ClientMetadata` パラメータからのデータを含めません。	
スコープ	ユーザーの OAuth 2.0 スコープ。アクセストークンに含まれるスコープは、ユーザーがリクエストし、アプリクライアントに発行を許可したユーザープールの標準スコープとカスタムスコープです。	2

トークン生成前のレスポンスパラメータ

名前	説明	トリガーイベントの最小バージョン
claimsOverrideDetails	`V1_0` トリガーイベントのすべての要素を格納するコンテナ。	1
claimsAndScopeOverrideDetails	`V2_0` トリガーイベントのすべての要素を格納するコンテナ。	2
idTokenGeneration	ユーザーの ID トークンで上書き、追加、または抑制するクレーム。ID トークンのカスタマイズ値に対するこの親はバージョン 2 のイベントにのみ表示されますが、子要素はバージョン 1 のイベントに表示されます。	2
accessTokenGeneration	ユーザーのアクセストークンで上書き、追加、または抑制するクレームとスコープ。アクセストークンのカスタマイズ値に対するこの親は、バージョン 2 のイベントにのみ表示されます。	2

名前	説明	トリガーイベントの最小バージョン
claimsToAddOrOverride	追加または変更する 1 つ以上のクレームとその値のマップ。グループ関連のクレームには、代わりに `groupOverrideDetails` を使用します。 バージョン 2 のイベントの場合、この要素は `accessTokenGeneration` と `idTokenGeneration` の両方の下に表示されます。	1-*
claimsToSuppress	Amazon Cognito で抑制するクレームのリスト。関数がクレーム値の非表示と置換の両方を行う場合、Amazon Cognito はクレームを非表示にします。 バージョン 2 のイベントの場合、この要素は `accessTokenGeneration` と `idTokenGeneration` の両方の下に表示されます。	1

名前	説明	トリガーイベントの最小バージョン
groupOverrideDetails	現在のグループ設定を含む出力オブジェクト。このオブジェクトには、groupsToOverride 、iamRolesToOverride 、および preferredRole が含まれています。 この関数は、groupOverrideDetails オブジェクトを、指定したオブジェクトに置き換えます。レスポンスで空または Null オブジェクトを返すと、Amazon Cognito はグループを非表示にします。既存のグループ設定を同じままにするには、リクエストの groupConfiguration オブジェクトの値をレスポンスの groupOverrideDetails オブジェクトにコピーします。その後、サービスに渡します。 Amazon Cognito ID とアクセストークンには、両方とも cognito:groups クレームが含まれています。groupOverrideDetails オブジェクトは、アクセストークンおよび ID トークンの cognito:groups クレームを置き換えます。	1
scopesToAdd	ユーザーのアクセストークンで scope クレームに追加する OAuth 2.0 スコープのリスト。1 つ以上の空白文字を含むスコープ値を追加することはできません。	2
scopesToSuppress	ユーザーのアクセストークンで scope クレームから削除する OAuth 2.0 スコープのリスト。	2

* バージョン 1 イベントへのレスポンスオブジェクトは文字列を返すことができます。バージョン 2 イベントへのレスポンスオブジェクトは、複雑なオブジェクト を返す可能性があります。

トークン生成前トリガーイベントバージョン 2 の例:クレーム、スコープ、グループの追加と非表示

この例では、ユーザーのトークンに以下の変更を行います。

1. ID トークンに family_name として Doe を設定します。
2. ID トークンに email クレームと phone_number クレームが表示されないようにします。
3. ID トークンの cognito:roles クレームを "arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB" に設定します。
4. ID トークンの cognito:preferred_role クレームを arn:aws:iam::123456789012:role/sns_caller に設定します。
5. スコープとして openid、email、solar-system-data/asteroids.add をアクセストークンに追加します。
6. アクセストークンの phone_number スコープと aws.cognito.signin.user.admin スコープを非表示にします。phone_number を削除すると、userInfo からユーザーの電話番号が取得されなくなります。aws.cognito.signin.user.admin を削除すると、ユーザーが Amazon Cognito ユーザープール API を使用して自分のプロファイルを読み取ったり変更したりするための API リクエストが禁止されます。

   > **Note**
   >
   > phone_number をスコープから削除した場合、アクセストークンの残りのスコープに openid と少なくとも 1 つの標準スコープが含まれていれば、ユーザーの電話番号のみが取得できなくなります。詳細については、「スコープについて」を参照してください。

7. ID トークンとアクセストークンの cognito:groups クレームを "new-group-A","new-group-B","new-group-C" に設定します。

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
    },
        "claimsToSuppress": [
          "email",
          "phone_number"
    ]
      },
```

```
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
    "new-group-A",
    "new-group-B",
    "new-group-C"
  ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role",
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
```

```
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

トークン生成前イベントバージョン 2 の例: 複雑なオブジェクトでクレームを追加する

この例では、ユーザーのトークンに以下の変更を行います。

1. ID トークンに数値、文字列、ブール型、JSON 型のクレームを追加します。これは、バージョン 2 のトリガーイベントが ID トークンで使用できるようにする唯一の変更です。
2. アクセストークンに数値、文字列、ブール型、JSON 型のクレームを追加します。
3. アクセストークンに 3 つのスコープを追加します。
4. ID トークンemailとアクセストークンの および subクレームを非表示にします。

5. アクセストークンのaws.cognito.signin.user.adminスコープを非表示にします。

JavaScript

```
export const handler = function(event, context) {

    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
    claims["exponentTest"] = 1.7976931348623157E308;
    claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
 true];
    claims["longStringTest"] = "\{\
        \"first_json_block\": \{\
            \"key_A\": \"value_A\",\
            \"key_B\": \"value_B\"\
        \},\
        \"second_json_block\": \{\
            \"key_C\": \{\
                \"subkey_D\": [\
                    \"value_D\",\
                    \"value_E\"\
                ],\
                \"subkey_F\": \"value_F\"\
            \},\
            \"key_G\": \"value_G\"\
        \}\
    \}";
    claims["jsonTest"] = {
     "first_json_block": {
      "key_A": "value_A",
      "key_B": "value_B"
     },
     "second_json_block": {
      "key_C": {
       "subkey_D": [
        "value_D",
        "value_E"
       ],
       "subkey_F": "value_F"
      },
```

```
      "key_G": "value_G"
     }
    };
    event.response = {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"],
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v ===
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
```

```
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED"
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

トークン生成前イベントバージョン 1 の例: 新しいクレームの追加と既存のクレームの非表示

この例では、トークン生成前の Lambda 関数でバージョン 1 のトリガーイベントを使用して、新しいクレームを追加し、既存のクレームを抑制します。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      claimsToAddOrOverride: {
        my_first_attribute: "first_value",
        my_second_attribute: "second_value",
      },
```

```
      claimsToSuppress: ["email"],
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。コード例 ではリクエストパラメータを処理しないため、空のリクエストでテストイベントを使用できます。共通のリクエストパラメータの詳細については、「ユーザープールの Lambda トリガーイベント」を参照してください。

JSON

```
{
  "request": {},
  "response": {}
}
```

トークン生成前イベントバージョン 1 の例: ユーザーグループのメンバーシップの変更

この例では、トークン生成前の Lambda 関数でバージョン 1 のトリガーイベントを使用し、ユーザーのグループメンバーシップを変更します。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
```

```
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "request": {},
  "response": {}
}
```

ユーザー移行の Lambda トリガー

パスワードによるサインイン時にユーザーがユーザープールに存在しない場合、またはユーザーがパスワードを忘れた場合のフローにいる場合に、Amazon Cognito は、このトリガーを呼び出します。Amazon Cognito は、Lambda 関数が正常に値を返した後でユーザーをユーザープールに作成します。ユーザー移行の Lambda トリガーを使用した認証フローの詳細については、「ユーザー移行の Lambda トリガーを使用したユーザープールへのユーザーのインポート」を参照してください。

サインイン時、またはパスワードを忘れた場合のフロー中に、ユーザーを既存のユーザーディレクトリから Amazon Cognito ユーザープールに移行するには、この Lambda トリガーを使用します。

トピック

- ユーザー移行の Lambda トリガーのソース
- ユーザー移行の Lambda トリガーのパラメータ

- 例: 既存のパスワードを使用したユーザーの移行

ユーザー移行の Lambda トリガーのソース

triggerSource 値	イベント
UserMigration_Authentication	サインイン時のユーザーの移行。
UserMigration_ForgotPassword	パスワードを忘れた場合のフロー実行時のユーザー移行

ユーザー移行の Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "userName": "string",
    "request": {
        "password": "string",
        "validationData": {
            "string": "string",
            . . .
        },
        "clientMetadata": {
            "string": "string",
       . . .
        }
    },
    "response": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "finalUserStatus": "string",
        "messageAction": "string",
        "desiredDeliveryMediums": [ "string", . . .],
        "forceAliasCreation": boolean,
```

```
        "enableSMSMFA": boolean
    }
}
```

ユーザー移行リクエストパラメータ

userName

ユーザーがサインイン時に入力するユーザー名。

password

ユーザーがサインイン時に入力するパスワード。Amazon Cognito は、パスワードを忘れた場合のフローによって開始されたリクエストでこの値を送信しません。

validationData

ユーザーサインインリクエストに検証データを含む 1 つ以上のキーバリューペア。このデータを Lambda 関数に渡すには、InitiateAuth および AdminInitiateAuth API アクションで ClientMetadata パラメータを使用できます。

clientMetadata

ユーザー移行のトリガーの Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキーバリューペア。このデータを Lambda 関数に渡すには、AdminRespondToAuthChallenge および ForgotPassword API アクションで ClientMetadata パラメータを使用できます。

ユーザー移行レスポンスパラメータ

userAttributes

このフィールドは必須です。

このフィールドには、Amazon Cognito がユーザープール内のユーザープロファイルに保存され、ユーザー属性として使用される、名前と値のペアが 1 つ以上含まれている必要があります。標準およびカスタムの両方のユーザー属性を含めることができます。カスタム属性は、標準属性と区別するために、`custom:` プレフィックスを必要とします。詳細については、「カスタム属性」を参照してください。

> **Note**
>
> パスワードを忘れた場合のフローでパスワードをリセットするには、ユーザーに検証済みの E メールアドレスまたは電話番号のどちらかが必要です。Amazon Cognito は、ユーザー属性にある E メールアドレスまたは電話番号宛てに、パスワードのリセットコードが含まれるメッセージを送信します。

属性	要件
ユーザープールの作成時に必須とマークした属性	移行時に必須属性が見当たらない場合は、Amazon Cognito はデフォルト値を使用します。
`username`	サインイン用のユーザー名に加えてエイリアス属性を使用してユーザープールを設定しており、ユーザーがユーザー名として有効なエイリアスを入力している場合は、必須です。エイリアスの値は、E メールアドレス、優先ユーザー名、電話番号です。 リクエストとユーザープールがエイリアスの要件を満たしている場合、関数からのレスポンスは、受け取った `username` パラメータをエイリアス属性に割り当てる必要があります。また、レスポンスは `username` 属性に自身の値を割り当てる必要があります。ユーザープールが、受け取った `username` をエイリアスにマッピングするのに必要な条件を満たしていない場合、レスポンスの `username` パラメータはリクエストと完全に一致するか、省略する必要があります。 **Note** `username` はユーザープールで一意である必要があります。

finalUserStatus

このパラメータを `CONFIRMED` に設定すると、ユーザーが以前のパスワードでサインインできるように自動確認することができます。ユーザーを `CONFIRMED` に設定した場合は、サインインす

る前に追加のアクション操作を実行する必要はありません。この属性を CONFIRMED に設定しない場合は、RESET_REQUIRED に設定されます。

RESET_REQUIRED の finalUserStatus は、サインイン時の移行後、ユーザーはすぐにパスワードを変更する必要があり、クライアントアプリは認証フロー中に PasswordResetRequiredException を処理する必要があることを意味します。

> **Note**
>
> Amazon Cognito は、Lambda トリガーを使用して移行中にユーザープールに設定したパスワード強度ポリシーを強制しません。パスワードが設定したパスワードポリシーを満たさない場合でも、Amazon Cognito は引き続きユーザーを移行できるように、パスワードを受け入れます。パスワード強度ポリシーを施行してポリシーを満たさないパスワードを拒否するには、コード内のパスワード強度を検証して、パスワードがポリシーを満たさない場合は、finalUserStatus を RESET_REQUIRED に設定します。

messageAction

このパラメータを SUPPRESS に設定すると、通常、Amazon Cognito は新規ユーザーに送信するウェルカムメッセージの送信を拒否することができます。関数がこのパラメータを返さない場合、Amazon Cognito はウェルカムメッセージを送信します。

desiredDeliveryMediums

このパラメータを EMAIL に設定すると E メールで、SMS に設定すると SMS でウェルカムメッセージを送信できます。関数がこのパラメータを返さない場合、Amazon Cognito は SMS でウェルカムメッセージを送信します。

forceAliasCreation

このパラメータを TRUE に設定した場合、UserAttributes パラメータの E メールアドレスまたは電話番号がすでに別のユーザーのエイリアスとして存在していると、API の呼び出しで以前のユーザーのエイリアスが新しく作成されたユーザーに移行されます。以前のユーザーはそのエイリアスを使用してログインできなくなります。

このパラメータを、FALSE に設定し、エイリアスが存在する場合、Amazon Cognito はユーザーを移行せず、クライアントアプリにエラーを返します。

このパラメータを返さない場合、Amazon Cognito はその値が「false」であると見なします。

enableSMSMFA

このパラメータを true に設定すると、移行したユーザーはサインインするために SMS テキストメッセージの多要素認証 (MFA) を完了することが必要になります。ユーザープールで MFA が有効になっている必要があります。リクエストパラメータのユーザー属性には、電話番号を含める必要があります。電話番号がないと、そのユーザーの移行は失敗します。

例: 既存のパスワードを使用したユーザーの移行

この例の Lambda 関数は、既存のパスワードを使用してユーザーを移行し、Amazon Cognito からのウェルカムメッセージを抑制します。

Node.js

```
const validUsers = {
  belladonna: { password: "Test123", emailAddress: "bella@example.com" },
};

// Replace this mock with a call to a real authentication service.
const authenticateUser = (username, password) => {
  if (validUsers[username] && validUsers[username].password === password) {
    return validUsers[username];
  } else {
    return null;
  }
};

const lookupUser = (username) => {
  const user = validUsers[username];

  if (user) {
    return { emailAddress: user.emailAddress };
  } else {
    return null;
  }
};

const handler = async (event) => {
  if (event.triggerSource == "UserMigration_Authentication") {
    // Authenticate the user with your existing user directory service
    const user = authenticateUser(event.userName, event.request.password);
    if (user) {
```

```
      event.response.userAttributes = {
        email: user.emailAddress,
        email_verified: "true",
      };
      event.response.finalUserStatus = "CONFIRMED";
      event.response.messageAction = "SUPPRESS";
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Look up the user in your existing user directory service
    const user = lookupUser(event.userName);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        // Required to enable password-reset code to be sent to user
        email_verified: "true",
      };
      event.response.messageAction = "SUPPRESS";
    }
  }

  return event;
};

export { handler };
```

カスタムメッセージの Lambda トリガー

Amazon Cognito は、E メールまたは電話による確認メッセージ、または多要素認証 (MFA) コードを送信する前にこのトリガーを呼び出します。カスタムメッセージトリガーを使用して、メッセージを動的にカスタマイズできます。静的なカスタムメッセージは、Amazon Cognito コンソールの [Message Customizations] (メッセージのカスタマイズ) タブで編集できます。

リクエストには `codeParameter` が含まれます。これは、Amazon Cognito がユーザーに配信するコードのプレースホルダーとなる文字列です。メッセージ本文で検証コードを表示する位置に `codeParameter` 文字列を挿入します。Amazon Cognito がこのレスポンスを受信すると、Amazon Cognito が `codeParameter` 文字列を実際の検証コードに置き換えます。

> **Note**
>
> `CustomMessage_AdminCreateUser` トリガーソースがあるカスタムメッセージの Lambda 関数 はユーザー名と検証コードを返します。管理者が作成したユー

ザーはユーザー名とコードの両方を受け取る必要があるため、関数からの応答には `request.usernameParameter` と `request.codeParameter`の両方を含める必要があります。

トピック

- カスタムメッセージの Lambda トリガーのソース
- カスタムメッセージの Lambda トリガーのパラメータ
- サインアップのカスタムメッセージの例
- 管理者作成ユーザーのカスタムメッセージの例

カスタムメッセージの Lambda トリガーのソース

triggerSource 値	イベント
`CustomMessage_SignUp`	カスタムメッセージ - サインアップ後に確認コードを送信するため。
`CustomMessage_AdminCreateUser`	カスタムメッセージ – 新規ユーザーに一時パスワードを送信するため。
`CustomMessage_ResendCode`	カスタムメッセージ – 既存ユーザーに確認コードを再送するため。
`CustomMessage_ForgotPassword`	カスタムメッセージ - 忘れたパスワードのリクエスト用の確認コードを送信するため。
`CustomMessage_UpdateUserAttribute`	カスタムメッセージ - ユーザーの E メールまたは電話番号が変更されると、このトリガーは検証コードをそのユーザーに自動的に送信します。他の属性には使用できません。
`CustomMessage_VerifyUserAttribute`	カスタムメッセージ – ユーザーが手動で新しい E メールや電話番号の認証コードをリクエストすると、このトリガーからユーザーに認証コードが送信されます。

triggerSource 値	イベント
CustomMessage_Authentication	カスタムメッセージ - 認証時に MFA コードを送信するため。

カスタムメッセージの Lambda トリガーのパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        }
    },
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

カスタムメッセージリクエストパラメータ

userAttributes

ユーザー属性を表す 1 つ以上の名前 - 値ペア。

codeParameter

カスタムメッセージで、検証コードのプレースホルダーとして使用する文字列。

usernameParameter

ユーザー名。Amazon Cognito は、管理者が作成したユーザーからのリクエストにこのパラメータを含めます。

clientMetadata

カスタムメッセージのトリガーに指定する Lambda 関数へのカスタム入力として提供できる 1 つ、または複数のキー/値ペア。カスタムメッセージ関数を呼び出すリクエストには、AdminInitiateAuthおよび InitiateAuth API オペレーションの ClientMetadata パラメータに渡されたデータは含まれません。このデータを Lambda 関数に渡すには、次の API アクションで ClientMetadata パラメータを使用できます。

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode
- SignUp
- UpdateUserAttributes

カスタムメッセージレスポンスパラメータ

レスポンスで、ユーザーへのメッセージに使用するカスタムテキストを指定します。Amazon Cognito がこれらのパラメータに適用する文字列制約については、「」を参照してくださいMessageTemplateType。

smsMessage

ユーザーに送信されるカスタム SMS メッセージ。リクエストで受信する `codeParameter` 値を含める必要があります。

emailMessage

ユーザーに送信するカスタム E メールメッセージ。`emailMessage` パラメータで HTML 書式を使用できます。リクエストで受信される `codeParameter` 値を変数 `{####}` として含める必要があります。Amazon Cognito は、ユーザープールの `EmailSendingAccount` 属性が `DEVELOPER` の場合のみ、`emailMessage` パラメータ

を使用できます。ユーザープールの EmailSendingAccount 属性が、DEVELOPER ではなく、emailMessage パラメータが返されると、Amazon Cognito は 400 エラーコード com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException を生成します。Amazon Simple Email Service (Amazon SES) を使用して E メールメッセージを送信する場合、ユーザープールの EmailSendingAccount 属性は DEVELOPER です。それ以外の場合、値は COGNITO_DEFAULT です。

emailSubject

カスタムメッセージの件名。emailSubject パラメータは、ユーザープールの 属性が の場合にのみ EmailSendingAccount使用できますDEVELOPER。ユーザープールの EmailSendingAccount 属性が DEVELOPER ではなく、Amazon Cognito が emailSubject パラメータを返した場合、Amazon Cognito は 400 エラーコード com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException を生成します。Amazon Simple Email Service (Amazon SES) を使用して E メールメッセージを送信する場合、ユーザープールの EmailSendingAccount 属性は DEVELOPER です。それ以外の場合、値は COGNITO_DEFAULT です。

サインアップのカスタムメッセージの例

この Lambda 関数の例は、サービスでアプリがユーザーへの検証コードの送信が必要とされる場合に、E メールまたは SMS メッセージをカスタマイズします。

Amazon Cognito は、登録後、検証コードの再送信時、パスワードを忘れた場合、ユーザー属性の検証時といった複数のイベントで Lambda トリガーを呼び出すことができます。レスポンスには、SMS と E メールの両方のメッセージが含まれます。メッセージにはコードのパラメータ "####" を含める必要があります。このパラメータは、ユーザーが受け取る検証コードのプレースホルダーです。

E メールメッセージの最大長は 20,000 UTF-8 文字です。この長さには検証コードが含まれます。これらの E メールメッセージでは、HTML タグを使用できます。

SMS メッセージの最大長は 140 UTF-8 文字です。この長さには検証コードが含まれます。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_SignUp") {
    const message = `Thank you for signing up. Your confirmation code is
 ${event.request.codeParameter}.`;
```

```
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service.";
  }
  return event;
};

export { handler };
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_SignUp/CustomMessage_ResendCode/
CustomMessage_ForgotPassword/CustomMessage_VerifyUserAttribute",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter>"
      "emailMessage": "<custom message to be sent in the message with code
 parameter>"
      "emailSubject": "<custom email subject>"
  }
```

```
}
```

管理者作成ユーザーのカスタムメッセージの例

このカスタムメッセージの Lambda 関数の例に Amazon Cognito が送信したリクエストのtriggerSource値は で、CustomMessage_AdminCreateUserユーザー名と一時パスワードがあります。関数は、リクエスト${event.request.codeParameter}の一時パスワードとリクエストのユーザー名${event.request.usernameParameter}から を入力します。

カスタムメッセージは、レスポンスオブジェクトemailMessageの codeParameterと usernameParameter に smsMessageと の値を挿入する必要があります。この例では、関数はレスポンスフィールド event.response.smsMessageおよび に同じメッセージを書き込みますevent.response.emailMessage。

E メールメッセージの最大長は 20,000 UTF-8 文字です。この長さには検証コードが含まれます。これらの E メールでは HTML タグを使用できます。SMS メッセージの最大長は 140 UTF-8 文字です。この長さには検証コードが含まれます。

レスポンスには、SMS と E メールの両方のメッセージが含まれます。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_AdminCreateUser") {
    const message = `Welcome to the service. Your user name is
 ${event.request.usernameParameter}. Your temporary password is
 ${event.request.codeParameter}`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service";
  }
  return event;
};

export { handler }
```

Amazon Cognito は Lambda 関数にイベント情報を渡します。関数はレスポンスで、同じイベントオブジェクトを変更と共に Amazon Cognito に返します。Lambda コンソールで、Lambda トリガーに

関連するデータを使用したテストイベントをセットアップできます。以下は、このコードサンプルのテストイベントです。

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_AdminCreateUser",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####",
      "usernameParameter": "username"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

カスタム送信者の Lambda トリガー

Amazon Cognito ユーザープールは、サードパーティーの E メールと SMS 通知をアクティブ化するために、CustomEmailSender および CustomSMSSender の Lambda トリガーを提供します。Lambda 関数コード内からユーザーに通知を送信するには、希望の SMS プロバイダーと E メールプロバイダーを選択して使用することができます。Amazon Cognito が、確認コード、検証コード、または一時パスワードなどの通知をユーザーに送信する必要がある場合、イベントは設定された

Lambda 関数を有効化します。Amazon Cognito は、有効化された Lambda 関数にコードと一時パスワード (シークレット) を送信します。Amazon Cognito は、これらのシークレットを AWS KMS カスタマーマネージドキーと AWS Encryption SDK で暗号化します。AWS Encryption SDK は、クライアント側暗号化ライブラリで、汎用データの暗号化および復号に役立ちます。

> **Note**
>
> これらの Lambda トリガーを使用するようにユーザープールを設定するには、AWS CLI または SDK を使用します。これらの設定は、Amazon Cognito コンソールからは利用できません。

CustomEmailSender

Amazon Cognito は、E メール通知をユーザーに送信するためにこのトリガーを呼び出します。

CustomSMSSender

Amazon Cognito は、SMS 通知をユーザーに送信するためにこのトリガーを呼び出します。

リソース

以下のリソースは、`CustomEmailSender` および `CustomSMSSender` トリガーを使用するために役立ちます。

AWS KMS

AWS KMS は AWS KMS キーの作成と制御を行うマネージドサービスです。これらのキーは、データを暗号化します。詳細については、「AWS Key Management Service とは何ですか?」を参照してください。

KMS キー

KMS キーは、暗号化キーの論理表現です。KMS キーには、キー ID、作成日、説明、キーステータスなどのメタデータが含まれます。KMS キーには、データの暗号化と復号に使用されるキーマテリアルも含まれています。詳細については、「AWS KMS キーの削除」を参照してください。

対称 KMS キー

対称 KMS キーは、暗号化されていない AWS KMS を終了しない 256 ビットの暗号化キーです。対称 KMS キーを使用するには、AWS KMS を呼び出す必要があります。Amazon Cognito は対称

キーを使用します。同じキーで暗号化と復号化が行われます。詳細については、「KMS キーの削除」を参照してください。

カスタム E メール送信者の Lambda トリガー

ユーザープールにカスタム E メール送信者トリガーを割り当てると、ユーザーイベントで E メールメッセージの送信が必要になった時点で、Amazon Cognito がデフォルトの動作ではなく Lambda 関数を呼び出します。カスタム送信者トリガーを使用すると、AWS Lambda 関数が、選択した方法とプロバイダーを通じてユーザーに e メール通知を送信できます。関数のカスタムコードは、すべての e メールメッセージを、ユーザープールから処理して配信する必要があります。

> **Note**
>
> 現在、Amazon Cognito コンソールでカスタム送信者のトリガーを割り当てることはできません。`CreateUserPool` または `UpdateUserPool` API リクエストで `LambdaConfig` パラメータを使用してトリガーを割り当てることができます。

このトリガーをセットアップするには、以下のステップを実行します。

1. AWS Key Management Service (AWS KMS) で対称暗号化キーを作成します。Amazon Cognito は、シークレット (一時的なパスワード、認可コード、および確認コード) を生成し、この KMS キーを使用してシークレットを暗号化します。次に、Lambda 関数で Decrypt API オペレーションを使用して、シークレットを復号化し、プレーンテキストでユーザーに送信できます。AWS Encryption SDK は、関数内の AWS KMS オペレーションに有用なツールです。
2. カスタム送信者のトリガーとして割り当てる Lambda 関数を作成します。Lambda 関数ロールに対して、KMS キーへの `kms:Decrypt` アクセス許可を付与します。
3. Amazon Cognito サービスプリンシパルに、Lambda 関数を呼び出すための `cognito-idp.amazonaws.com` へのアクセス権を付与します。
4. カスタム配信メソッドまたはサードパーティープロバイダーにメッセージを転送する Lambda 関数コードを書き込みます。ユーザーの認証コードまたは確認コードを配信するには、Base64 がリクエストで `code` パラメーターの値をデコードして復号化します。このオペレーションでは、メッセージに含める必要があるプレーンテキストのコードまたはパスワードが生成されます。
5. カスタム送信者 Lambda トリガーを使用するようにユーザープールを更新します。カスタム送信者トリガーを使用してユーザープールを更新または作成する IAM プリンシパルには、KMS キーの許可を作成する権限が必要です。以下の `LambdaConfig` スニペットは、SMS とメール送信者関数を割り当てます。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

カスタム E メール送信者の Lambda トリガーパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

カスタム E メール送信者のリクエストパラメータ

type

リクエストバージョン。カスタム E メール送信者イベントの場合、この文字列の値は常に `customEmailSenderRequestV1` です。

コード

関数が復号してユーザーに送信できる暗号化されたコード。

clientMetadata

カスタム E メール送信者 Lambda 関数トリガーへのカスタム入力として提供できる 1 つ以上のキーバリューペア このデータを Lambda 関数に渡すには、AdminRespondToAuthChallenge および RespondToAuthChallenge API アクションで ClientMetadata パラメータを使用します。Amazon Cognito は、認証後関数に渡すリクエストの AdminInitiateAuth および InitiateAuth API オペレーションの ClientMetadata パラメータからのデータを含めません。

userAttributes

ユーザー属性を表す 1 つ以上のキーバリューペア。

カスタム E メール送信者の応答パラメータ

Amazon Cognito は、カスタム E メール送信者のレスポンスに追加の情報が返されることを期待しません。関数では、API オペレーションを使用してリソースをクエリして変更したり、イベントメタデータを外部システムに記録することができます。

カスタム E メール送信者の Lambda トリガーのアクティブ化

カスタムロジックを使用してユーザープールの E メールメッセージを送信するカスタム E メール送信者トリガーを設定するには、次のようにトリガーをアクティブ化します。以下の手順では、カスタム E メールトリガー、カスタム SMS トリガー、またはその両方をユーザープールに割り当てます。カスタム E メール送信者トリガーを追加すると、Amazon Cognito は常に E メールアドレスなどのユーザー属性とワンタイムコードを Lambda 関数に送信します (それ以外の場合は、Amazon Simple Email Service で E メールメッセージを送信します)。

> **Important**
>
> Amazon Cognito は、ユーザーの一時パスワードで `<` (<) および `>` (>) などの予約文字を HTML エスケープします。これらの文字は、Amazon Cognito がカスタム E メール送信者機能に送信する一時パスワードには表示される場合がありますが、一時的な確認コードには表示されません。一時パスワードを送信するには、Lambda 関数がパスワードを復号した後、ユーザーにメッセージを送信する前に、これらの文字をアンエスケープする必要があります。

1. AWS KMS で暗号化キーを作成します。このキーは、Amazon Cognito が生成する一時パスワードと認可コードを暗号化します。次に、カスタム送信者の Lambda 関数でこれらのシークレットを復号して、プレーンテキストでユーザーに送信できます。
2. Amazon Cognito サービスプリンシパルに KMS キーでコードを暗号化するための `cognito-idp.amazonaws.com` アクセス権を付与します。

   次のリソースベースのポリシーを KMS キーに適用します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. カスタム送信者トリガーのための Lambda 関数を作成します。Amazon Cognito は AWS 暗号化 SDK を使用してシークレット (一時パスワードまたはユーザーの API リクエストを承認するコード) を暗号化します。
   - 少なくとも KMS キーに対する `kms:Decrypt` アクセス許可を持つ IAM ロールを Lambda 関数に割り当てます。
4. Amazon Cognito サービスプリンシパルに、Lambda 関数を呼び出すための `cognito-idp.amazonaws.com` へのアクセス権を付与します。

   次の AWS CLI コマンドでは、Lambda 関数を呼び出すためのアクセス許可を Amazon Cognito に付与します。

```
        aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. メッセージを送信する Lambda 関数コードを作成します。Amazon Cognito は、シークレットをカスタム送信者 Lambda 関数に送信する前に、AWS Encryption SDK を使用してそれらを暗号化します。関数内でシークレットを復号し、関連するメタデータを処理します。次に、メッセージを配信するカスタム API に、コード、独自のカスタムメッセージ、宛先の電話番号を送信します。
6. AWS Encryption SDK を Lambda 関数に追加します。詳細については、「AWS Encryption SDK programming languages」(AWS Encryption SDK のプログラミング言語) を参照してください。Lambda パッケージを更新するには、次のステップを完了します。
   a. Lambda 関数を.zip ファイルとして AWS Management Console にエクスポートします。
   b. 関数を開いて、AWS Encryption SDK を追加します。詳細とダウンロードリンクについては、AWS Encryption SDK デベロッパーガイドの「AWS Encryption SDK プログラミング言語」を参照してください。
   c. SDK の依存関係を使用して関数を ZIP 圧縮し、その関数を Lambda にアップロードします。詳細については、AWS Lambda デベロッパーガイドの「Lambda 関数の .zip ファイルアーカイブとしてのデプロイ」を参照してください。
7. ユーザープールを更新してカスタム送信者の Lambda トリガーを追加します。UpdateUserPool API リクエストに CustomSMSSender または CustomEmailSender パラメータを含めます。API UpdateUserPool オペレーションには、ユーザープールのすべてのパラメータと、変更するパラメータが必要です。関連するすべてのパラメータを指定しない場合、Amazon Cognito は不足しているパラメータの値をデフォルトに設定します。次の例に示すように、ユーザープールに追加または保持するすべての Lambda 関数のエントリを含めます。詳細については、「ユーザープール設定の更新」を参照してください。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                     CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
```

```
                    CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                    KMSKeyID=key-id"
```

update-user-pool AWS CLI を使用してカスタム送信者の Lambda トリガーを削除するには、--lambda-config から CustomSMSSender または CustomEmailSender パラメータを省略し、ユーザープールで使用する他のすべてのトリガーを含めます。

UpdateUserPool API リクエストを使用してカスタム送信者の Lambda トリガーを削除するには、ユーザープール設定の残りを含むリクエスト本文から CustomSMSSender または CustomEmailSender パラメータを省略します。

コードサンプル

次の Node.js 例は、カスタム E メール送信者の Lambda 関数で E メールメッセージイベントを処理します。この例では、関数に 2 つの環境変数が定義されていることを前提としています。

KEY_ALIAS

ユーザーのコードを暗号化および復号化するために使用する KMS キーのエイリアス。

KEY_ARN

ユーザーのコードを暗号化および復号化するために使用する KMS キーの Amazon リソースネーム (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.
const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
```

```
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomEmailSender_SignUp'){
  //Send an email message to your user via a custom provider.
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomEmailSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomEmailSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomEmailSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AccountTakeOverNotification'){
 }
 return;
};
```

カスタム E メール送信者の Lambda トリガーのソース

以下の表には、Lambda コード内のカスタム E メールトリガーのソースに対するトリガーイベントが記載されています。

TriggerSource value	イベント
CustomEmailSender_SignUp	ユーザーがサインアップすると、Amazon Cognito がウェルカムメッセージを送信します。
CustomEmailSender_ForgotPassword	ユーザーがパスワードをリセットするコードを要求します。
CustomEmailSender_ResendCode	ユーザーがパスワードをリセットするための置換コードを要求します。

TriggerSource value	イベント
CustomEmailSender_UpdateUserAttribute	ユーザーが E メールアドレスまたは電話番号の属性を更新すると、Amazon Cognito は属性を検証するためのコードを送信します。
CustomEmailSender_VerifyUserAttribute	ユーザーが新しい E メールアドレスまたは電話番号属性を作成し、Amazon Cognito は属性を検証するコードを送信します。
CustomEmailSender_AdminCreateUser	ユーザープールに新しいユーザーを作成すると、Amazon Cognito から一時パスワードが送信されます。
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito は、ユーザーアカウントを引き継ぐ試みを検出し、ユーザーに通知を送信します。

カスタム SMS 送信者の Lambda トリガー

ユーザープールにカスタム SMS 送信者トリガーを割り当てると、ユーザーイベントで SMS メッセージの送信が必要になった時点で、Amazon Cognito がデフォルトの動作ではなく Lambda 関数を呼び出します。カスタム送信者トリガーを使用すると、 AWS Lambda 選択したメソッドとプロバイダーを通じて SMS 通知をユーザーに送信できます。関数のカスタムコードは、すべての SMS メッセージを、ユーザープールから処理して配信する必要があります。

> **Note**
>
> 現在、Amazon Cognito コンソールでカスタム送信者のトリガーを割り当てることはできません。`CreateUserPool` または `UpdateUserPool` API リクエストで `LambdaConfig` パラメータを使用してトリガーを割り当てることができます。

このトリガーをセットアップするには、以下のステップを実行します。

1. AWS Key Management Service () で対称暗号化キーを作成しますAWS KMS。Amazon Cognito は、シークレット (一時的なパスワード、認可コード、および確認コード) を生成し、この KMS キーを使用してシークレットを暗号化します。次に、Lambda 関数で Decrypt API オペレーショ

ンを使用して、シークレットを復号化し、プレーンテキストでユーザーに送信できます。AWS Encryption SDK は、 関数での AWS KMS オペレーションに役立つツールです。

2. カスタム送信者のトリガーとして割り当てる Lambda 関数を作成します。Lambda 関数ロールに対して、KMS キーへの `kms:Decrypt` アクセス許可を付与します。
3. Amazon Cognito サービスプリンシパルに、Lambda 関数を呼び出すための `cognito-idp.amazonaws.com` へのアクセス権を付与します。
4. カスタム配信メソッドまたはサードパーティープロバイダーにメッセージを転送する Lambda 関数コードを書き込みます。ユーザーの認証コードまたは確認コードを配信するには、Base64 がリクエストで `code` パラメーターの値をデコードして復号化します。このオペレーションでは、メッセージに含める必要があるプレーンテキストのコードまたはパスワードが生成されます。
5. カスタム送信者 Lambda トリガーを使用するようにユーザープールを更新します。カスタム送信者トリガーを使用してユーザープールを更新または作成する IAM プリンシパルには、KMS キーの許可を作成する権限が必要です。以下の `LambdaConfig` スニペットは、SMS とメール送信者関数を割り当てます。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

カスタム SMS 送信者の Lambda トリガーパラメータ

Amazon Cognito がこの Lambda 関数に渡すリクエストは、以下のパラメータと Amazon Cognito がすべてのリクエストに追加する共通パラメータを組み合わせたものです。

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
```

```
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

カスタム SMS 送信者のリクエストパラメータ

type

リクエストバージョン。カスタム SMS 送信者イベントの場合、この文字列の値は常に `customSMSSenderRequestV1` です。

コード

関数が復号してユーザーに送信できる暗号化されたコード。

clientMetadata

カスタム SMS 送信者 Lambda 関数トリガーへのカスタム入力として提供できる 1 つ以上のキーバリューペア このデータを Lambda 関数に渡すには、 AdminRespondToAuthChallengeおよび RespondToAuthChallenge API アクションで ClientMetadata パラメータを使用できます。Amazon Cognito は、認証後関数に渡すリクエストに、 AdminInitiateAuth および InitiateAuth API オペレーションの ClientMetadata パラメータからのデータを含めません。

userAttributes

ユーザー属性を表す 1 つ以上のキーバリューペア。

カスタム SMS 送信者の応答パラメータ

Amazon Cognito は、レスポンスに追加の返品情報を期待しません。関数では、API オペレーションを使用してリソースをクエリして変更したり、イベントメタデータを外部システムに記録することができます。

カスタム SMS 送信者の Lambda トリガーのアクティブ化

カスタムロジックを使用してユーザープールの SMS メッセージを送信するカスタム SMS 送信者トリガーを設定できます。以下の手順では、カスタム SMS トリガー、カスタム E メールトリ

ガー、またはその両方をユーザープールに割り当てます。カスタム SMS 送信者トリガーを追加すると、Amazon Cognito は常に Amazon Simple Notification Service で SMS メッセージを送信するデフォルトの動作ではなく、電話番号やワンタイムコードなどのユーザー属性を Lambda 関数に送信します。

> **Important**
>
> Amazon Cognito は、ユーザーの一時パスワードで `<` (<) および `>` (>) などの予約文字を HTML エスケープします。これらの文字は、Amazon Cognito がカスタム E メール送信者機能に送信する一時パスワードには表示される場合がありますが、一時的な確認コードには表示されません。一時パスワードを送信するには、Lambda 関数がパスワードを復号した後、ユーザーにメッセージを送信する前に、これらの文字をアンエスケープする必要があります。

1. AWS KMSで暗号化キーを作成します。このキーは、Amazon Cognito が生成する一時パスワードと認可コードを暗号化します。次に、カスタム送信者の Lambda 関数でこれらのシークレットを復号して、プレーンテキストでユーザーに送信できます。
2. Amazon Cognito サービスプリンシパルに KMS キーでコードを暗号化するための `cognito-idp.amazonaws.com` アクセス権を付与します。

   次のリソースベースのポリシーを KMS キーに適用します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
```

```
            }
        }
    }]
}
```

3. カスタム送信者トリガーのための Lambda 関数を作成します。Amazon Cognito は AWS 暗号化 SDK を使用してシークレット (一時パスワードまたはユーザーの API リクエストを承認するコード) を暗号化します。

   - 少なくとも KMS キーに対する `kms:Decrypt` アクセス許可を持つ IAM ロールを Lambda 関数に割り当てます。

4. Amazon Cognito サービスプリンシパルに、Lambda 関数を呼び出すための `cognito-idp.amazonaws.com` へのアクセス権を付与します。

   次の AWS CLI コマンドは、Lambda 関数を呼び出すアクセス許可を Amazon Cognito に付与します。

```
aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. メッセージを送信する Lambda 関数コードを作成します。Amazon Cognito は AWS Encryption SDK 、Amazon Cognito がシークレットをカスタム送信者 Lambda 関数に送信する前に、 を使用してシークレットを暗号化します。関数内でシークレットを復号し、関連するメタデータを処理します。次に、メッセージを配信するカスタム API に、コード、独自のカスタムメッセージ、宛先の電話番号を送信します。

6. を Lambda 関数 AWS Encryption SDK に追加します。詳細については、「AWS Encryption SDK programming languages」(AWS Encryption SDK のプログラミング言語) を参照してください。Lambda パッケージを更新するには、次のステップを完了します。

   a. Lambda 関数を.zip ファイルとして AWS Management Consoleにエクスポートします。

   b. 関数を開き、 を追加します AWS Encryption SDK。詳細とダウンロードリンクについては、AWS Encryption SDK デベロッパーガイドの「AWS Encryption SDK プログラミング言語」を参照してください。

   c. SDK の依存関係を使用して関数を ZIP 圧縮し、その関数を Lambda にアップロードします。詳細については、AWS Lambda デベロッパーガイドの「Lambda 関数の .zip ファイルアーカイブとしてのデプロイ」を参照してください。

7. ユーザープールを更新してカスタム送信者の Lambda トリガーを追加します。UpdateUserPool API リクエストに CustomSMSSender または CustomEmailSender パラメータを含めます。API UpdateUserPool オペレーションには、ユーザープールのすべてのパラメータと、変更するパラメータが必要です。関連するすべてのパラメータを指定しない場合、Amazon Cognito は不足しているパラメータの値をデフォルトに設定します。次の例に示すように、ユーザープールに追加または保持するすべての Lambda 関数のエントリを含めます。詳細については、「ユーザープール設定の更新」を参照してください。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

を使用してカスタム送信者の Lambda トリガーを削除するにはupdate-user-pool AWS CLI、 から CustomSMSSenderまたは CustomEmailSenderパラメータを省略し--lambda-config、ユーザープールで使用する他のすべてのトリガーを含めます。

UpdateUserPool API リクエストを使用してカスタム送信者の Lambda トリガーを削除するには、ユーザープール設定の残りを含むリクエスト本文から CustomSMSSender または CustomEmailSender パラメータを省略します。

コード例

次の Node.js 例は、カスタム SMS 送信者 Lambda 関数で SMS メッセージイベントを処理します。この例では、関数に 2 つの環境変数が定義されていることを前提としています。

KEY_ALIAS

ユーザーのコードを暗号化および復号化するために使用する KMS キーのエイリアス。

KEY_ARN

ユーザーのコードを暗号化および復号化するために使用する KMS キーの Amazon リソースネーム (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.

const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomSMSSender_SignUp'){
  //Send an SMS message to your user via a custom provider.
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomSMSSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomSMSSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomSMSSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AccountTakeOverNotification'){
 }
 return;
};
```

トピック

- カスタム SMS 送信者関数で SMS メッセージ機能を評価する
- カスタム SMS 送信者の Lambda トリガーのソース

カスタム SMS 送信者関数で SMS メッセージ機能を評価する

カスタム SMS 送信者 Lambda 関数は、ユーザープールが送信する SMS メッセージを受け入れ、関数はカスタムロジックに基づいてコンテンツを配信します。Amazon Cognito は関数に カスタム SMS 送信者の Lambda トリガーパラメータ を送信します。この情報を使用して、関数はユーザーが望むことを実行できます。例えば、Amazon Simple Notification Service (Amazon SNS) トピックにコードを送信できます。Amazon SNS トピックのサブスクライバーは、SMS メッセージ、HTTPS エンドポイント、または E メールアドレスです。

カスタム SMS 送信者 Lambda 関数を使用して Amazon Cognito SMS メッセージングのテスト環境を作成するには、 の aws-samples ライブラリ GitHubamazon-cognito-user-pooldevelopment-and-testing-withsms-redirected-to-emailの -- を参照してください。リポジトリには、新しいユーザープールを作成したり、既に持っているユーザープールを操作できる AWS CloudFormation テンプレートが含まれています。これらのテンプレートでは、Lambda 関数と Amazon SNS トピックが作成されます。テンプレートがカスタム SMS 送信者トリガーとして割り当てる Lambda 関数は、SMS メッセージを Amazon SNS トピックのサブスクライバーにリダイレクトします。

このソリューションをユーザープールにデプロイすると、Amazon Cognito が通常送信するすべてのメッセージが SMS メッセージングを介して送信され、代わりに Lambda 関数が中央の E メールアドレスに送信します。このソリューションを使用して、SMS メッセージをカスタマイズおよびプレビューし、Amazon Cognito が SMS メッセージを送信する原因となるユーザープールイベントをテストします。テストが完了したら、 CloudFormation スタックをロールバックするか、ユーザープールからカスタム SMS 送信者関数の割り当てを削除します。

> **Important**
>
> amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email のテンプレートを使用して本番環境を構築しないでください。ソリューション内のカスタム SMS 送信者 Lambda 関数は SMS メッセージをシミュレートしますが、Lambda 関数はそれらすべてを単一の中央の E メールアドレスに送信します。本番環境の Amazon Cognito ユーザープールで SMS メッセージを送信する前に、Amazon Cognito ユーザープール用の SMS メッセージ設定 に示す要件を満たす必要があります。

カスタム SMS 送信者の Lambda トリガーのソース

以下の表には、Lambda コード内のカスタム SMS トリガーのソースに対するトリガーイベントが記載されています。

TriggerSource value	イベント
CustomSMSSender_SignUp	ユーザーがサインアップすると、Amazon Cognito がウェルカムメッセージを送信します。
CustomSMSSender_ForgotPassword	ユーザーがパスワードをリセットするコードを要求します。
CustomSMSSender_ResendCode	ユーザーが登録を確認するための新しいコードをリクエストします。
CustomSMSSender_VerifyUserAttribute	ユーザーが新しい E メールアドレスまたは電話番号の属性を作成すると、Amazon Cognito は属性を検証するコードを送信します。
CustomSMSSender_UpdateUserAttribute	ユーザーが E メールアドレスまたは電話番号の属性を更新すると、Amazon Cognito は属性を検証するためのコードを送信します。
CustomSMSSender_Authentication	ユーザーが SMS 多要素認証 (MFA) を使用して設定されたユーザーがサインインします。
CustomSMSSender_AdminCreateUser	ユーザープールに新しいユーザーを作成すると、Amazon Cognito から一時パスワードが送信されます。

Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用

Amazon Cognito ユーザープールは、Amazon Cognito ユーザープールの分析を提供し、Amazon Pinpoint キャンペーンのユーザーデータを強化するために Amazon Pinpoint と統合されています。Amazon Pinpoint は、プッシュ通知を使用するモバイルアプリケーションでのユーザーエンゲージメントを促進するために、分析と的を絞ったキャンペーンを提供します。Amazon Cognito ユーザープールでの Amazon Pinpoint 分析のサポートを使用すると、Amazon Pinpoint コンソールでユーザープールへのサインアップ、サインイン、失敗した認証、デイリーアクティブユーザー数 (DAU)、および月間アクティブユーザー数 (MAU) を追跡できます。デバイスプラットフォーム、デ

バイスロケール、およびアプリケーションのバージョンなどのデータを異なる日付範囲や属性で表示できます。

また、アプリケーションにカスタム属性を設定することもできます。これらは、Amazon Pinpoint でユーザーを分割し、的を絞ったプッシュ通知をユーザーに送信するために使用できます。Amazon Cognito コンソールの [Analytics] (分析) タブで、[Share user attribute data with Amazon Pinpoint] (Amazon Pinpoint とユーザー属性データを共有する) を選択すると、Amazon Pinpoint は、ユーザーの E メールアドレスと電話番号用に追加のエンドポイントを作成します。

Amazon Cognito コンソールを使用してユーザープールの Amazon Pinpoint 分析をアクティブ化すると、Amazon Cognito がユーザープールの Amazon Pinpoint に API リクエストを送信するときに引き継ぐサービスにリンクされたロールも作成されます。分析設定を追加する IAM プリンシパルには、CreateServiceLinkedRole 権限が必要です。サービスにリンクされたロールは、AWSServiceRoleForAmazonCognitoIdp です。詳細については、「Amazon Cognito のサービスリンクロールの使用」を参照してください。

Amazon Cognito API で `AnalyticsConfiguration` をアプリケーションクライアントに適用する際、Amazon Pinpoint にカスタム IAM ロールを割り当てて、そのロールを引き継ぐ外部 ID を割り当てることができます。ロールは `cognito-idp` サービスプリンシパルを信頼している必要があり、ロール信頼ポリシーで外部 ID が必要な場合は、`AnalyticsConfiguration` と一致する必要があります。Amazon Pinpoint プロジェクトには、ロール `cognito-idp:Describe*` 権限と以下の権限を付与する必要があります。

- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:PutEvents`

Amazon Cognito と Amazon Pinpoint の利用可能なリージョン

次の表は、次の条件のいずれかを満たす Amazon Cognito と Amazon Pinpoint 間の AWS リージョンマッピングを示しています。

- Amazon Pinpoint プロジェクトは、米国東部 (バージニア北部) (us-east-1) リージョンでのみ使用できます。
- Amazon Pinpoint プロジェクトは、同じリージョンまたは米国東部 (バージニア北部) (us-east-1) リージョンで使用できます。

デフォルトでは、Amazon Cognito は同じ AWS リージョン の Amazon Pinpoint プロジェクトにのみ分析を送信できます。このルールの例外は、次の表のリージョンと Amazon Pinpoint が利用できないリージョンです。

Amazon Pinpoint は、次のリージョンでは利用できません。これらのリージョンの Amazon Cognito ユーザープールは分析をサポートしていません。

- 欧州 (ミラノ)
- 中東 (バーレーン)
- アジアパシフィック (大阪)
- イスラエル (テルアビブ)
- アフリカ (ケープタウン)
- アジアパシフィック (ジャカルタ)

この表は、Amazon Cognito ユーザープールを構築したリージョンと、Amazon Pinpoint の対応するリージョンとの関係を示しています。Amazon Pinpoint プロジェクトを Amazon Cognito と統合するには、使用可能なリージョンに設定する必要があります。

Amazon Cognito ユーザープールのリージョン	Amazon Pinpoint プロジェクトのリージョン
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1、ap-south-1
ap-southeast-1	us-east-1
ap-southeast-2	us-east-1、ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-east-1、eu-central-1
eu-west-1	us-east-1、eu-west-1
eu-west-2	us-east-1

Amazon Cognito ユーザープールのリージョン	Amazon Pinpoint プロジェクトのリージョン
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-east-1、us-west-2

リージョンマッピングの例

- ap-northeast-1 でユーザープールを作成する場合、Amazon Pinpoint プロジェクトは us-east-1 で作成する必要があります。
- ap-south-1 でユーザープールを作成する場合、Amazon Pinpoint プロジェクトは us-east-1 または ap-south-1 のいずれかで作成できます。

> **Note**
>
> 前の表にある以外の AWS リージョン の場合、Amazon Cognito はユーザープールと同じリージョンの Amazon Pinpoint プロジェクトのみを使用できます。ユーザープールを構築したリージョンで Amazon Pinpoint が利用できず、表にも記載されていない場合、Amazon Cognito はそのリージョンで Amazon Pinpoint 分析をサポートしません。詳細な AWS リージョン 情報については、「Amazon Pinpoint エンドポイントとクォータ」を参照してください。

Amazon Pinpoint 分析設定の指定 (AWS Management Console)

Amazon Cognito ユーザープールを設定して、分析データを Amazon Pinpoint に送信できます。Amazon Cognito は、ローカルユーザーの分析データのみを Amazon Pinpoint に送信します。ユーザープールを Amazon Pinpoint プロジェクトに関連付けるように設定したら、API リクエストに `AnalyticsMetadata` を含める必要があります。詳細については、「アプリを Amazon Pinpoint と統合する」を参照してください。

分析の設定を指定する

1. Amazon Cognito コンソールに移動します。AWS 認証情報を求められる場合があります。
2. [User Pools] (ユーザープール) を選択して、リストから既存のユーザープールを選択します。

3. [App integration] (アプリケーションの統合) タブを選択します。
4. [App clients and analytics] (アプリケーションクライアントと分析) で、リストから既存の [App client name] (アプリケーションクライアント名) を選択します。
5. [Pinpoint analytics] (Pinpoint 分析) で [Enable] (有効化) を選択します。
6. [Pinpoint Region] (Pinpoint リージョン) を選択します。
7. [Amazon Pinpoint project] (Amazon Pinpoint プロジェクト) を選択するか、[Create Amazon Pinpoint project] (Amazon Pinpoint プロジェクトを作成する) を選択します。

> **Note**
>
> Amazon Pinpoint プロジェクト ID は、Amazon Pinpoint プロジェクトに固有の 32 文字の文字列です。これは Amazon Pinpoint コンソールにリストされています。
> 複数の Amazon Cognito アプリケーションを単一の Amazon Pinpoint プロジェクトにマッピングできますが、各 Amazon Cognito アプリは 1 つの Amazon Pinpoint プロジェクトにしかマップできません。
> Amazon Pinpoint では、各プロジェクトを単一のアプリにする必要があります。例えば、ゲームデベロッパーが 2 つのゲームを持っている場合、両方のゲームが同じ Amazon Cognito ユーザープールを使用しているとしても、各ゲームを個別の Amazon Pinpoint プロジェクトにする必要があります。Amazon Pinpoint プロジェクトの詳細については、「Amazon Pinpoint でプロジェクトを作成する」を参照してください。

8. Amazon Cognito が E メールアドレスと電話番号を Amazon Pinpoint に送信し、ユーザー用の追加のエンドポイントを作成する場合は、[User data sharing] (ユーザーデータの共有) で [Share user data with Amazon Pinpoint] (Amazon Pinpoint でユーザーデータを共有する) を選択します。ユーザーが E メールアドレスと電話番号を確認した後、Amazon Cognito はそれらがユーザーアカウントで利用可能な場合にのみ Amazon Pinpoint と共有します。

> **Note**
>
> エンドポイントは、Amazon Pinpoint でプッシュ通知を送信することができるユーザーデバイスを一意に識別します。エンドポイントの詳細については、Amazon Pinpoint デベロッパーガイドの「エンドポイントの追加」を参照してください。

9. [Save changes] (変更の保存) をクリックします。

Amazon Pinpoint 分析設定の指定 (AWS CLI および AWS API)

以下のコマンドを使用して、ユーザープールの Amazon Pinpoint 分析設定を指定します。

アプリケーション作成時にユーザープールの既存のクライアントアプリケーションの分析設定を指定する

- AWS CLI: `aws cognito-idp create-user-pool-client`
- AWS API:CreateUserPoolClient

ユーザープールの既存のクライアントアプリケーションの分析設定を更新する

- AWS CLI: `aws cognito-idp update-user-pool-client`
- AWS API:UpdateUserPoolClient

> **Note**
>
> Amazon Cognito は、`ApplicationArn` の使用時にリージョン内の統合をサポートします。

アプリを Amazon Pinpoint と統合する

ユーザープール API の Amazon Cognito ローカルユーザーの分析メタデータを Amazon Pinpoint に公開できます。

ローカルユーザー

: サードパーティー ID プロバイダー (IdP) を通じてサインインするのではなく、アカウントにサインアップしたユーザー、またはユーザープールで作成されたユーザー。

ユーザープール API

: カスタムユーザーインターフェイス (UI) を備えたアプリを使用して、AWS SDK と統合できるオペレーション。ホストされた UI からサインインするフェデレーティッドユーザーまたはローカルユーザーの分析メタデータを渡すことはできません。ユーザープール API オペレーションのリストについては、「Amazon Cognito API リファレンス」を参照してください。

キャンペーンに公開するようにユーザープールを設定すると、Amazon Cognito は次の API オペレーションのためのメタデータを Amazon Pinpoint に渡します。

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- InitiateAuth
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp

ユーザーのセッションに関するメタデータを Amazon Pinpoint キャンペーンに渡すには、API リクエストの AnalyticsMetadata パラメータに AnalyticsEndpointId 値を含めます。JavaScript の例については、「AWS ナレッジセンター」の「Amazon Pinpoint ダッシュボードに Amazon Cognito ユーザープールの分析が表示されないのはなぜですか?」を参照してください。

ユーザープール分析の設定

Amazon Pinpoint 分析を使用すると、Amazon Cognito ユーザープールへのサインアップ、サインイン、失敗した認証、デイリーアクティブユーザー数 (DAU)、および月間アクティブユーザー数 (MAU) を追跡できます。AWS Mobile SDK for Android または AWS Mobile SDK for iOS を使用して、アプリケーションに固有のユーザー属性を設定することもできます。これらは、Amazon Pinpoint でユーザーを分割し、的を絞ったプッシュ通知をユーザーに送信するために使用できます。

[アプリの統合] タブの [アプリクライアントと分析] で、既存のアプリクライアントに移動するか、新しいアプリクライアントを作成できます。アプリクライアントの設定では、アプリで使用する Amazon Pinpoint プロジェクトを指定できます。詳細については、「Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用」を参照してください。

> **Note**
>
> Amazon Pinpoint は、北米、ヨーロッパ、アジア、およびオセアニアにある複数の AWS リージョンで利用できます。Amazon Pinpoint リージョンには、Amazon Pinpoint API が含まれています。Amazon Pinpoint リージョンが Amazon Cognito でサポートされている場合、Amazon Cognito は同じ Amazon Pinpoint リージョン内の Amazon Pinpoint プロジェクトにイベントを送信します。リージョンが Amazon Pinpoint でサポートされていない場合、Amazon Cognito は us-east-1 でのイベントの送信のみをサポートします。Amazon

Pinpoint の詳細なリージョン情報については、「Amazon Pinpoint エンドポイントとクォータ」および「Amazon Cognito ユーザープールを使用して Amazon Pinpoint 分析を使用する」を参照してください。

分析とキャンペーンを追加する

1. [分析とキャンペーンの追加] を選択します。
2. リストから [Cognito アプリクライアント] を選択します。
3. Amazon Cognito アプリを [Amazon Pinpoint project] (Amazon Pinpoint プロジェクト) にマッピングするには、リストから Amazon Pinpoint プロジェクトを選択します。

   > **Note**
   >
   > Amazon Pinpoint プロジェクト ID は、Amazon Pinpoint プロジェクトに固有の 32 文字の文字列です。これは Amazon Pinpoint コンソールにリストされています。
   > 複数の Amazon Cognito アプリケーションを単一の Amazon Pinpoint プロジェクトにマッピングできますが、各 Amazon Cognito アプリは 1 つの Amazon Pinpoint プロジェクトにしかマップできません。
   > Amazon Pinpoint では、各プロジェクトを単一のアプリにする必要があります。例えば、ゲームデベロッパーが 2 つのゲームを持っている場合、両方のゲームが同じ Amazon Cognito ユーザープールを使用しているとしても、各ゲームを個別の Amazon Pinpoint プロジェクトにする必要があります。

4. ユーザーのために追加のエンドポイントを作成するために Amazon Cognito が Amazon Pinpoint に E メールアドレスと電話番号を送信するようにするには、[Share user attribute data with Amazon Pinpoint] (Amazon Pinpoint とユーザー属性データを共有する) を選択します。

   > **Note**
   >
   > エンドポイントは、Amazon Pinpoint でプッシュ通知を送信することができるユーザーデバイスを一意に識別します。エンドポイントの詳細については、Amazon Pinpoint デベロッパーガイドの「Amazon Pinpoint へのエンドポイントの追加」を参照してください。

5. すでに作成した IAM ロールを入力するか、[Create new role] を選択して IAM コンソールで新しいロールを作成します。

6. [Save changes] (変更の保存) をクリックします。
7. 追加のアプリマッピングを指定するには、[Add app mapping (アプリマッピングの追加)] を選択します。
8. [Save changes] (変更の保存) をクリックします。

ユーザープール内のユーザーを管理する

ユーザープールを作成した後で、ユーザーアカウントの作成、確認、および管理を行うことができます。Amazon Cognito グループでは、IAM ロールをグループにマップすることによって、ユーザーと、ユーザーのリソースへのアクセスを管理できます。

ユーザーは、ユーザー移行の Lambda トリガーを使用してユーザープールにインポートできます。このアプローチでは、ユーザープールに初めてサインインするときに、既存のユーザーディレクトリからユーザープールにユーザーをシームレスに移行できます。

トピック

ユーザー作成ポリシーの設定

ユーザープールでは、ユーザーにサインアップを許可するか、ユーザーを管理者として作成できます。また、サインアップ後の検証と確認のプロセスを、どの程度ユーザーに委ねるかを制御できます。例えば、サインアップを確認して、外部の検証プロセスに基づいてサインアップを承認できます。この設定 (管理者作成ユーザーポリシー) では、ユーザーが自分のユーザーアカウントを確認できなくなるまでの時間も設定します。

Amazon Cognito は、ソフトウェアのカスタマー ID およびアクセス管理 (CIAM) プラットフォームとして、一般顧客のニーズに応えることができます。サインアップを受け入れるユーザープールにアプリケーションクライアントがある場合は、ホストされた UI の有無にかかわらず、一般に検出可能なアプリケーションクライアント ID を使用してインターネットからサインアップをリクエストしたすべてのユーザーに対してユーザープロファイルを作成します。サインアップしたユーザープロファイルは、アクセストークンと ID トークンを受け取り、アプリケーション用に承認されたリソースにアクセスできます。ユーザープールでサインアップを有効にする前に、オプションを確認し、設定がセキュリティ基準に準拠していることを確認してください。[自己登録を有効化] と `AllowAdminCreateUserOnly` を、以下の手順で説明するように注意して設定します。

AWS Management Console

ユーザープールの [サインアップエクスペリエンス] タブと、ユーザープール作成ウィザードの [サインアップエクスペリエンスを設定] ステップには、ユーザープール内のユーザーのサインアップと管理作成に関する設定の一部が含まれています。

サインアップエクスペリエンスを設定するには

1. [Cognito による検証と確認] で、[Cognito が検証と確認のためのメッセージを自動的に送信することを許可] するかどうかを選択します。この設定を有効にすると、Amazon Cognito はユーザープールに提示する必要があるコードを含む E メールまたは SMS メッセージを新規ユーザーに送信します。これにより、E メールアドレスまたは電話番号の所有権が確認され、同等の属性が検証済みとして設定され、ユーザーアカウントのサインインが確認されます。選択した [検証する属性] により、検証メッセージの配信方法と送信先が決まります。
2. [属性変更の確認] は、ユーザーの作成時には重要ではありませんが、属性の検証に関係します。[サインイン属性] を変更したが、まだ検証していないユーザーに対して、新しい属性値または元の属性値を使用して引き続きサインインすることを許可できます。詳細については、「ユーザーが E メールまたは電話番号を変更するときに確認する」を参照してください。
3. [必須の属性] には、ユーザーのサインアップやユーザーの作成前に値を指定する必要がある属性が表示されます。必須の属性は、ユーザープールの作成ウィザードでのみ設定できます。
4. カスタム属性は、ユーザーの作成とサインアップのプロセスにとって重要です。イミュータブルなカスタム属性の値を設定できるのは、ユーザーの初回作成時のみであるためです。カスタム属性の詳細については、「カスタム属性」を参照してください。
5. 認証されていない `SignUp` API を使用して新しいアカウントを生成することをユーザーに許可する場合は、[セルフサービスのサインアップ] で、[自己登録を有効化] を設定します。自

己登録を無効にした場合は、Amazon Cognito コンソールまたは AdminCreateUser API リクエストで新しいユーザーを管理者としてのみ作成できます。自己登録が非アクティブなユーザープールでは、SignUp API リクエストが `NotAuthorizedException` を返し、ホストされた UI に [サインアップ] リンクが表示されません。

管理者としてのユーザーを作成する予定のユーザープールでは、[サインインエクスペリエンス] タブの [管理者が設定した一時パスワードの有効期限] で一時パスワードの有効期間を設定できます。

管理者としてのユーザーを作成する場合の別の重要な要素は、招待メッセージです。新しいユーザーを作成すると、Amazon Cognito はアプリへのリンクを含むメッセージをユーザーに送信し、初回のサインインができるようにします。このメッセージテンプレートを [メッセージング] の [メッセージテンプレート] でカスタマイズします。

[機密アプリクライアント] (通常はウェブアプリケーション) には、アプリクライアントシークレットがないとサインアップできないようにするクライアントシークレットを設定できます。セキュリティ上のベストプラクティスとして、パブリックアプリクライアント (通常はモバイルアプリ) にはアプリクライアントシークレットを配布しないでください。クライアントシークレットを使用したアプリケーションクライアントは、Amazon Cognito コンソールの [アプリケーションの統合] タブで作成できます。

Amazon Cognito user pools API

ユーザープールのユーザーを作成するためのパラメータは、CreateUserPool または UpdateUserPool API リクエストを使用してプログラムで設定できます。

AdminCreateUserConfig 要素は、ユーザープールの以下のプロパティの値を設定します。

1. セルフサービスのサインアップの有効化
2. 管理者が新規作成したユーザーに送信する招待メッセージ

次の例を API リクエスト本文全体に追加すると、セルフサービスのサインアップが無効なユーザープールと、基本的な招待メールが設定されます。

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
```

```
        "SMSMessage": "Your username is {username} and temporary password is
{####}."
     }
  }
```

CreateUserPool または UpdateUserPool API リクエストの以下の追加パラメータは、新規ユーザーの作成を制御します。

AutoVerifiedAttributes

新しいユーザーを登録するときに自動的にメッセージを送信する先の属性、メールアドレス、または電話番号。

ポリシー

ユーザープールのパスワードポリシー。

Schema

ユーザープールのカスタム属性。イミュータブルなカスタム属性の値を設定できるのはユーザーの初回作成時のみであるため、カスタム属性はユーザーの作成とサインアップのプロセスにとって重要です。

このパラメータでは、ユーザープールの必須の属性も設定します。次のテキストを API リクエスト本文全体の Schema 要素に挿入すると、email 属性が必要に応じて設定されます。

```
{
            "Name": "email",
            "Required": true
}
```

ユーザーアカウントのサインアップと確認

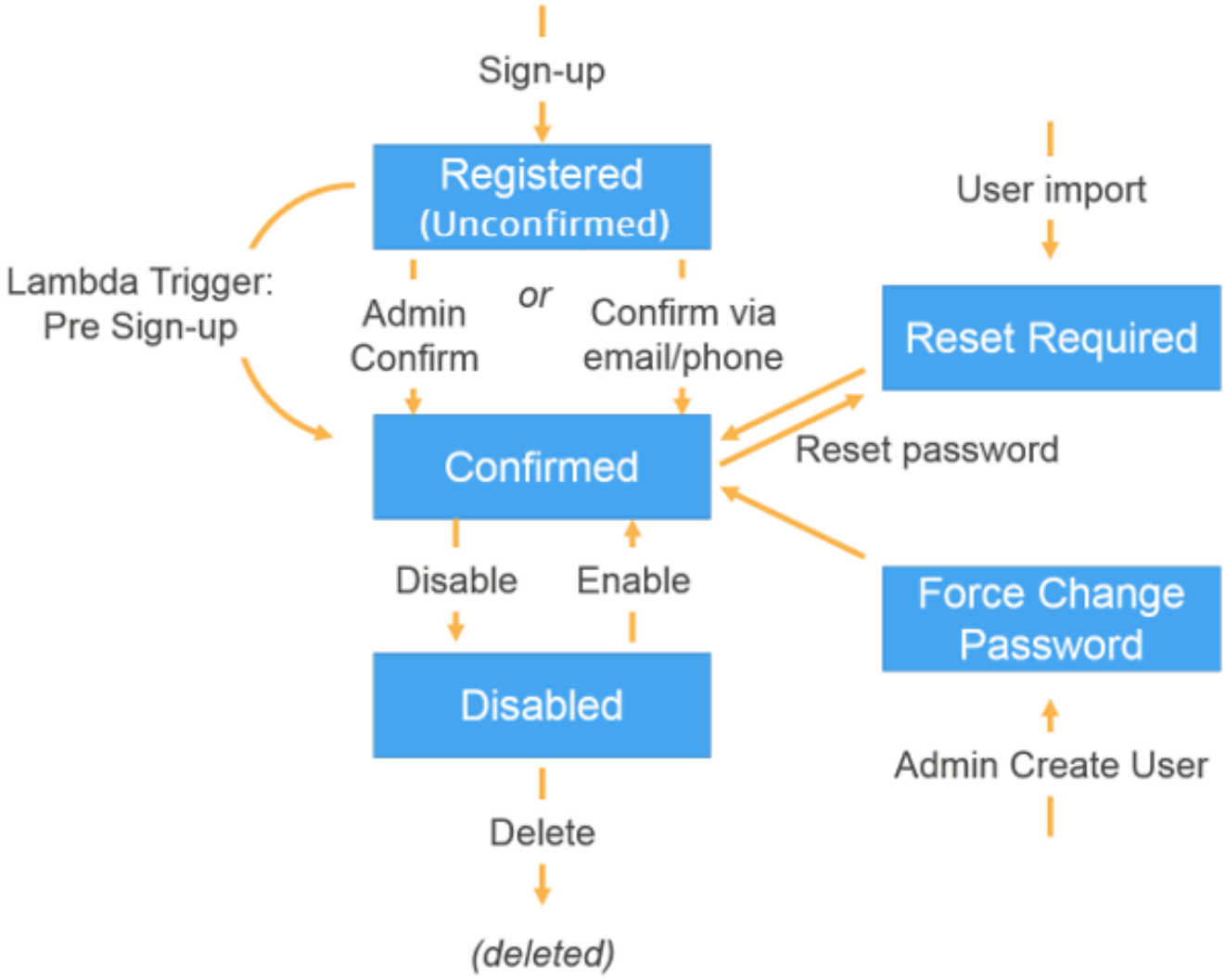
ユーザーアカウントは次の方法の 1 つでユーザープールに追加されます。

- ユーザーは、ユーザープールのクライアントアプリにサインアップします。これはモバイルまたはウェブアプリです。
- ユーザープールにユーザーのアカウントをインポートできます。詳細については、「CSV ファイルからユーザープールへのユーザーのインポート」を参照してください。
- ユーザープールにユーザーのアカウントを作成し、サインインへユーザーを招待することができます。詳細については、「管理者としてのユーザーアカウントの作成」を参照してください。

サインアップするユーザーは、サインインする前に確認が求められます。インポートされ作成されたユーザーはすでに確認されていますが、初めてサインインするときはパスワードを作成する必要があります。以下のセクションで、確認プロセス、E メール、電話確認について説明します。

ユーザーアカウント確認の概要

次の図は確認プロセスを示しています。



ユーザーアカウントは、以下のいずれかの状態になります。

登録済み (未確認)

ユーザーは正常にサインアップできますが、ユーザーアカウントを確認するまではサインインできません。ユーザーは有効ですが、この状態で確認されていません。

サインアップする新しいユーザーはこの状態から開始します。

確認済み

ユーザーアカウントが確認され、ユーザーはサインインできます。ユーザーがユーザーアカウントを確認するためにコードを入力したり、E メールのリンクをたどったりすると、その E メールや電話番号が自動的に認証されます。コードまたはリンクは 24 時間有効です。

ユーザーアカウントが管理者またはサインアップ前の Lambda トリガーによって確認された場合は、検証済みの E メールまたは電話番号がアカウントに関連付けられていないことがあります。

パスワードのリセットが必要

ユーザーアカウントは確認されましたが、ユーザーはサインインする前にコードをリクエストし、自身のパスワードをリセットする必要があります。

管理者またはデベロッパーがインポートしたユーザーアカウントは、この状態から開始します。

パスワードの強制変更

ユーザーアカウントが確認され、ユーザーは一時パスワードを使用してサインインできますが、初回のサインイン時に他の操作を行う前にパスワードを新しい値に変更する必要があります。

管理者またはデベロッパーが作成したユーザーアカウントは、この状態から開始します。

Disabled

ユーザーアカウントを削除する前に、そのユーザーのサインインアクセスを無効にする必要があります。

サインアップ時に連絡先情報を検証する

新しいユーザーがアプリにサインアップする際、1 つ以上の連絡先の入力を求める必要があります。たとえば、ユーザーの連絡先情報は、以下のように使用されます。

- ユーザーが自分のパスワードをリセットする際、仮パスワードを送信する。
- ユーザーの個人情報または財務情報が更新された際に通知する。
- プロモーションメッセージ (特価販売や割引など) を送信する。
- アカウントの概要または請求日のお知らせを送信する。

このようなユースケースでは、検証済みの送信先にメッセージを送信することが重要です。そうしないと、誤入力された無効な E メールアドレスまたは電話番号にメッセージが送信される可能性があります。さらに悪いケースでは、ユーザーを騙る悪人に機密情報が送信される可能性があります。

適切なユーザーにのみメッセージが送信されることを確実にするには、サインアップ時にユーザーが以下の情報を入力することを必須とするように Amazon Cognito ユーザープールを設定します。

a. E メールアドレスおよび電話番号。

b. Amazon Cognito がその E メールアドレスまたは電話番号に送信する検証コード。24 時間が経過し、ユーザーのコードまたはリンクが無効になった場合は、 ResendConfirmationCode API オペレーションを呼び出して新しいコードまたはリンクを生成して送信します。

検証コードを入力することで、コードを受信したメールボックスまたは電話へのアクセス権があるユーザーであることが証明されます。ユーザーがコードを入力すると、Amazon Cognito が以下を行うことによってユーザープールのユーザーに関する情報を更新します。

- ユーザーのステータスを `CONFIRMED` に設定する。
- ユーザーの属性を更新し、E メールアドレスまたは電話番号が検証されていることを示す。

この情報は、Amazon Cognito コンソールを使用して表示できます。または、 `AdminGetUser` API オペレーション、 で `admin-get-user` コマンド AWS CLI、または AWS SDKs のいずれかの対応するアクションを使用できます。

ユーザーに検証済みの連絡方法がある場合は、ユーザーがパスワードリセットをリクエストする時に、Amazon Cognito が自動的にメッセージをユーザーに送信します。

E メールまたは電話による検証を求めるようにユーザープールを設定するには

ユーザーの E メールアドレスと電話番号を確認することで、ユーザーに連絡できることを確認します。の次の手順を実行して AWS Management Console 、ユーザーが E メールアドレスまたは電話番号を確認するようにユーザープールを設定します。

> **Note**
>
> アカウントにまだユーザープールがない場合は、「ユーザープールの開始方法」を参照してください。

ユーザープールを設定するには

1. Amazon Cognitoコンソール に移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ナビゲーションペインで、[User Pools] (ユーザープール) を選択します。リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. Sign-up experience] (サインアップエクスペリエンス) タブを選択して、[Attribute verification and user account confirmation] (属性の検証とユーザーアカウントの確認) を検索します。[編集] を選択します。
4. [Cognito による検証と確認] で、[Cognito が検証と確認のためのメッセージを自動的に送信することを許可] するかどうかを選択します。この設定を有効にすると、Amazon Cognito は、ユーザーのサインアップ時またはユーザープロファイルの作成時に選択したユーザー連絡先属性にメッセージを送信します。サインイン用の属性を検証し、ユーザープロファイルを確認するために、Amazon Cognito はコードまたはリンクをメッセージでユーザーに送信します。次に、ユーザーは UI にコードを入力し、アプリが `ConfirmSignUp` または `AdminConfirmSignUp` API リクエストで確認できるようにする必要があります。

> **Note**
>
> [Cognito-assisted verification and confirmation] (Cognito アシストの検証と確認) を無効にして、API アクションまたは Lambda トリガーを使用して属性を検証し、ユーザーを確認することもできます。
> このオプションを選択すると、Amazon Cognito はユーザーのサインアップ時に検証コードを送信しません。このオプションは、Amazon Cognito からの検証コードを使用せずに、少なくとも 1 つの連絡方法を検証するカスタムの認証フローを使用している場合に選択します。例えば、特定のドメインに属する E メールアドレスを自動的に確認するサインアップ前 Lambda トリガーを使用することができます。
> ユーザーの連絡先情報を検証しない場合は、アプリケーションを使用できない場合があります。以下を行うには、ユーザーの連絡情報の検証が必要となる点にご注意ください。
>
> - [Reset their passwords] (パスワードのリセット) — ユーザーが `ForgotPassword` API アクションを呼び出すオプションをアプリケーションで選択すると、Amazon Cognito はユーザーの E メールアドレスまたは電話番号に一時パスワードを送信します。Amazon Cognito がこのパスワードを送信するのは、ユーザーが検証済みの連絡方法を少なくとも 1 つ持っている場合のみです。

- [Sign in by using an email address or phone number as an alias] (E メールアドレスまたは電話番号をエイリアスとして使用してサインインする) — これらのエイリアスを許可するようにユーザープールを設定した場合、ユーザーは、検証済みである場合にのみ、そのエイリアスを使用してサインインできます。詳細については、「ログイン属性のカスタマイズ」を参照してください。

5. [Attributes to verify] (検証する属性) を以下から選択します。

SMS メッセージを送信し、電話番号を確認する

ユーザーがサインアップすると Amazon Cognito が SMS メッセージで検証コードを送信します。通常 SMS メッセージでユーザーと通信する場合は、このオプションを選択してください。例えば、配信通知、予約確認、または警告を送信する場合は、検証済みの電話番号を使用します。アカウントが確認されると、ユーザーの電話番号が検証済み属性になります。ユーザーの E メールアドレスを確認して通信するには、追加のアクションを実行する必要があります。

E メールメッセージを送信し、E メールアドレスを確認する

ユーザーがサインアップすると Amazon Cognito が E メール経由で検証コードを送信します。通常 E メールでユーザーと通信する場合は、このオプションを選択してください。たとえば、請求明細、注文書、特価販売を使用する場合は、検証済みの E メールアドレスを使用する必要があります。アカウントが確認されると、ユーザーのメールアドレスが検証済み属性になります。ユーザーの電話番号を確認して通信するには、追加のアクションを実行する必要があります。

電話番号が利用可能な場合は SMS メッセージを送信し、そうでない場合は E メールメッセージを送信する

すべてのユーザーに同じ検証済みの連絡方法を要求する必要がない場合は、このオプションを選択してください。この場合、アプリのサインアップページで、希望する連絡方法のみを検証するようにユーザーに求めることができます。Amazon Cognito が検証コードを送信するときは、アプリからの `SignUp` リクエストで指定された連絡方法にコードが送信されます。ユーザーが E メールアドレスと電話番号の両方を入力し、アプリが `SignUp` リクエストに両方の連絡方法を指定する場合、Amazon Cognito は検証コードを電話番号のみに送信します。

ユーザーが E メールアドレスと電話番号の両方を検証することを必須としている場合は、このオプションを選択してください。Amazon Cognito は、ユーザーのサインアップ時に 1 つの連絡方法を検証するので、アプリがユーザーのサインイン後にもう一方の連絡方法を検証

する必要があります。詳細については、「ユーザーに E メールアドレスと電話番号の両方の確認を要求する場合」を参照してください。

6. [変更の保存] を選択します。

E メールアドレスまたは電話による検証を使用した認証フロー

ユーザープールで連絡先情報の検証をユーザーに要求する場合、ユーザーがアプリにサインアップした後のフローを容易にする必要があります。

1. ユーザーは、ユーザー名、メールアドレス、電話番号やその他の属性を入力してアプリにサインアップします。
2. Amazon Cognito サービスは、アプリからサインアップリクエストを受け取ります。サインアップに必要なすべての属性がリクエストに含まれることを確認した後で、サービスはサインアッププロセスを完了し、確認コードをユーザーの電話 (SMS メッセージ) または E メールに送信します。コードは 24 時間有効です
3. サービスは、サインアップが完了しユーザーアカウントが確認保留であることを返します。応答には、確認コードの送信先に関する情報が含まれます。この時点でユーザーアカウントは未確認状態になり、ユーザーのメールアドレスと電話番号は未確認です。
4. アプリは、ユーザーに確認コードを入力するよう求めるメッセージを表示できます。ユーザーは、コードをすぐに入力する必要はありません。ただし、確認コードを入力するまでユーザーはサインインできません。
5. ユーザーがアプリに確認コードを入力します。
6. アプリは、Amazon Cognito サービスにコードを送信する `ConfirmSignUp` を呼び出します。これは、コードを検証し、コードが正しい場合はユーザーのアカウントを確認済み状態に設定します。ユーザーアカウントが正常に確認されると、Amazon Cognito サービスが自動的に、E メールアドレスまたは電話番号を確認するために使用された属性を検証済みとしてマークします。この属性の値が変更されていない場合、ユーザーはそれを再度確認する必要はありません。
7. この時点で、ユーザーアカウントは確認済みの状態になっており、ユーザーはサインインできます。

ユーザーに E メールアドレスと電話番号の両方の確認を要求する場合

Amazon Cognito は、ユーザーのサインアップ時に 1 つの連絡方法しか検証しません。Amazon Cognito が E メールアドレスまたは電話番号のどちらを検証するかを選択する必要がある場合は、SMS メッセージ経由で検証コードを送信して電話番号を検証することを選択します。例えば、

ユーザーが E メールアドレスと電話番号のいずれかを検証できるようにユーザープールを設定する場合、およびアプリがサインアップ時にこれら両方の属性を提供する場合、Amazon Cognito は電話番号のみを検証します。ユーザーが電話番号を検証したら、Amazon Cognito がユーザーのステータスを CONFIRMED に設定し、ユーザーはアプリにサインインできるようになります。

ユーザーがアプリにサインインした後、サインアップ時に検証されなかった連絡方法を検証するオプションを提示することができます。この 2 つ目の方法を検証するには、アプリで VerifyUserAttribute API アクションを呼び出します。このアクションには AccessToken パラメータが必要で、Amazon Cognito は認証されたユーザーにアクセストークンしか提供しないことに注意してください。したがって、2 つ目の連絡方法は、ユーザーがサインインした後にのみ検証できます。

E メールアドレスと電話番号の両方を検証するようにユーザーに求める場合は、以下のように行います。

1. E メールアドレスまたは電話番号を検証することをユーザーに許可するようにユーザープールを設定します。
2. アプリのサインアップフローで、E メールアドレスおよび電話番号の両方を入力するようユーザーに求めます。SignUp API アクションを呼び出し、UserAttributes パラメータの E メールアドレスおよび電話番号を指定します。この時点で、Amazon Cognito がユーザーの携帯電話に検証コードを送信します。
3. アプリのインターフェイスで、ユーザーが検証コードを入力する確認ページを提示します。ユーザーを確認するには、ConfirmSignUp API アクションを呼び出します。この時点で、ユーザーのステータスは CONFIRMED となり、ユーザーの電話番号は検証されますが、E メールアドレスはまだ検証されていません。
4. サインインページを提示し、ユーザーを認証するには、InitiateAuth API アクションを呼び出します。ユーザーが認証されると、Amazon Cognito がアプリにアクセストークンを返します。
5. GetUserAttributeVerificationCode API アクションを呼び出します。リクエストで次のパラメータを指定します。

   - AccessToken – ユーザーのサインイン時に Amazon Cognito が返したアクセストークン。
   - AttributeName – "email" を属性値として指定します。

   Amazon Cognito がユーザーの E メールアドレスに検証コードを送信します。

6. ユーザーが検証コードを入力する確認ページを提示します。ユーザーがコードを入力したら、`VerifyUserAttribute` API アクションを呼び出します。リクエストで次のパラメータを指定します。

   - `AccessToken` – ユーザーのサインイン時に Amazon Cognito が返したアクセストークン。
   - `AttributeName` – `"email"` を属性値として指定します。
   - `Code` – ユーザーが提供した検証コード。

   この時点で、E メールアドレスが検証されます。

ユーザーにアプリケーションへのサインアップを許可するがユーザープール管理者として確認する

ユーザープールで確認メッセージを自動的に送信しないにも関わらず、誰でもアカウントにサインアップできるようにしたい場合があります。このモデルでは、たとえば、新しい登録リクエストを人間が確認したり、サインアップの一括検証と処理を行ったりする余地があります。新しいユーザーアカウントは、Amazon Cognito コンソールまたは IAM 認証 API オペレーション で確認できますAdminConfirmSignUp。ユーザープールが確認メッセージを送信するかどうかにかかわらず、ユーザーアカウントを管理者として確認できます。

この方法では、ユーザーのセルフサービスサインアップを確認することしかできません。管理者として作成したユーザーを確認するには、 を `Permanent`に設定して AdminSetUserPassword API リクエストを作成します`True`。

1. ユーザーは、ユーザー名、メールアドレス、電話番号やその他の属性を入力してアプリにサインアップします。
2. Amazon Cognito サービスは、アプリからサインアップリクエストを受け取ります。サインアップに必要なすべての属性がリクエストに含まれることを確認した後で、サービスはサインアッププロセスを完了し、アプリにサインアップが完了し確認が保留中であることをアプリに返します。この時点でユーザーのアカウントは未確認状態です。アカウントが確認されるまで、ユーザーはサインインすることはできません。
3. ユーザーのアカウントを確認します。アカウントを確認するには、 にサインイン AWS Management Console するか、 AWS 認証情報を使用して API リクエストに署名する必要があります。

   a. Amazon Cognito コンソールでユーザーを確認するには、[ユーザー] タブに移動して確認するユーザーを選択し、[アクション] メニューから [確認] を選択します。

b. AWS API または CLI でユーザーを確認するには、 AdminConfirmSignUp API リクエスト、または admin-confirm-sign-upで を作成します AWS CLI。

4. この時点で、ユーザーアカウントは確認済みの状態になっており、ユーザーはサインインできます。

シークレットハッシュ 値の計算

ベストプラクティスとして、機密アプリケーションクライアントにクライアントシークレットを割り当てます。アプリケーションクライアントにクライアントシークレットを割り当てる場合、Amazon Cognito ユーザープール API リクエストには、リクエスト本文にクライアントシークレットを含むハッシュを含める必要があります。以下のリストにある API オペレーションのクライアントシークレットに関する知識を検証するには、クライアントシークレットをアプリケーションのクライアント ID とユーザーのユーザー名と連結し、その文字列を base64 でエンコードします。

シークレットハッシュを持つクライアントにアプリがユーザーをサインインすると、ユーザープールのサインイン属性の値をシークレットハッシュのユーザー名要素として使用できます。アプリが `REFRESH_TOKEN_AUTH` による認証オペレーションで新しいトークンをリクエストする場合、ユーザー名要素の値はサインイン属性によって異なります。ユーザープールにサインイン属性として `username` がない場合は、ユーザーのアクセストークンまたは ID トークンの `sub` クレームからのシークレットハッシュユーザー名の値を設定します。`username` がサインイン属性である場合は、`username` クレームからのシークレットハッシュユーザー名の値を設定します。

以下の Amazon Cognito ユーザープール API は、`SecretHash` パラメータにクライアントシークレットのハッシュ値を受け入れます。

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- ResendConfirmationCode
- SignUp

また、以下の API は、認証パラメータまたはチャレンジレスポンス内の `SECRET_HASH` パラメータにクライアントシークレットのハッシュ値を受け入れます。

API オペレーション	SECRET_HASH の親パラメータ
InitiateAuth	AuthParameters
AdminInitiateAuth	AuthParameters
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

シークレットハッシュ値は、Base 64 でエンコードされたキー付きハッシュメッセージ認証コード (HMAC) であり、ユーザープールクライアントおよびユーザー名、さらにメッセージ内のクライアント ID を使用して計算されたものです。次の擬似コードは、この値の計算方法を示します。この擬似コードで + は連結を表し、HMAC_SHA256 は HmacSHA256 を使用して HMAC 値を生成する関数を、Base64 は Base-64 でエンコードされたバージョンのハッシュ出力を生成する関数を示します。

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

SecretHash パラメータの計算方法と使用方法の詳細な概要については、 AWS ナレッジセンターのAmazon Cognito ユーザープール API からクライアント <client-id> のエラーのシークレットハッシュを確認できない」のトラブルシューティング方法を教えてください。

サーバー側の Java アプリケーションコードで、次のコード例を使用できます。

Shell

```
echo -n "[username][app client ID]" | openssl dgst -sha256 -hmac [app client secret]
 -binary | openssl enc -base64
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
 userPoolClientSecret, String userName) {
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";
```

```
    SecretKeySpec signingKey = new SecretKeySpec(
            userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
            HMAC_SHA256_ALGORITHM);
    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac =
 mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
}
```

Python

```
import sys
import hmac, hashlib, base64
username = sys.argv[1]
app_client_id = sys.argv[2]
key = sys.argv[3]
message = bytes(sys.argv[1]+sys.argv[2],'utf-8')
key = bytes(sys.argv[3],'utf-8')
secret_hash = base64.b64encode(hmac.new(key, message,
 digestmod=hashlib.sha256).digest()).decode()
print("SECRET HASH:",secret_hash)
```

E メールまたは電話番号の確認なしでユーザーアカウントを確認する

検証コードを要求したり、E メールまたは電話番号を検証したりすることなくサインアップ時にユーザーアカウントを自動的に確認するには、サインアップ前の Lambda トリガーを使用できます。この方法で確認されたユーザーは、コードを受け取らないですぐにサインインできます。

このトリガーによって検証されたユーザーのメールまたは電話番号をマーキングすることもできます。

> **Note**
>
> これはユーザーが始めるにあたって便利な方法ですが、E メールまたは電話番号を少なくとも 1 つ自動検証することをお勧めします。そうしないと、ユーザーがパスワードを忘れた際に復旧できないままになる場合があります。

ユーザーがサインアップ時に検証コードを受信して入力することを必須とせず、サインアップ前の Lambda トリガーで E メールと電話番号を自動検証しない場合は、そのユーザーアカウントに検証済みの E メールアドレスまたは電話番号が存在しないリスクが生じます。ユーザーはメールアドレスまたは電話番号を後で確認できます。ただし、パスワードを忘れて確認済みの E メールアドレスまたは電話番号がない場合、パスワードを忘れた場合のフローで検証コードをユーザーに送信するために確認済みの E メールまたは電話番号が必要となるため、ユーザーはアカウントからロックアウトされます。

ユーザーが E メールまたは電話番号を変更するときに確認する

ユーザーがアプリで自分の E メールアドレスまたは電話番号を更新すると、その属性を自動的に検証するようにユーザープールを設定した場合、Amazon Cognito は検証コード付きのメッセージをすぐにユーザーに送信します。その後、ユーザーは検証メッセージからアプリにコードを提供する必要があります。次に、アプリケーションは VerifyUserAttribute API リクエストでコードを送信して、新しい属性値の検証を完了します。

ユーザープールで更新された E メールアドレスまたは電話番号の検証がユーザーに要求されない場合、Amazon Cognito はただちに更新された `email` または phone_number 属性の値を変更し、その属性を未検証としてマークします。ユーザーは、未確認の E メールまたは電話番号でサインインできません。その属性をサインインエイリアスとして使用するには、更新された値の検証を完了する必要があります。

ユーザープールで更新された E メールアドレスまたは電話番号の検証がユーザーに要求される場合、Amazon Cognito はユーザーが新しい属性値を検証するまで、属性を検証されたままにし、元の値に設定します。属性がサインイン用のエイリアスである場合、検証によって属性が新しい値に変更されるまで、ユーザーは元の属性値を使用してサインインできます。更新された属性の検証をユーザーに要求するようにユーザープールを設定する方法の詳細については、「E メールまたは電話による検証の設定」を参照してください。

検証メッセージは、カスタムメッセージの Lambda トリガーを使用してカスタマイズできます。詳細については、「カスタムメッセージの Lambda トリガー」を参照してください。ユーザーの E メールアドレスや電話番号が未確認の場合、アプリはユーザーに属性を確認する必要があることを通

知し、ユーザーが新しい E メールアドレスや電話番号を確認するためのボタンやリンクを提供する必要があります。

管理者またはデベロッパーが作成するユーザーアカウントの確認および検証プロセス

管理者またはデベロッパーが作成したユーザーアカウントは常に確認済み状態にあるため、ユーザーは確認コードを入力する必要はありません。Amazon Cognito サービスがこれらのユーザーに送信する招待メッセージには、ユーザー名と一時パスワードが含まれています。ユーザーは、サインインする前にパスワードを変更する必要があります。詳細について、は、「E メールメッセージと SMS メッセージのカスタマイズ」の「管理者としてのユーザーアカウントの作成」と「Lambda トリガーを使用したユーザープールワークフローのカスタマイズ」のカスタムメッセージのトリガーを参照してください。

インポート済みユーザーアカウントの確認および検証プロセス

、CLI AWS Management Console、または API のユーザーインポート機能を使用して作成されたユーザーアカウント (「」を参照CSV ファイルからユーザープールへのユーザーのインポート) は既に確認済み状態になっているため、ユーザーは確認コードを入力する必要はありません。招待メッセージは送信されません。ただし、インポートされたユーザーアカウントでは、サインインする前にまず `ForgotPassword` API を呼び出してコードをリクエストし、次に、提供されたコードを使用して `ConfirmForgotPassword` API を呼び出すことによってパスワードを作成する必要があります。詳細については、「インポートされたユーザーにパスワードをリセットするように要求」を参照してください。

ユーザーアカウントがインポートされたときに、ユーザーの E メールまたは電話番号は確認済みとしてマークする必要があります。したがって、ユーザーがサインインする際に認証は必要ありません。

アプリケーションのテスト中に E メールを送信する

ユーザープールのクライアントアプリでアカウントを作成および管理する場合は、Amazon Cognito がユーザーに E メールメッセージを送信します。E メール検証を必須とするようにユーザープールを設定した場合は、以下のタイミングで Amazon Cognito が E メールを送信します。

- ユーザーのサインアップ時。
- ユーザーが E メールアドレスを更新する際。
- ユーザーが `ForgotPassword` API アクションを呼び出すアクションを実行する際。
- 管理者としてユーザーアカウントを作成する際。

E メールを開始するアクションに応じて、検証コードまたは仮パスワードが含まれます。ユーザーは、これらの E メールを受け取り、メッセージを理解する必要があります。そうしないと、アプリにサインインして使用できない場合があります。

E メールが正常に送信され、メッセージの内容が正しいことを確認するには、Amazon Cognito からの E メール配信を開始するアプリのアクションをテストします。たとえば、アプリでサインアップページを使用するか、`SignUp` API アクションを使用して、テスト用メールアドレスでサインアップすることで E メールを配信できます。この方法でテストする場合は、次の点に注意してください。

> [重要]
>
> Amazon Cognito からの E メールを開始するアクションをテストするために E メールアドレスを使用する場合は、偽の E メールアドレス (メールボックスが存在しないメールアドレス) は使用しないでください。ハードバウンスを発生させずに Amazon Cognito からの E メールを受信する実際の E メールアドレスを使用します。
> ハードバウンスは、Amazon Cognito が受取人のメールボックスへの E メールの配信に失敗した場合に発生します。これは、常にメールボックスが存在しない場合に発生します。
> Amazon Cognito は、ハードバウンスが永続的に発生する AWS アカウントによって送信できる E メールの数を制限します。

E メールを配信するアクションをテストする場合、次のいずれかの E メールアドレスを使用してハードバウンスを回避します。

- テスト用に所有し、使用している E メールアカウントのメールアドレス。独自の E メールアドレスを使用すると、Amazon Cognito が送信する E メールを受け取ります。この E メールでは、検証コードを使用して、アプリのサインアップエクスペリエンスをテストできます。ユーザープールの E メールメッセージをカスタマイズした場合は、正しく表示されていることを確認します。
- メールボックスシミュレーターのアドレス (success@simulator.amazonses.com)。シミュレーターのアドレスを使用する場合、Amazon Cognito は E メールを正常に送信しますが、それを表示することはできません。このオプションは、検証コードを使用する必要がなく、E メールのメッセージを確認する必要がない場合に役立ちます。
- success+user1@simulator.amazonses.com または success+user2@simulator.amazonses.com などの、任意のラベルを使用するメールボックスシミュレーターのアドレス。Amazon Cognito はこれらのアドレスに E メールを正常に送信しますが、送信された E メールを表示することはできません。このオプションは、複数のテストユーザーをユーザープールに追加してサインアッププロセスをテストし、各テストユーザーに一意の E メールアドレスがある場合に便利です。

E メールまたは電話による検証の設定

E メールまたは電話による確認の設定は、[メッセージング] タブで選択できます。多要素認証 (MFA) の詳細については、「SMS テキストメッセージ MFA」を参照してください。

Amazon Cognito は、SMS メッセージの送信に Amazon SNS を使用します。Amazon Cognito または他の から SMS メッセージを送信したことがない場合 AWS のサービス 、Amazon SNS はアカウントを SMS サンドボックスに配置することがあります。アカウントをサンドボックスから本番環境に削除する前に、検証済みの電話番号にテストメッセージを送信することをお勧めします。さらに、米国の宛先電話番号に SMS メッセージを送信する場合は、Amazon Pinpoint から発信元 ID または送信者 ID を取得する必要があります。SMS メッセージ用に Amazon Cognito ユーザープールを設定するには、「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。

Amazon Cognito は、E メールアドレスまたは電話番号を自動的に検証できます。この検証を行うために、Amazon Cognito は検証コードまたは検証リンクを送信します。E メールアドレスの場合、Amazon Cognito は、コードまたはリンクを E メールメッセージで送信できます。Amazon Cognito コンソールの [メッセージング] タブで [検証メッセージ] テンプレートを編集するときに、検証タイプとしてコードまたはリンクを選択できます。詳細については、「E メール検証メッセージのカスタマイズ」を参照してください。

電話番号の場合、Amazon Cognito は、コードを SMS テキストメッセージで送信します。

Amazon Cognito は、ユーザーを確認し、パスワードを忘れた場合の回復を支援するために、電話番号または E メールアドレスを検証する必要があります。または、サインアップ前の Lambda トリガーでユーザーを自動的に確認するか、 AdminConfirmSignUp API オペレーションを使用できます。詳細については、「ユーザーアカウントのサインアップと確認」を参照してください。

確認コードまたはリンクは 24 時間有効です。

E メールアドレスや電話番号の検証を必須とした場合、Amazon Cognito はユーザーのサインアップ時に自動的に検証コードまたはリンクを送信します。ユーザープールに カスタム SMS 送信者の Lambda トリガー または カスタム E メール送信者の Lambda トリガー を設定すると、その関数が代わりに呼び出されます。

> **メモ**
>
> - 電話番号の検証に SMS テキストメッセージングを使用すると、Amazon SNS の料金が別途請求されます。E メールメッセージを送信しても料金はかかりません。Amazon SNS の料金については、「Worldwide SMS Pricing」を参照してください。SMS メッセージを利

用可能な国の最新のリストについては、「サポートされるリージョンと国」を参照してください。

- Amazon Cognito からの E メールメッセージを生成するアプリ内のアクションをテストするときは、Amazon Cognito がハードバウンスを発生させずに送信できる実際の E メールアドレスを使用してください。詳細については、「the section called “アプリケーションのテスト中に E メールを送信する”」を参照してください。
- パスワードを忘れた場合のフローでは、ユーザーのメールまたはユーザーの電話番号が検証される必要があります。

Important

ユーザーが電話番号と E メールアドレスの両方にサインアップし、ユーザープール設定で両方の属性の確認が必要な場合は、Amazon Cognito は、検証コードを SMS メッセージ経由で電話番号に送信します。Amazon Cognito は E メールアドレスをまだ検証していないため、アプリは GetUserを呼び出して、E メールアドレスが検証を待っているかどうかを確認する必要があります。検証が必要な場合、アプリは を呼び出しGetUserAttributeVerificationCodeて E メール検証フローを開始する必要があります。次に、 を呼び出して検証コードを送信する必要がありますVerifyUserAttribute。

SMS メッセージの支出クォータは、 AWS アカウント および個々のメッセージに対して調整できます。制限は SMS メッセージを送信するコストにのみ適用されます。詳細については、Amazon SNS よくある質問の「アカウントレベルとメッセージレベルの支出クォータとは何ですか。また、どのように機能しますか。」を参照してください。

Amazon Cognito は、ユーザープール AWS リージョン を作成した または次の表のレガシー Amazon SNS 代替リージョンのいずれかで、Amazon SNS リソースを使用して SMS メッセージを送信します。 Amazon SNS 例外は、アジアパシフィック (ソウル) リージョンの Amazon Cognito ユーザープールです。これらのユーザープールは、アジアパシフィック (東京) リージョンで Amazon SNS 設定を使用します。詳細については、「Amazon SNS SMS メッセージの AWS リージョン を選択する」を参照してください。

Amazon Cognito リージョン	レガシー Amazon SNS 代替リージョン
米国東部 (オハイオ)	米国東部 (バージニア北部)

Amazon Cognito リージョン	レガシー Amazon SNS 代替リージョン
アジアパシフィック (ムンバイ)	アジアパシフィック (シンガポール)
アジアパシフィック (ソウル)	アジアパシフィック (東京)
カナダ (中部)	米国東部 (バージニア北部)
欧州 (フランクフルト)	欧州 (アイルランド)
欧州 (ロンドン)	欧州 (アイルランド)

例: Amazon Cognito のユーザープールがアジアパシフィック (ムンバイ) にあり、ap-southeast-1 で使用制限を引き上げている場合は、ap-south-1 で別途引き上げる要求をしないでください。代わりに、アジアパシフィック (シンガポール) で Amazon SNS リソースを使用できます。

E メールアドレスと電話番号の更新を検証する

E メールアドレスまたは電話番号の属性は、ユーザーが値を変更した直後にアクティブになり、検証されない場合があります。Amazon Cognito は、Amazon Cognito が属性を更新する前に、ユーザーに新しい値を検証するように要求することもできます。ユーザーが新しい値を最初に検証することを要求する場合、新しい値を検証するまでは、サインインおよびメッセージの受信に元の値を使用することができます。

ユーザーがユーザープールの E メールアドレスまたは電話番号をサインインエイリアスとして使用できる場合、更新された属性のサインイン名は、更新された属性の検証が必要かどうかによって異なります。更新された属性を検証する必要がある場合、新しい値を検証するまで、ユーザーは元の属性値を使用してサインインできます。更新された属性を検証する必要がない場合、新しい値を検証するまで、ユーザーは新しい属性値または元の属性値でサインインまたはメッセージを受信することはできません。

例えば、ユーザープールでは、E メールアドレスのエイリアスでのサインインが許可され、ユーザーが更新時に E メールアドレスを検証する必要があるとします。`sue@example.com` としてサインインしている Sue は、自分の E メールアドレスを `sue2@example.com` に変更しようとしましたが、誤って **s**`sue2@example.com` と入力してしまいました。Sue は確認用の E メールを受信していないので、`ssue2@example.com` を検証することができません。`sue@example.com` としてサインインし、アプリでフォームを再送信して、E メールアドレスを `sue2@example.com` に更新します。この E メールを受信し、アプリに確認コードを提供して、`sue2@example.com` としてサインインを開始します。

ユーザーが属性を更新し、ユーザープールが新しい属性値を検証する場合

- コードを確認する前は、元の属性値でサインインして新しい値を確認できます。
- コードを確認した後は、新規の属性値でサインインして新しい値を確認できます。
- AdminUpdateUserAttributes API リクエスト`true`で `email_verified`または `phone_number_verified`を に設定した場合、Amazon Cognito が送信したコードを確認する前にサインインできます。

ユーザーが属性を更新し、ユーザープールが新しい属性値を検証しない場合

- 元の属性値を使用してサインインしたり、元の属性値でメッセージを受信したりすることはできません。
- 新しい値を確認するコードを確認するま、新しい属性値を使用してサインインしたり、確認コード以外のメッセージを受信したりすることはできません。
- AdminUpdateUserAttributes API リクエスト`true`で `email_verified`または `phone_number_verified`を に設定した場合、Amazon Cognito が送信したコードを確認する前にサインインできます。

ユーザーが E メールアドレスまたは電話番号を更新するときに属性の検証を要求するには

1. Amazon Cognito コンソールにサインインします。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [Sign-up experience] (サインアップエクスペリエンス) タブで、[Attribute verification and user account confirmation] (属性の検証とユーザーアカウントの確認) から [Edit] (編集) を選択します。
4. [Keep original attribute value active when an update is pending] (更新が保留中の場合、元の属性値をアクティブに保つ) を選択します。
5. [Active attribute values when an update is pending] (更新が保留中の場合にアクティブな属性値) から、Amazon Cognito が値を更新する前にユーザーに検証を要求する属性を選択します。
6. [変更を保存] を選択します。

Amazon Cognito API で属性更新の検証を要求するには、 UpdateUserPoolリクエストで `AttributesRequireVerificationBeforeUpdate`パラメータを設定できます。

SMS メッセージを代理送信するために Amazon Cognito を承認します。

SMS メッセージをユーザーに代理送信するには、Amazon Cognito に許可が必要です。このアクセス許可を付与するには、 AWS Identity and Access Management (IAM) ロールを作成します。Amazon Cognito コンソールの [メッセージング] タブの [SMS] で、[編集] を選択してロールを設定します。

SMS と E メール検証メッセージおよびユーザー招待メッセージの設定

Amazon Cognito では、SMS および E メール検証メッセージとユーザー招待メッセージをカスタマイズして、アプリケーションのセキュリティとユーザーエクスペリエンスを向上させることができます。Amazon Cognito では、アプリケーションのニーズに応じて、コードベースのリンク検証またはワンクリックリンク検証を選択できます。このトピックでは、Amazon Cognito コンソールで多要素認証 (MFA) および検証通信をパーソナライズする方法について説明します。

[メッセージング] タブの [メッセージテンプレート] で、以下をカスタマイズできます。

- SMS テキストメッセージ多要素認証 (MFA) メッセージ
- SMS と E メールの検証メッセージ
- E メールの検証タイプ (コードまたはリンク)
- ユーザーの招待メッセージ
- ユーザープールを介するメールの FROM および REPLY-TO E メールアドレス

> **Note**
>
> SMS と電子メールの確認メッセージのテンプレートは、[検証] タブで電話番号と E メールの確認を要求するよう選択した場合にのみ表示されます。同様に、SMS MFA メッセージのテンプレートは、MFA 設定が [required] (必要) または [optional] (任意) の場合にのみ表示されます。

トピック

- E メールアドレスのカスタマイズ
- Amazon SES E メールを代理送信するための Amazon Cognito の承認 (カスタム REPLY-TO E メールアドレスからの送信)

メッセージテンプレート

メッセージテンプレートを使用すると、対応する値が置き換わるプレースホルダーを使用して、メッセージにフィールドを挿入することができます。

テンプレートのプレースホルダー

説明	トークン
確認コード	{####}
一時パスワード	{####}
ユーザー名	{username}

> **Note**
>
> 検証 E メールメッセージに {username} プレースホルダーを使用することはできません。{username} プレースホルダーは、 AdminCreateUserオペレーションで生成する招待 E メールメッセージで使用できます。これらの招待 E メールメッセージには 2 つのプレースホルダーを使用します。{username} というユーザー名、{####} という一時パスワードです。

アドバンスドセキュリティのテンプレートのプレースホルダーを使用して、以下を行うことができます。

- IP アドレス、市、国、サインイン時間、デバイス名など、イベントに関する具体的な詳細を記載します。Amazon Cognito のアドバンスドセキュリティ機能は、これらの詳細を分析できます。
- ワンクリックリンクが有効であるかどうかを確認します。
- イベント ID、フィードバックトークン、およびユーザー名を使用して独自のワンクリックリンクを構築します。

> **Note**
>
> ワンクリックリンクを生成し、アドバンスドセキュリティ E メールテンプレートで {one-click-link-valid} および {one-click-link-invalid} プレースホルダーを使用するには、ユーザープール用にドメインが既に設定されている必要があります。

アドバンスドセキュリティのテンプレートのプレースホルダー

説明	トークン
IP アドレス	{ip-address}
City	{city}
Country	{country}
ログイン時間	{login-time}
デバイス名	{device-name}
オンクリックリンクが有効	{one-click-link-valid}
オンクリックリンクが無効	{one-click-link-invalid}
イベント ID	{event-id}
フィードバックトークン	{feedback-token}

SMS メッセージのカスタマイズ

> **Note**
>
> 新しい Amazon Cognito コンソールエクスペリエンスでは、SMS メッセージをカスタマイズできます。

多要素認証 (MFA) 用の SMS メッセージは、[メッセージテンプレート] 見出しの下にある [メッセージング] タブでカスタマイズできます。

> **Important**
>
> カスタムメッセージには、{####}プレースホルダーを含む必要があります。このプレースホルダーは、メッセージが送信される前に認証コードへ置き換えられます。

Amazon Cognito では、SMS メッセージの最大長を 140 文字 (認証コードを含む) の UTF-8 文字に設定しています。

SMS 検証メッセージのカスタマイズ

[SMS 検証メッセージをカスタマイズしますか?] ヘッダーの下のテンプレートを編集して、電話番号認証用の SMS メッセージをカスタマイズできます。

> **Important**
>
> カスタムメッセージには、{####}プレースホルダーを含む必要があります。このプレースホルダーは、メッセージが送信される前に検証コードへ置き換えられます。

メッセージの最大長は検証コードを含めて UTF-8 で 140 文字です。

E メール検証メッセージのカスタマイズ

Amazon Cognito でユーザープール内のユーザーの E メールアドレスを確認するには、ユーザーが選択できるリンクを記載した E メールメッセージをユーザーに送信するか、ユーザーが入力できるコードを送信します。

E メールアドレス検証メッセージの E メールの件名とメッセージ内容をカスタマイズするには、ユーザープールの [メッセージング] タブで [検証メッセージ] テンプレートを編集します。[検証メッセージ] テンプレートを編集するときに、検証タイプとしてコードまたはリンクを選択できます。

検証タイプとしてコードを選択した場合、カスタムメッセージには {####} プレースホルダーを含める必要があります。メッセージを送信するときに、検証コードはこのプレースホルダーを置き換えます。

検証タイプとしてリンクを選択した場合、カスタムメッセージにはプレースホルダーを {##Verify Your Email##} の形式で含める必要があります。プレースホルダー文字間のテキスト文字列は、{##Click here##} のように変更できます。E メールを検証するというタイトルの検証リンは、このプレースホルダーを置き換えます。

E メール検証メッセージのリンクは、ユーザーを次の例のような URL に誘導します。

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

メッセージの最大長は検証コード (ある場合) を含めて UTF-8 で 20,000 文字です。このメッセージでは、HTML タグを使用してコンテンツの書式を設定できます。

ユーザー招待メッセージのカスタマイズ

Amazon Cognito が SMS または E メールメッセージを使用して新規ユーザーに送信するユーザー招待メッセージをカスタマイズするには、[メッセージング] タブの []招待メッセージ] テンプレートを編集します。

> **Important**
>
> カスタムメッセージには、`{username}` および `{####}` のプレースホルダーを含む必要があります。Amazon Cognito が招待メッセージを送信すると、これらのプレースホルダーはユーザーのユーザー名とパスワードに置き換えられます。

SMS メッセージの最大長は検証コードを含めて UTF-8 で 140 文字です。E メールメッセージの最大長は検証コードを含めて UTF-8 で 20,000 文字です。E メールメッセージに HTML タグを使用して、コンテンツの書式を設定できます。

E メールアドレスのカスタマイズ

デフォルトでは、Amazon Cognito は、no-reply@verificationemail.com からユーザープール内のユーザーに E メールメッセージを送信します。no-reply@verificationemail.com の代わりにカスタムの FROM と REPLY-TO メールアドレスを指定するか選択できます。

FROM と REPLY-TO の E メールアドレスをカスタマイズするには

1. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. [Messaging] (メッセージング) タブを選択します。[Email] (E メール) で、[Edit] (編集) を選択します。

4. [SES Region] (SES リージョン)を選択します。
5. 選択した [SES Region] (SES リージョン) の Amazon SES で検証した E メールアドレスリストから [FROM email address] (FROM E メールアドレス) を選択します。検証済みドメインのメールアドレスを使用する場合は、AWS Command Line Interface または AWS API で E メール設定を行います。詳細については、Amazon Simple Email Service デベロッパーガイドの「Amazon SES での E メールアドレスとドメインの検証」を参照してください。
6. 選択した[SES Region] (SES リージョン)の構成セットのリストから、[Configuration set] (構成セット) を選択します。
7. E メールメッセージ用に、わかりやすい [FROM sender name] (FROM 送信者名) を`John Stiles <johnstiles@example.com>`の形式で入力します。
8. 返信先 E メールアドレスをカスタマイズするには、[REPLY-TO email address (返信先 E メールアドレス)] フィールドに有効な E メールアドレスを入力します。

Amazon SES E メールを代理送信するための Amazon Cognito の承認 (カスタム REPLY-TO E メールアドレスからの送信)

Amazon Cognito は、デフォルトのアドレスではなく、カスタムの FROM メールアドレスからメールを送信するように設定できます。カスタムアドレスを使用するには、Amazon SES で検証済みの ID から E メールメッセージを送信できる許可を Amazon Cognito に付与する必要があります。ほとんどの場合、送信承認ポリシーを作成することで許可が付与できます。詳細については、Amazon Simple Email Service デベロッパーガイドの「Amazon SES での送信承認の使用」を参照してください。

メールメッセージに Amazon SES の使用許可をユーザープールを設定すると、Amazon Cognito はアカウントに `AWSServiceRoleForAmazonCognitoIdpEmailService` ロールを作成して、Amazon SES へのアクセスを許可します。`AWSServiceRoleForAmazonCognitoIdpEmailService` サービスにリンクされたロールが使用される場合、送信承認ポリシーは必要ありません。ユーザープールのデフォルト E メール機能と、検証済みの Amazon SES ID の両方を FROM アドレスとして使用する場合のみ、送信承認ポリシーを追加する必要があります。

Amazon Cognito が作成するサービスリンクロールの詳細については、「Amazon Cognito のサービスリンクロールの使用」を参照してください。

次の送信承認ポリシーの例では、Amazon SES 検証済み ID を使用するための限定された権限が Amazon Cognito に付与されています。Amazon Cognito は、条件 `aws:SourceArn` のユーザープールと条件 `aws:SourceAccount` のアカウントの両方に代わって送信する場合にのみ、E メールメッ

セージを送信できます。その他の例については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES 送信承認ポリシーの例」を参照してください。

> **Note**
>
> この例では、「Sid」値はステートメントを一意に識別する任意の文字列です。ポリシー構文の詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES 送信承認ポリシー」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

Amazon Cognito コンソールは、ドロップダウンメニューから Amazon SES の ID を選択するときに同様のポリシーを追加します。CLI または API を使用してユーザープールを設定する場合は、前の例のように構造化されたポリシーを Amazon SES アイデンティティにアタッチする必要があります。

管理者としてのユーザーアカウントの作成

ユーザープールを作成したら、AWS Management Console、AWS Command Line Interface または Amazon Cognito API を使用してユーザーを作成できます。ユーザープールで新しいユーザーのプロファイルを作成し、サインアップ手順を含めたウェルカムメッセージをそのユーザーに SMS または E メールで送信できます。

デベロッパーと管理者は以下のタスクを実行できます。

- AWS Management Console を使用するか、`AdminCreateUser` API を呼び出すことで、新しいユーザープロファイルを作成します。
- ユーザー属性値を設定します。
- カスタム属性を作成します。
- `AdminCreateUser` API リクエストにイミュータブルなカスタム属性の値を設定します。この機能は Amazon Cognito コンソールでは利用できません。
- 一時パスワードを指定する、または Amazon Cognito が一時パスワードを自動生成できるようにする。
- 提供された E メールアドレスと電話番号が新しいユーザーに対して確認済みとマークされているかどうかを指定する。
- AWS Management Console またはカスタムメッセージ Lambda トリガーを使用して、新しいユーザー向けのカスタム SMS および E メール招待メッセージを指定する。詳細については、「Lambda トリガーを使用したユーザープールワークフローのカスタマイズ」を参照してください。
- 招待メッセージが SMS、E メール、その両方のいずれで送信されるかを指定する。
- `AdminCreateUser` パラメータに `RESEND` を指定して、`MessageAction` API を呼び出すことで、既存のユーザーにウェルカムメッセージを再送信する。

> **Note**
>
> このアクションは現在、AWS Management Console を使用して実行することはできません。

- ユーザー作成時の招待メッセージが送信されないようにする。
- ユーザーアカウントの有効期限を指定する (最大 90 日)。
- ユーザーに自己サインアップを許可したり、管理者にのみ新しいユーザーの追加を許可したりする。

管理者またはデベロッパーによって作成されるユーザーの認証フロー

これらのユーザーの認証フローには、新しいパスワードを送信するためや、必須の属性の未設定値を指定するための、追加の手順が含まれます。これらの手順は以下に説明しており、手順 5、6、7 はこれらのユーザーに固有のものです。

1. ユーザーは、ユーザー名とパスワードを送信することで、初回のログインを開始します。
2. SDK が `InitiateAuth(Username, USER_SRP_AUTH)` を呼び出します。
3. Amazon Cognito が Salt & Secret ブロックで `PASSWORD_VERIFIER` チャレンジを返します。
4. SDK が SRP 計算を実行し、`RespondToAuthChallenge(Username, <SRP variables>, PASSWORD_VERIFIER)` を呼び出します。
5. Amazon Cognito は `NEW_PASSWORD_REQUIRED` チャレンジを返します。このチャレンジの本文には、ユーザーの現在の属性、およびユーザーのプロファイル内で現時点で値がないユーザープールの必須属性が含まれます。詳細については、「RespondToAuthChallenge」を参照してください。
6. ユーザーは新しいパスワードと必須の属性の未設定値を入力するように求められます。
7. SDK が `RespondToAuthChallenge(Username, <New password>, <User attributes>)` を呼び出します。
8. ユーザーに MFA の 2 番目の要素が必要な場合、Amazon Cognito が SMS_MFA チャレンジを返し、コードが送信されます。
9. ユーザーが正常に自分のパスワードを変更し、必要に応じて属性の値を指定するか MFA を完了すると、ユーザーがサインインされ、トークンが発行されます。

ユーザーがすべてのチャレンジに合格すると、Amazon Cognito サービスがユーザーを確認済みとマークし、ユーザーに ID トークン、アクセストークン、および更新トークンを発行します。詳細については、「ユーザープールでのトークンの使用」を参照してください。

AWS Management Console での新しいユーザーの作成

Amazon Cognito コンソールを使用して、ユーザーパスワードの要件を設定し、ユーザーに送信される招待メッセージと確認メッセージを設定し、新しいユーザーを追加できます。

パスワードポリシーを設定し、自己登録を有効にする

パスワードの複雑さを最小限に抑え、ユーザープールでパブリック API を使用してユーザーがサインアップできるかどうかを設定できます。

パスワードポリシーの設定

1. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. [Sign-in experience] (サインインエクスペリエンス) タブを選択して、[Password policy] (パスワードポリシー) を検索します。[Edit] (編集) を選択します。
4. [Custom] (カスタム) の [Password policy mode] (パスワードポリシーモード) を選択します。
5. [Password minimum length] (パスワードの最小長) を選択します。パスワード長要件の制限については、「ユーザープールのリソースクォータ」を参照してください。
6. [Password complexity] (パスワードの複雑さ) 要件を選択してください。
7. 管理者が設定したパスワードの有効期間を選択します。
8. [Save changes] (変更の保存) をクリックします。

セルフサービスサインアップを許可する

1. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. [Sign-up experience] (サインアップエクスペリエンス) タブを選択して、[Self-service sign-up] (セルフサービスサインアップ) を検索します。[Edit] (編集) を選択します。
4. [Enable self-registration] (自己登録を有効化) するかどうかを選択します。自己登録は、通常、クライアントシークレット、または AWS Identity and Access Management (IAM) API 認証情報を配布せずにユーザープールに新しいユーザーを登録する必要があるパブリッククアプリケーションクライアントで使用されます。

   > **自己登録の無効化**
   >
   > 自己登録を有効にしない場合は、IAM API 認証情報、またはフェデレーションプロバイダーとのサインインを使用して管理 API アクションで新しいユーザーを作成する必要があります。

5. [Save changes] (変更の保存) をクリックします。

E メールメッセージと SMS メッセージのカスタマイズ

ユーザーメッセージのカスタマイズ

Amazon Cognito がユーザーにサインインするように招待したとき、ユーザーがユーザーアカウントへのサインアップ、またはサインインして多要素認証 (MFA) を要求される際に Amazon Cognito はユーザーに送信するメッセージをカスタマイズできます。

> **Note**
>
> [Invitation message] (招待メッセージ) はユーザープールにユーザーを作成し、サインインに招待すると送信されます。Amazon Cognito はユーザーの E メールアドレスまたは電話番号に初期サインイン情報を送信します。
> 検証メッセージは、ユーザーがユーザープールのユーザーアカウントに登録したときに送信されます。Amazon Cognito はユーザーにコードを送信します。ユーザーが Amazon Cognito にコードを提供すると、連絡先情報を確認し、サインインのためにアカウントを確認します。検証コードは 24 時間有効です。
> [MFA message] (MFA メッセージ) は、ユーザープールで SMS MFA を有効にし、SMS MFA を構成したユーザーがサインインして MFA を要求されたときに送信されます。

1. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. [Messaging] (メッセージング) タブで、[Message templates] (メッセージテンプレート) を検索します。[Verification messages] (検証メッセージ)、[Invitation messages] (招待メッセージ)、または [MFA messages] (MFA メッセージ) を選択して、[Edit] (編集) を選択します。
4. 選択したメッセージタイプのメッセージをカスタマイズします。

   > **Note**
   >
   > メッセージをカスタマイズするときは、メッセージテンプレート内のすべての変数を含める必要があります。例えば、変数の場合 {####} は含まれません。ユーザーにはメッセージアクションを完了するのに十分な情報がありません。
   > 詳細については、「メッセージテンプレート」を参照してください。

5. a. 検証メッセージ

i. [Email] (E メール) メッセージの [Verification type] (検証タイプ) を選択します。[Code] (コード) 検証は、ユーザーが入力する必要がある数値コードを送信します。[Link] (リンク) 検証は、ユーザーがクリックして連絡先情報を検証できるリンクを送信します。変数内のテキスト [Link] (リンク) メッセージはハイパーリンクテキストとして表示されます。例えば、変数 {# #Click here##} を使用したメッセージテンプレートは、E メールメッセージで「Click here (ここをクリックしてください) 」というように表示されます。

ii. [Email] (E メール) メッセージの [Email subject] (E メールの件名) を入力します。

iii. [Email] (E メール) メッセージのカスタムの [Email message] (E メールメッセージ) のテンプレートを入力します。このテンプレートは HTML でカスタマイズできます。

iv. [SMS] メッセージのカスタムの [SMS message] (SMS メッセージ) のテンプレートを入力します。

v. [Save changes] (変更の保存) をクリックします。

b. [Invitation messages] (招待メッセージ)

i. [Email] (E メール) メッセージの [Email subject] (E メールの件名) を入力します。

ii. [Email] (E メール) メッセージのカスタムの [Email message] (E メールメッセージ) のテンプレートを入力します。このテンプレートは HTML でカスタマイズできます。

iii. [SMS] メッセージのカスタムの [SMS message] (SMS メッセージ) のテンプレートを入力します。

iv. [Save changes] (変更の保存) をクリックします。

c. MFA メッセージ

i. [SMS] メッセージのカスタムの [SMS message] (SMS メッセージ) のテンプレートを入力します。

ii. [Save changes] (変更の保存) をクリックします。

ユーザーの作成

ユーザーの作成

Amazon Cognito コンソールからユーザープールの新しいユーザーを作成できます。通常、ユーザーはパスワードを設定した後にサインインできます。E メールアドレスでサインインするには、ユーザーは `email` 属性を確認する必要があります。電話番号でサインインするには、ユーザーは `phone_number` 属性を確認する必要があります。管理者としてのアカウントを確認するには、AWS

CLI または API を使用するか、フェデレーション ID プロバイダーを使用してユーザープロファイルを作成することもできます。詳細については、「Amazon Cognito API Reference」を参照してください。

1. [Amazon Cognito console] (Amazon Cognito コンソール) に移動し、[User Pools] (ユーザープール) を選択します。
2. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
3. [Users] (ユーザー) タブを選択して、[Create use] (ユーザーを作成) をクリックします。
4. パスワード要件、使用可能なアカウント回復方法、およびユーザープールのエイリアス属性に関するガイダンスは、「ユーザープールのサインインとセキュリティ要件」を参照してください。
5. 招待メッセージの送信方法を選択します。SMS メッセージ、E メールメッセージ、または両方を選択します。

   > **Note**
   >
   > 招待メッセージを送信する前に、ユーザープールの [Messaging] (メッセージング) タブで、Amazon Simple Notification Service および Amazon Simple Email Service で送信者と AWS リージョン を設定します。受信者メッセージとデータレートが適用されます。Amazon SES は、メールメッセージの請求を別途請求し、Amazon SNS は SMS メッセージについて別途請求します。

6. 新規ユーザー用に [Username] (ユーザー名) を選択します。
7. [Create a password] (パスワードを作成する) または、Amazon Cognito にユーザーの [Generate a password] (パスワードを生成する) を許可するかどうかを選択してください。一時パスワードは、ユーザープールのパスワードポリシーに準拠する必要があります。
8. [Create] (作成) を選択します。
9. [Users] (ユーザー) タブを選択し、ユーザーの [User name] (ユーザー名) エントリを選択します。[User attributes] (ユーザー属性) および [Group memberships] (グループメンバーシップ) を追加して編集します。ユーザーイベント履歴の確認

ユーザープールにグループを追加する

Amazon Cognito ユーザープール内のグループのサポートにより、ユーザーの作成と管理、グループへのユーザーの追加、およびグループからのユーザーの削除が可能になります。グループを使用して、ユーザーのコレクションを作成してそのアクセス権限を管理したり、異なるタイプのユーザーを

表したりできます。 AWS Identity and Access Management (IAM) ロールをグループに割り当てて、グループのメンバーのアクセス許可を定義できます。

グループを使用して、ユーザープールでユーザーのコレクションを作成できます。この操作は、これらのユーザーのアクセス権限を設定するためによく行われます。例えば、ウェブサイトやアプリの読者、寄稿者、および編集者であるユーザーに対して別個のグループを作成できます。また、グループに関連付けられた IAM ロールを使用することで、これらの異なるグループに異なる許可を設定して、コントリビュータのみがコンテンツを Amazon S3 に配置し、エディタのみが Amazon API Gateway の API を通じてコンテンツをパブリッシュできるようにすることも可能です。

、APIs、および CLI から AWS Management Consoleユーザープール内のグループを作成および管理できます。開発者 (AWS 認証情報を使用) として、ユーザープールのグループを作成、読み取り、更新、削除、および一覧表示できます。また、グループに対してユーザーを追加、削除できます。

ユーザープール内のグループを使用しても追加コストは発生しません。詳細については、「Amazon Cognito の料金」を参照してください。

グループへの IAM ロールの割り当て

グループを使用して、IAM ロールでリソースへの許可を制御できます。IAM ロールには、信頼ポリシーと許可ポリシーが含まれます。ロールの信頼ポリシーでは、ロールを使用できるユーザーを指定します。アクセス許可ポリシーでは、グループメンバーがアクセスできるアクションとリソースを指定します。IAM ロールを作成するときに、グループユーザーがロールを引き受けることを許可するロールの信頼ポリシーを設定します。ロールの許可ポリシーで、グループに付与する許可を指定します。

Amazon Cognito でグループを作成するときは、ロールの ARNを指定して IAM ロールを指定します。グループメンバーが Amazon Cognito を使用してサインインすると、ID プールから一時的な認証情報を受け取ることができます。それらの許可は、関連付けられた IAM ロールによって決まります。

各ユーザーは複数のグループに属することができます。デベロッパーとして、ユーザーが複数のグループに属している場合に IAM ロールを自動的に選択するための以下のオプションがあります。

- 各グループに優先順位の値を割り当てることができます。優先順位が高い (低い) グループが選択され、それに関連付けられた IAM ロールが適用されます。
- アプリは、 GetCredentialsForIdentityCustomRoleARNパラメータでロール ARN を指定することで、ID プールを通じてユーザーの認証情報をリクエスト AWS するときに、使用可能なロールから選択することもできます。指定された IAM ロールは、ユーザーが利用できるロールに一致する必要があります。

グループへの優先順位の値の割り当て

ユーザーは複数のグループに属することができます。ユーザーのアクセストークンと ID トークンの `cognito:groups` クレームには、ユーザーが属するすべてのグループのリストが含まれます。`cognito:roles` クレームには、グループに対応するロールのリストが含まれます。

ユーザーは複数のグループに属することができるので、各グループに優先順位を割り当てることができます。これは、ユーザーがユーザープールで属するその他のグループに対するこのグループの優先順字を指定する正数です。ゼロが最優先順位値です。低い優先順位の値を持つグループは、高い優先順位の値または null 値を持つグループよりも優先されます。ユーザーが複数のグループに属している場合、ユーザーの ID トークンの `cognito:preferred_role` クレームに適用される IAM ロールは、優先順位の値が最も低いグループのものになります。

2 つのグループは、同じ優先順位の値を持つことができます。その場合は、どちらのグループも他方に対して優先されません。同じ優先順位の値を持つ 2 つのグループのロール ARN が同じである場合、そのロールは、各グループのユーザーの ID トークンの `cognito:preferred_role` クレームで使用されます。2 つのグループのロール ARN が異なる場合、`cognito:preferred_role` クレームは、ユーザーの ID トークンで設定されません。

Amazon API Gateway を使用した許可の管理でのグループの使用

Amazon API Gateway を使用した許可の管理には、ユーザープールのグループを使用することができます。ユーザーがメンバーであるグループは、IDトークンと `cognito:groups` クレームのユーザープールからのアクセストークンの両方に含まれています。リクエストとともに ID またはアクセストークンを Amazon API Gateway に送信し、REST API に Amazon Cognito ユーザープールオーソライザーを使用できます。詳細については、API Gateway デベロッパーガイドの「Amazon Cognito ユーザープールをオーソライザーとして使用して REST API へのアクセスを制御する」を参照してください

カスタム JWT オーソライザーを使用して Amazon API Gateway HTTP API へのアクセスを承認することもできます。詳細については、API Gateway デベロッパーガイドの「JWT オーソライザーを使用した HTTP API へのアクセスの制御」を参照してください。

グループの制限

ユーザーグループには、次の制限が適用されます。

- 作成できるグループの数は、Amazon Cognito サービスクォータ によって制限されます。
- グループはネストできません。

- グループのユーザーを検索することはできません。
- グループを名前で検索することはできませんが、グループをリストすることはできます。

AWS Management Consoleでの新しいグループの作成

以下の手順に従って、新しいグループを作成します。

新しいグループを作成する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Group] (グループ) タブを選択してから、[Create a Groups] (グループの作成) を選択します。
5. [Create a group] (グループを作成する) ページで、[Group name] (グループ名) に新しいグループのフレンドリ名を入力します。
6. オプションで、次のフィールドのいずれかを使用して、このグループに関する追加情報を指定できます。

   - [Description] (説明) - この新しいグループの使用目的の詳細を入力します。
   - [Precedence] (優先順位) - Amazon Cognito は、優先度が低いグループに属するグループに基づいて、特定のユーザーに対するすべてのグループ権限を評価し、適用します。優先順位が低いグループが選択され、それに関連付けられた IAM ロールが適用されます。詳細については、「グループへの優先順位の値の割り当て」を参照してください。
   - [IAM role] (IAM ロール) -リソースへのアクセス許可を制御する必要がある場合は、グループに IAM ロールを割り当てることができます。ID プールを持つユーザープールを統合する場合、[IAM ロール] の設定によって、トークンからロールを選択するよう ID プールが設定されているときに、ユーザーの ID トークンに割り当てられるロールが決定されます。詳細については、「グループへの IAM ロールの割り当て」を参照してください。
   - [Add users to this group] (ユーザーをこのグループに追加する) - 既存のユーザーを作成した後、このグループのメンバーとして追加します。
7. [Create] (作成) を選択して確定します。

ユーザーアカウントの管理と検索

ユーザープールを作成すると、AWS Management Console、AWS Command Line Interface または Amazon Cognito API を使用してユーザーを表示し、管理することができます。このトピックでは、AWS Management Console を使用してユーザーを表示および検索する方法について説明します。

ユーザー属性の表示

Amazon Cognito コンソールでユーザー属性を表示するには、以下の手順に従います。

ユーザー属性を表示するには

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Users] (ユーザー) タブをクリックし、リストからユーザーを選択します。
5. ユーザーの詳細ページの [User attributes] (ユーザー属性) で、ユーザーに関連付けられている属性を表示できます。

ユーザーのパスワードのリセット

Amazon Cognito コンソールでユーザーのパスワードをリセットするには、以下の手順に従います。

ユーザーのパスワードのリセット

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Users] (ユーザー) タブをクリックし、リストからユーザーを選択します。
5. ユーザーの詳細ページで、[Actions] (アクション)、[Reset password] (パスワードのリセット) を選択します。
6. [Reset password] (パスワードのリセット) ダイアログで、情報を確認し、準備ができたら [Reset] (リセット) を選択します。

この行動により、確認コードがユーザーに即時送信され、ユーザーの状態が `RESET_REQUIRED` に変更されることでユーザーの現在のパスワードが無効にされます。[Reset password] (ユーザーパスワードのリセット) コードは、1 時間有効です。

ユーザー属性の検索

ユーザープールがすでに作成されている場合は、AWS Management Consoleの [Users] (ユーザー) パネルから検索できます。また、[Filter] (フィルター) パラメータを受け入れる Amazon Cognitoの ListUsers API も使用できます。

以下のどの標準属性でも検索できます。カスタム属性は検索できません。

- username (大文字と小文字が区別されます)
- email
- phone_number
- name
- given_name
- family_name
- preferred_username
- cognito:user_status (コンソールでは [Status (ステータス)] となっています) (大文字と小文字が区別されません)
- status (コンソールでは [Enabled (有効)] となっています) (大文字と小文字が区別されます)
- sub

> **Note**
>
> また、クライアント側のフィルターを使用してユーザーを一覧表示することもできます。サーバー側のフィルターは、1 つ以上の属性に一致しません。高度な検索を行うには、AWS Command Line Interface での `list-users` のアクションの `--query` パラメータを用いたクライアント側のフィルターを使用します。クライアント側のフィルターを使用すると、ListUsers は 0 人以上のユーザーからなるページ分割されたリストを返します。結果がゼロで、複数のページを連続して受信できます。Null ページネーショントークン値を受け取るまで、返される各ページネーショントークンでクエリを繰り返し、組み合わせた結果を確認します。

サーバー側とクライアント側のフィルタリングの詳細については、ユーザーガイドの「AWS Command Line Interface での AWS CLI 出力の フィルタリング」を参照してください。

AWS Management Console を使用したユーザーの検索

ユーザープールがすでに作成されている場合は、AWS Management Consoleの [Users] (ユーザー) パネルから検索できます。

AWS Management Console は、常にプレフィックス ("先頭の文字") 検索を行います。

Amazon Cognito コンソールでユーザーを検索する

1. Amazon Cognito コンソールに移動します。AWS 認証情報を求められる場合があります。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Users] (ユーザー) タブを選択し、検索フィールドに、ユーザーのユーザーネームを入力します。[Username] (ユーザーネーム) など、一部の属性値では、大文字と小文字が区別される点に注意してください。

   また、検索フィルターを調整して、スコープを他のユーザープロパティ ([Email] (E メール)、[Phone number] (電話番号)、または [Last name] (姓)など) に絞り込むことで、ユーザーを検索することもできます。

`ListUsers` API を使用したユーザーの検索

アプリからユーザーを検索するには、Amazon Cognito の ListUsers API を使用します。この API は以下のパラメータを使用します。

- `AttributesToGet`: 文字列の配列です。各文字列は、検索結果でユーザーごとに返されるユーザー属性の名前です。すべての属性を取得するには、`AttributesToGet` パラメータを含めないか、またはリテラル文字列の値 `null` を使って `AttributesToGet` をリクエストしてください。
- `Filter`:「 ""」形式のフィルター文字列です。`AttributeNameFilter-TypeAttributeValue`フィルター文字列内の引用符は、円記号 (\) を使用してエスケープする必要があります。例えば、`"family_name = \"Reddy\""` です。フィルター文字列が空の場合、`ListUsers` はユーザープールのすべてのユーザーを返します。
  - `AttributeName`: 検索する属性の名前。同時に検索できる属性は 1 つのみです。

> **Note**
>
> 標準属性のみを検索できます。カスタム属性は検索できません。これはインデックスが付けられた属性のみが検索可能なためで、カスタム属性にインデックスを作成することはできません。

  - `Filter-Type`: 完全一致を検索する場合は、`=` を使用します (`given_name = "Jon"` など)。プレフィックス (「先頭の文字」) 一致を検索する場合、`^=` を使用します (`given_name ^= "Jon"` など)。
  - `AttributeValue`: ユーザーごとに一致する必要がある属性値です。
- `Limit`: 返されるユーザーの最大数です。
- `PaginationToken`: 前の検索からさらに結果を取得するためのトークンです。Amazon Cognito は 1 時間後にページ分割トークンを期限切れにします。
- `UserPoolId`: 検索を実行する必要があるユーザープールのユーザープール ID です。

すべての検索で大文字と小文字が区別されません。検索結果は、`AttributeName` 文字列により指定された属性によって昇順に並べ替えられます。

`ListUsers` API の使用例

次の例はすべてのユーザーを返します。すべての属性が含まれています。

```
{
    "AttributesToGet": null,
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

次の例は、電話番号の先頭が「+1312」のすべてのユーザーを返します。また、すべての属性が含まれています。

```
{
    "AttributesToGet": null,
```

```
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

次の例は、姓が「Reddy」の先頭 10 人のユーザーを返します。各ユーザーの検索結果には、ユーザーの名、電話番号、E メールアドレスが含まれています。ユーザープールに存在する一致ユーザーが 10 人を超える場合、レスポンスにはページ分割トークンが含まれています。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

前の例でページ分割トークンが返された場合、次の例は同じフィルター文字列と一致する次の 10 人のユーザーを返します。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

ユーザーアカウントの復旧

AccountRecoverySetting パラメータを使用すると、ユーザーが ForgotPassword API を呼び出したときにパスワードの復旧に使用できる方法をカスタマイズできます。ForgotPassword は、検証済みの E メールまたは検証済みの電話番号に回復用コードを送信します。回復用コードは

1 時間有効です。ユーザープールに AccountRecoverySetting を指定した場合、Amazon Cognito は、設定した優先度に基づいてコードの配信先を選択します。

AccountRecoverySetting を定義し、ユーザーに SMS MFA が設定されている場合、SMS をアカウント復旧メカニズムとして使用することはできません。この設定の優先度は、1 が最も高いと優先度となります。Cognito は、指定された方法の 1 つだけに検証を送信します。

たとえば admin_only は、管理者がユーザー自身でアカウントを復旧することを望まず、その代わりに管理者に連絡してアカウントをリセットするようリクエストする場合に使用される値です。他のアカウント復旧メカニズムでは admin_only を使用できません。

AccountRecoverySetting を指定しない場合、Amazon Cognito はレガシーメカニズムを使用してパスワード復旧方法を決定します。この場合、Cognito は最初に確認済みの電話番号を使用します。ユーザーの確認済み電話番号が見つからない場合、Cognito はフォールバックし、次に確認済みの E メールを使用します。

AccountRecoverySetting の詳細については、Amazon Cognito ID プロバイダー API リファレンスの「CreateUserPool」と「UpdateUserPool」を参照してください。

パスワードを忘れた場合の対応

ユーザーがパスワードを忘れた場合や忘れたパスワードを確認する場合のアクションの一環として、パスワードのリセットコードをリクエストしたり、入力したりする試行を所定時間内に 5 ~ 20 回許可します。正確な回数は、リクエストに関連付けられたリスクパラメータによって異なります。この対応は変更される場合があることに注意してください。

ユーザープールへのユーザーのインポート

既存のユーザーディレクトリまたはユーザーデータベースから Amazon Cognito にユーザーをインポートまたは移行する方法は 2 つあります。1 つは、ユーザー移行の Lambda トリガーを使用して、ユーザーが Amazon Cognito を使用して初めてサインインするときにユーザーを移行する方法です。この方法では、ユーザーが既存のパスワードを引き続き使用でき、ユーザープールへの移行後にパスワードをリセットする必要はありません。もう 1 つは、すべてのユーザーのユーザープロファイル属性を含む CSV ファイルをアップロードして、ユーザーを一括で移行する方法です。以降のセクションでは、これらの両方のアプローチについて説明します。

トピック

- ユーザー移行の Lambda トリガーを使用したユーザープールへのユーザーのインポート
- CSV ファイルからユーザープールへのユーザーのインポート

ユーザー移行の Lambda トリガーを使用したユーザープールへのユーザーのインポート

このアプローチでは、ユーザーがアプリケーションで初めてサインインするとき、またはパスワードのリセットをリクエストするときに、既存のユーザーディレクトリからユーザープールにユーザーをシームレスに移行できます。ユーザー移行の Lambda トリガー 関数をユーザープールに追加すると、サインインしようとするユーザーに関するメタデータを受け取り、外部 ID ソースからユーザープロファイル情報を返します。この Lambda トリガーの詳細 (リクエストとレスポンスのパラメータなど) とサンプルコードについては、「ユーザー移行の Lambda トリガーのパラメータ」を参照してください。

ユーザーの移行を開始する前に、AWS アカウント でユーザー移行の Lambda 関数を作成し、ユーザープールに Lambda 関数 をユーザー移行トリガーとして設定します。Amazon Cognito サービスアカウントプリンシパルのみを許可する認証ポリシーを Lambda 関数に追加します。`cognito-idp.amazonaws.com` は、Lambda 関数を呼び出し、独自のユーザープールのコンテキスト内でのみ実行します。詳細については、「AWS Lambda のリソースベースのポリシーを使用する (Lambda 関数ポリシー)」を参照してください。

サインインプロセス

1. ユーザーがアプリを開き、Amazon Cognito ユーザープール API または Amazon Cognito ホストされた UI を使用してサインインします。Amazon Cognito API でサインインを容易にする方法の詳細については、「Amazon Cognito の認証と承認を、ウェブアプリケーションとモバイルアプリケーションに統合する」を参照してください。
2. アプリケーションはユーザー名とパスワードを Amazon Cognito に送信します。アプリケーションに AWS SDK を使用して構築したカスタムサインイン UI がある場合、アプリケーションは、`USER_PASSWORD_AUTH` または `ADMIN_USER_PASSWORD_AUTH` フローで InitiateAuth または AdminInitiateAuth を使用する必要があります。アプリがこれらのフローのいずれかを使用する場合、SDK はパスワードをサーバーに送信します。

> **Note**
>
> ユーザー移行トリガーを追加する前に、アプリケーションクライアントの設定で `USER_PASSWORD_AUTH` または `ADMIN_USER_PASSWORD_AUTH` フローを有効化します。デフォルトの `USER_SRP_AUTH`フロー の代わりにこれらのフローを使用する必要があります。Amazon Cognito は、他のディレクトリでのユーザーの認証を検証できるよう

に、Lambda 関数にパスワードを送信する必要があります。SRP は、Lambda 関数からユーザーのパスワードを隠します。

3. Amazon Cognito は、送信されたユーザー名がユーザープール内のユーザー名またはエイリアスと一致するかどうかをチェックします。ユーザーの電子メールアドレス、電話番号、または優先ユーザー名を、ユーザープールのエイリアスとして設定できます。ユーザーが存在しない場合、Amazon Cognito は、ユーザー名やパスワードなどのパラメータを ユーザー移行の Lambda トリガー 関数に送信します。
4. ユーザー移行の Lambda トリガー 関数は、既存のユーザーディレクトリまたはユーザーデータベースを使用してユーザーをチェックまたは認証します。この関数は、Amazon Cognito がユーザープールのユーザープロファイルに保存するユーザー属性を返します。送信されたユーザー名がエイリアス属性と一致する場合にのみ、`username` パラメータを返します。ユーザーが既存のパスワードを引き続き使用できるようにする場合は、Lambda レスポンスで属性 `finalUserStatus` を `CONFIRMED` に設定します。アプリケーションは、ユーザー移行の Lambda トリガーのパラメータ に示されるすべての `"response"` パラメータを返す必要があります。

> **Important**
>
> リクエストイベントオブジェクト全体をユーザー移行 Lambda コードに記録しないでください。このリクエストイベントオブジェクトには、ユーザーのパスワードが含まれます。ログをサニタイズしない場合、パスワードが CloudWatch Logs に表示されます。

5. Amazon Cognito はユーザープールにユーザープロファイルを作成し、トークンをアプリケーションクライアントに返します。
6. アプリケーションはトークンの取り込みを実行し、ユーザー認証を受け入れ、リクエストされたコンテンツに進みます。

ユーザーを移行したら、`USER_SRP_AUTH` を使用してサインインします。Secure Remote Password (SRP) プロトコルは、パスワードをネットワーク上に送信しないため、移行時に使用する `USER_PASSWORD_AUTH` フローよりもセキュリティ上の利点が大きくなります。

移行中にクライアントデバイスやネットワークの問題などのエラーが発生した場合、アプリケーションは Amazon Cognito ユーザープール API からエラーレスポンスを受け取ります。この場合 Amazon Cognito はユーザープールにユーザーアカウントを作成しない可能性があります。その後、ユーザーは再度サインインを試行する必要があります。サインインが繰り返し失敗した場合、アプリ

ケーションでパスワードを忘れた場合のフローを使用してユーザーのパスワードをリセットする必要があります。

パスワードを忘れた場合のフローでは、`UserMigration_ForgotPassword` イベントソースを使用して ユーザー移行の Lambda トリガー 関数も呼び出します。ユーザーはパスワードのリセットをリクエストしてもパスワードを送信しないため、Amazon Cognito は Lambda 関数に送信するイベントにパスワードを含めません。関数では、既存のユーザーディレクトリ内のユーザーのみを検索し、ユーザープールのユーザープロファイルに追加する属性を返すことができます。関数が呼び出しを完了して Amazon Cognito に応答を返すと、ユーザープールはパスワードリセットコードを E メールまたは SMS で送信します。アプリケーションで、ユーザーに確認コードと新しいパスワードの入力を促し、その情報を ConfirmForgotPassword API リクエストで Amazon Cognito に送信します。。Amazon Cognito がホストする UI で、パスワードを忘れた場合のフロー用の組み込みページを使用することもできます。

CSV ファイルからユーザープールへのユーザーのインポート

Amazon Cognito ユーザープールにユーザーをインポートできます。ユーザー情報は、特別にフォーマットされた .csv ファイルからインポートされます。インポートプロセスは、[パスワード] 以外のすべてのユーザー属性の値を設定します。セキュリティのベストプラクティスでは、パスワードはプレーンテキスト形式として使用できない必要があり、ハッシュのインポートはサポートされていないため、パスワードのインポートはサポートされません。つまり、ユーザーは最初にサインインした際にパスワードを変更する必要があります。そのため、このメソッドを使用してインポートすると、ユーザーは RESET_REQUIRED 状態になります。

`Permanent` パラメータを `true` に設定する AdminSetUserPassword API リクエストでユーザーのパスワードを設定できます。

> **Note**
>
> 各ユーザーの作成日は、ユーザーがユーザープールにインポートされた時間です。作成日は、インポートされた属性の 1 つではありません。

基本的なステップは次のとおりです。

1. AWS Identity and Access Management (IAM) コンソールで Amazon CloudWatch Logs ロールを作成します。
2. ユーザーインポート .csv ファイルを作成します。
3. ユーザーインポートジョブを作成し、実行します。

4. ユーザーインポート .csv ファイルをアップロードします。
5. ユーザーインポートジョブを起動し、実行します。
6. CloudWatch を使用してイベントログを確認します。
7. インポートされたユーザーがパスワードをリセットするように要求します。

トピック

- CloudWatch Logs IAM ロールの作成
- ユーザーインポート CSV ファイルの作成
- Amazon Cognito ユーザープールのインポートジョブの作成と実行
- ユーザープールのインポート結果を CloudWatch コンソールに表示
- インポートされたユーザーにパスワードをリセットするように要求

CloudWatch Logs IAM ロールの作成

Amazon Cognito の CLI または API を使用している場合は、CloudWatch IAM ロールを作成する必要があります。以下の手順では、Amazon Cognito がインポートジョブの結果を CloudWatch Logs に書き込むために使用できる IAM ロールを作成する方法について説明します。

> **Note**
>
> Amazon Cognito コンソールでインポート ジョブを作成すると、同時に IAM ロールを作成できます。新しい IAM ロールを作成することを選択すると、Amazon Cognito は適切な信頼ポリシーと IAM ポリシーをロールに自動的に適用します。

ユーザープールインポート用の CloudWatch Logs IAM ロールを作成するには (AWS CLI、API)

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. AWS のサービス 用の新しい IAM ロールを作成します。詳しい手順については、AWS Identity and Access Management ユーザーガイドの「AWS のサービス にアクセス許可を委任するロールの作成」を参照してください。
   a. 信頼されたエンティティタイプのユースケースを選択するときは、任意のサービスを選択してください。Amazon Cognito は現在、サービスのユースケースに記載されていません。

b. [Add permissions] (アクセス許可の追加) 画面で、[Create policy] (ポリシーの作成) を選択し、次のポリシーステートメントを挿入します。*REGION* をユーザープールの AWS リージョン に置き換えます (us-east-1 など)。*ACCOUNT* を AWS アカウント ID に置き換えます (111122223333 など)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"
            ]
        }
    ]
}
```

3. ロールを作成したときに Amazon Cognito を信頼されたエンティティとして選択しなかったため、ロールの信頼関係を手動で編集する必要があります。IAM コンソールのナビゲーションペインから [Roles] (ロール) を選択し、作成した新しいロールを選択します。
4. [信頼関係] タブを選択します。
5. [Edit trust policy] (信頼ポリシーを編集) を選択します。
6. 次のポリシーステートメントを [Edit trust policy] (信頼ポリシーを編集) に貼り付け、既存のテキストを置き換えます。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "cognito-idp.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
```

```
            }
        ]
    }
```

7. [ポリシーの更新] を選択します。
8. ロールの ARN を書き留めておきます。インポートジョブを作成するときに、ARN を指定します。

ユーザーインポート CSV ファイルの作成

既存のユーザーをユーザープールにインポートする前に、インポートするユーザーとその属性を含むカンマ区切り値 (CSV) ファイルを作成する必要があります。ユーザープールから、ユーザープールの属性スキーマを反映するヘッダーを含むユーザーインポートファイルを取得できます。その後、CSV ファイルのフォーマット のフォーマット要件に一致するユーザー情報を挿入できます。

CSV ファイルヘッダーのダウンロード (コンソール)

CSV ヘッダーファイルをダウンロードするには、次の手順に従います。

CSV ファイルヘッダーをダウンロードするには

1. Amazon Cognito コンソールに移動します。AWS 認証情報を求められる場合があります。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Users] (ユーザー) タブを選択します。
5. [Import users] (ユーザーのインポート) セクションで、[Create an import job] (インポートジョブの作成) を選択します。
6. [Upload CSV] (CSV のアップロード) で [template.csv] リンクを選択し、CSV ファイルをダウンロードします。

CSV ファイルヘッダーのダウンロード (AWS CLI)

正しいヘッダーのリストを取得するには、*USER_POOL_ID* がユーザーをインポートするユーザープールのユーザープール ID である次の CLI コマンドを実行します。

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

レスポンス例:

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
        "profile",
        "picture",
        "website",
        "email",
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
        "cognito:mfa_enabled",
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

CSV ファイルのフォーマット

ダウンロードしたユーザーインポート CSV ヘッダーファイルは次の文字列のようになります。ユーザープールに追加したカスタム属性も含まれます。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
```

ユーザーのためのこのヘッダーと属性値が含まれ、次のルールに従ってフォーマットされているように CSV ファイルを編集します。

> **Note**
>
> 電話番号の適切な形式など属性値の詳細については、「ユーザープール属性」を参照してください。

- ファイルの最初の行はユーザー属性の名前を含む、ダウンロードされたヘッダーの行です。
- CSV ファイルの行の順序は重要ではありません。
- 最初の行の後の各行に、ユーザーの属性値が含まれます。
- ヘッダーのすべての列が存在している必要がありますが、列の値を指定する必要はありません。
- 以下の属性は必須です:
  - cognito:username
  - cognito:mfa_enabled
  - [email_verified] または [phone_number_verified]
    - 自動確認された属性の少なくとも一つは、各ユーザーに対して `true` である必要があります。自動検証属性は、新しいユーザーがユーザープールに参加したときに Amazon Cognito が自動的にコードを送信する E メールアドレスまたは電話番号です。
    - ユーザープールは [email_verified] または [phone_number_verified] の少なくとも一つの自動確認された属性が必要です。ユーザープールに自動確認された属性が存在しない場合は、インポートジョブは開始されません。
    - ユーザープールに一つの自動確認された属性がある場合のみ、その属性はユーザーごとに確認する必要があります。たとえば、ユーザープールに自動確認された属性として [phone_number] だけがある場合、[phone_number_verified] 値はそれぞれのユーザーに対して `true` である必要があります。

    > **Note**
    >
    > ユーザーがパスワードをリセットするには、検証済みの E メールまたは電話番号が必要です。Amazon Cognito は、CSV ファイルで指定された E メールアドレスまたは電話番号にパスワードのリセットコードが含まれるメッセージを送信します。メッセージが電話番号に送信される場合は、SMS メッセージで送信されます。詳細については、「サインアップ時に連絡先情報を検証する」を参照してください。

  - [email] ([email_verified] が `true` の場合)
  - [phone_number] ([phone_number] が `true` の場合)
  - ユーザープールを作成する際に必須と指定した任意の属性
- 文字列である属性値には、引用符は使用できません。
- 属性値にカンマが含まれる場合、カンマの前にバックスラッシュ (\) を置く必要があります。これは、CSV ファイルのフィールドがカンマで区切られているためです。
- CSV ファイルのコンテンツは、バイトオーダーマークのない UTF-8 形式にする必要があります。

- [cognito:username] フィールドは必須であり、ユーザープール内で一意である必要があります。任意の Unicode 文字列を使えます。ただし、スペースまたはタブを含めることはできません。
- [birthdate] (生年月日) の値がある場合、「*mm/dd/yyyy*」の形式である必要があります。たとえば、生年月日が 1985 年 2 月 1 日だとすると、**02/01/1985** にエンコードされる必要があります。
- [cognito:mfa_enabled] フィールドは必須です。ユーザープールで Multi-Factor Authentication (MFA) が必要であると設定している場合、このフィールドはすべてのユーザーに `true` である必要があります。MFA をオフに設定している場合、このフィールドはすべてのユーザーに `false` である必要があります。MFA をオプションとして設定している場合、このフィールドは `true` または `false` で、空にすることはできません。
- 最大行数は 16,000 文字です。
- CSV ファイルの最大サイズは 100 MB です。
- ファイルの最大行数 (ユーザー) は 500,000 です。この最大値にはヘッダー行は含まれません。
- [updated_at] フィールド値はエポック時間 (秒) であると想定します。たとえば **1471453471** などです。
- 属性値の先頭または末尾の空白は切り捨てられます。

次のリストは、カスタム属性のないユーザープールの CSV インポートファイルの例です。ユーザープールスキーマは、この例とは異なる場合があります。その場合は、ユーザープールからダウンロードした CSV テンプレートにテスト値を入力する必要があります。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

Amazon Cognito ユーザープールのインポートジョブの作成と実行

このセクションは、Amazon Cognito コンソールおよび AWS Command Line Interface (AWS CLI) を使用してユーザープールのインポートジョブを作成し、実行する方法を説明します。

トピック

- CSV ファイルからのユーザーのインポート (コンソール)
- ユーザーのインポート (AWS CLI)

CSV ファイルからのユーザーのインポート (コンソール)

次の手順は、CSV ファイルからユーザーをインポートする方法について説明します。

CSV ファイルからユーザーをインポートするには (コンソール)

1. Amazon Cognito コンソールに移動します。AWS 認証情報を求められる場合があります。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Users] (ユーザー) タブを選択します。
5. [Import users] (ユーザーのインポート) セクションで、[Create an import job] (インポートジョブの作成) を選択します。
6. [Create import job] (インポートジョブの作成) ページで、ジョブ名を入力します。
7. [Create a new IAM role] (新しい IAM ロールの作成)、または[Use an existing IAM role] (既存のロールを使用する) を選択します。
   a. [Create a new IAM role] (新しい IAM ロールの作成) を選択した場合は、新しいロールの名前を入力します。Amazon Cognito は、正しいアクセス許可と信頼関係を持つロールを自動的に作成します。インポートジョブを作成する IAM プリンシパルに、IAM ロールを作成するアクセス許可が必要です。
   b. [Use an existing IAM role] (既存の IAM ロールを使用する) を選択した場合は、[IAM role selection] (IAM ロール選択) のリストからロールを選択します。このロールには、CloudWatch Logs IAM ロールの作成 で説明されているアクセス許可と信頼ポリシーが必要です。
8. [Create job] (ジョブの作成) を選択してジョブを送信しますが、後で開始します。[Create and start job] (ジョブを作成して開始) を選択してジョブを送信し、すぐに開始します。
9. ジョブを作成したが開始しなかった場合は、後で開始できます。[Users] (ユーザー) タブの [Import users] (ユーザーのインポート) で、インポートジョブを選択し、[Start] (開始) を選択します。また、AWS SDK から StartUserImportJob API リクエストを送信することもできます。
10. ユーザーインポートジョブの進行状況は、[Import users] (ユーザーのインポート) の [Users] (ユーザー) タブで監視します。ジョブが成功しない場合は、[Status] (ステータス) 値を選択できます。詳細については、[View the CloudWatch logs for more details] (詳細については CloudWatch ログを表示する) を選択して、CloudWatch ログコンソールで問題がないか確認してください。

ユーザーのインポート (AWS CLI)

ユーザープールにユーザーをインポートするには、次の CLI コマンドが使用できます:

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

これらのコマンドに関するコマンドラインオプションのリストを取得するには、`help` コマンドラインオプションを使用します。次に例を示します。

```
aws cognito-idp get-csv-header help
```

ユーザーインポートジョブの作成

CSV ファイルを作成した後、次の CLI コマンドを使用してユーザーインポートジョブを作成します。*[JOB_NAME]* にはジョブに付ける名前を指定します。*[USER_POOL_ID]* には以前と同じユーザープール ID を指定します。*[ROLE_ARN]* には、CloudWatch Logs IAM ロールの作成 で次のとおり受け取ったロール ARN を指定します。

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id
 "USER_POOL_ID" --cloud-watch-logs-role-arn "ROLE_ARN"
```

レスポンスで返される *PRE_SIGNED_URL* は 15 分間有効です。この時間以後は無効になり、新規のユーザーインポートジョブを作成して新しい URL を取得し直す必要があります。

Example レスポンス例:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
```

```
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

ユーザーインポートジョブのステータス値

ユーザーインポートのコマンドに対する応答には、Status を示す次のような値が見つかります。

- Created - ジョブが作成されましたが、開始前の状態です。
- Pending - 遷移状態です。ジョブを開始しましたが、まだユーザーのインポートは開始していません。
- InProgress - ジョブが開始され、ユーザーインポートが進行中です。
- Stopping - ジョブを停止する処理中です。ユーザーインポートは停止していません。
- Stopped - ジョブは停止中で、ユーザーインポートも停止しています。
- Succeeded - ジョブは正常に完了しました。
- Failed - エラーのためジョブは停止しました。
- Expired - ジョブを作成しましたが、24 ~ 48 時間以内にジョブを起動しませんでした。ジョブに関連付けられたすべてのデータは削除され、ジョブを開始することはできません。

CSV ファイルのアップロード

以下の curl コマンドを使用して、ユーザーデータが含まれる CSV ファイルを create-user-import-job コマンドのレスポンスから取得した事前署名済みの URL にアップロードします。

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"
 "PRE_SIGNED_URL"
```

このコマンドの出力から、次のようなメッセージを見つけます: "We are completely uploaded and fine" このメッセージは、ファイルが正常にアップロードされたことを示します。

ユーザーインポートジョブの説明

ユーザーインポートジョブの説明を取得するには、次のコマンドを使用します。*USER_POOL_ID* はユーザープール ID です。*JOB_ID* は、ユーザーインポートジョブの作成時に返されたジョブ ID です。

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id
 "JOB_ID"
```

Example レスポンス例:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn":"ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

前述の例の出力では、*PRE_SIGNED_URL* は CSV のアップロード先の URL です。*ROLE_ARN* は、ロールの作成時に受け取った CloudWatch Logs ロール ARN です。

ユーザーインポートジョブを表示する

ユーザーインポートジョブを一覧表示するには、次のコマンドを使用します:

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example レスポンス例:

```
{
    "UserImportJobs": [
        {
            "Status": "Created",
            "SkippedUsers": 0,
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
```

```
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the
 import.",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
}
```

ジョブは時系列で、作成が最後から最初の順で表示されます。2 番目のジョブのあとの *PAGINATION_TOKEN* 文字列は、このリストコマンドについて、まだ結果が残っていることを示します。残りの結果を表示するには、次のように --pagination-token オプションを使用します:

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

ユーザーインポートジョブの開始

ユーザーインポートジョブを開始するには、次のコマンドを使用します:

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

インポートジョブは、アカウントごとに 1 つしかアクティブにできません。

Example レスポンス例:

```
{
```

```
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

ユーザーインポートジョブの停止

ユーザーインポートジョブの進行中に停止するには、次のコマンドを使用します。ジョブを停止すると、再開することはできません。

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example レスポンス例:

```
{
    "UserImportJob": {
        "CompletionDate": 1470958050.571,
        "StartDate": 1470958047.797,
        "Status": "Stopped",
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "CompletionMessage": "The Import Job was stopped by the developer.",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957972.387
    }
}
```

ユーザープールのインポート結果を CloudWatch コンソールに表示

インポートジョブの結果は、Amazon CloudWatch コンソールで表示できます。

トピック

結果の表示

次のステップでは、ユーザープールのインポートの結果を表示する方法を説明します。

ユーザープールのインポート結果を見るには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. [ログ] を選択します。
3. ユーザープールのインポートジョブに使用するロググループを選択します。ロググループ名は `/aws/cognito/userpools/`*`USER_POOL_ID`*`/`*`USER_POOL_NAME`* の形式です。
4. 先ほど実行したユーザーインポートジョブのログを選択します。ログ名は *`JOB_ID`*`/`*`JOB_NAME`* の形式です。ログの結果は行番号でユーザーを示しています。ユーザーデータはログに書き込まれていません。各ユーザーに対して、次のような行が表示されます。

   - `[SUCCEEDED] Line Number 5956 - The import succeeded.`
   - `[SKIPPED] Line Number 5956 - The user already exists.`
   - `[FAILED] Line Number 5956 - The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).`

結果の解釈

正常にインポートしたユーザーは、ステータスが「PasswordReset」に設定済みです。

次の場合、ユーザーはインポートされませんが、インポートジョブは継続します。

- `true` に設定された自動検証された属性はありません。
- ユーザーデータはスキーマに一致しません。
- 内部エラーによりユーザーをインポートできませんでした。

以下の場合、インポートジョブは失敗します。

- Amazon CloudWatch Logs ロールを想定することができず、正しいアクセスポリシーがない、または削除されている。
- ユーザープールが削除されている。
- Amazon Cognito が .csv ファイルを解析できない。

インポートされたユーザーにパスワードをリセットするように要求

インポートされた各ユーザーが初めてサインインしてパスワードを入力するときは、新しいパスワードを入力する必要があります。以下の手順では、CSV ファイルをインポートした後のローカルユーザーによるカスタム App でのユーザーエクスペリエンスについて説明します。ユーザーがホストされた UI でサインインすると、Amazon Cognito はユーザーが最初にサインインしたときに新しいパスワードを設定するように求めます。

インポートされたユーザーにパスワードをリセットするように要求

1. アプリケーションで、ランダムなパスワードを使用した `InitiateAuth` により、現在のユーザーのサインインをサイレントに試行します。
2. `PreventUserExistenceErrors` が有効な場合、Amazon Cognito は `NotAuthorizedException` を返します。そうでない場合は `PasswordResetRequiredException` を返します。
3. アプリケーションが `ForgotPassword` API リクエストを行い、ユーザーのパスワードをリセットします。

    a. アプリケーションは `ForgotPassword` API リクエストでユーザー名を送信します。

    b. Amazon Cognito は、確認済みの E メールまたは電話番号にコードを送信します。宛先は、CSV ファイルで `email_verified` と `phone_number_verified` に指定した値によって異なります。`ForgotPassword` リクエストへのレスポンスは、コードの宛先を示します。

> **Note**
>
> ユーザープールは、E メールまたは電話番号を確認するように設定する必要があります。詳細については、「ユーザーアカウントのサインアップと確認」を参照してください。

c. アプリケーションは、コードがコードの送信先を確認するメッセージをユーザーに表示し、ユーザーにコードと新しいパスワードを入力するように求めます。

d. ユーザーがアプリでコードと新しいパスワードを入力します。

e. アプリケーションは `ConfirmForgotPassword` API リクエストでコードと新しいパスワードを送信します。

f. アプリケーションはユーザーをサインインにリダイレクトします。

ユーザープール属性

属性は、名前、E メールアドレス、電話番号など、個々のユーザーを識別できる各種情報です。新しいユーザープールにはデフォルトの標準属性のセットがあります。のユーザープール定義にカスタム属性を追加することもできます AWS Management Console。このトピックでは、それらの属性について詳しく説明し、ユーザープールの設定のヒントを示します。

ユーザーに関するすべての情報を属性に保存しないでください。例えば、使用統計やゲームスコアなどの頻繁に変化するユーザーデータは、Amazon Cognito Sync または Amazon DynamoDB などの個別のデータストアに保存してください。

> **Note**
>
> ドキュメントや標準によっては、属性をメンバーと呼んでいます。

トピック

標準属性

Amazon Cognito は、[OpenID Connect の仕様] に基づいて、以下の標準属性セットをすべてのユーザーに割り当てます。標準属性値およびカスタム属性値は、デフォルトでは最大 2048 文字の任意の文字列ですが、一部の属性値には形式の制限があります。

標準属性は以下のとおりです。

- address
- birthdate
- email
- family_name
- gender
- given_name
- locale
- middle_name
- name
- nickname
- phone_number
- picture
- preferred_username
- profile
- sub
- updated_at
- website
- zoneinfo

sub を除き、標準属性はすべてのユーザーに対してデフォルトでオプションとなります。属性を必須にする場合は、ユーザープールの作成プロセスで、属性の横にある [必須] チェックボックスをオンにします。Amazon Cognito は、各ユーザーの sub 属性に一意のユーザー識別子の値を割り当てます。検証できるのは E メールと phone_number 属性だけです。

> **Note**
>
> 標準属性を必須とマークすると、ユーザーは、属性の値が指定されない限り登録することはできません。ユーザーを作成し、必要な属性の値を指定しないようにするには、管理者は AdminCreateUser API を使用できます。ユーザープールを作成した後で、属性を必須と必須ではないの間で切り替えることができません。

標準属性の詳細と形式の制限

birthdate

値は、YYYY-MM-DD 形式で有効な 10 文字の日付にする必要があります。

email

ユーザーおよび管理者は、E メールアドレスの値を確認できます。

適切な AWS アカウント アクセス許可を持つ管理者は、ユーザーの E メールアドレスを変更し、それを検証済みとしてマークすることもできます。AdminUpdateUserAttributes API または admin-update-user-attributes AWS Command Line Interface （ AWS CLI) コマンドを使用して、E メールアドレスを検証済みとしてマークします。このコマンドを使用すると、管理者はemail_verified 属性を true に変更できます。のユーザータブでユーザーを編集 AWS Management Console して、E メールアドレスを検証済みとしてマークすることもできます。

値は、@ 記号とドメインを含む標準の電子メール形式に従った有効な E メールアドレス文字列で、長さは 2048 文字までです。

phone_number

SMS 多要素認証 (MFA) がアクティブである場合、ユーザーは電話番号を入力する必要があります。詳細については、「ユーザープールに MFA を追加します」を参照してください。

ユーザーおよび管理者は、電話番号の値を確認できます。

適切な AWS アカウント アクセス許可を持つ管理者は、ユーザーの電話番号を変更し、検証済みとしてマークすることもできます。AdminUpdateUserAttributes API または admin-update-user-attributes AWS CLI コマンドを使用して、電話番号を検証済みとしてマークします。このコマンドを使用すると、管理者はphone_number_verified 属性を true に変更できます。のユーザータブでユーザーを編集 AWS Management Console して、電話番号を検証済みとしてマークすることもできます。

> **Important**
>
> 電話番号は次の形式に従う必要があります: 電話番号は、プラス記号 (+) から始めて国番号がその後に続く必要があります。電話番号には、プラス記号 (+) および数値のみを含めることができます。値をサービスに送信する前に、電話番号から括弧、スペース、ハイフン (-) などの他の文字を削除してください。例えば、米国の電話番号は **+14325551212** の形式に従っています。

preferred_username

preferred_username は、必要に応じて、またはエイリアスとして選択できますが、両方を選択することはできません。preferred_username がエイリアスである場合は、UpdateUserAttributes API オペレーションにリクエストを行い、ユーザーを確認した後に属性値を追加できます。

sub

sub 属性に基づいてユーザーをインデックス化して検索します。sub 属性は、各ユーザープール内で一意のユーザー識別子です。ユーザーは phone_number や email などの属性を変更できます。sub 属性には固定値が含まれています。ユーザーの検索の詳細については、「ユーザーアカウントの管理と検索」を参照してください。

必須属性を表示する

以下の手順に従って、特定のユーザープールに必要な属性を表示します。

> **Note**
> ユーザープールを作成した後は、必須の属性を変更することはできません。

必須属性を表示する

1. の Amazon Cognito に移動します AWS Management Console。コンソールでプロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Sign-up experience] (サインアップエクスペリエンス) タブを選択します。
5. [Required attributes] (必須の属性) セクションで、ユーザープールの必須属性が表示されます。

ユーザー名および優先ユーザー名

username 値は個別の属性であり、name 属性とは異なります。各ユーザーには、username 属性があります。Amazon Cognito は、フェデレーティッドユーザーのユーザー名を自動的に生成します。Amazon Cognito ディレクトリにローカルユーザーを作成するには、username 属性を指定する必要があります。ユーザーを作成した後で、username 属性の値を変更することはできません。

デベロッパーは、preferred_username 属性を使用して、変更可能なユーザー名をユーザーに付与できます。詳細については、「ログイン属性のカスタマイズ」を参照してください。

お客様のアプリケーションでユーザー名が不要の場合は、ユーザーにユーザー名を指定するように求める必要はありません。アプリでユーザー用の一意のユーザー名をバックグラウンドで作成できます。例えば、E メールアドレスとパスワードを登録しそれを使用してサインインするようにユーザーに求める場合は便利です。詳細については、「ログイン属性のカスタマイズ」を参照してください。

username はユーザープール内で一意である必要があります。username は再利用できますが、削除されて使用されなくなった場合のみです。username 属性の文字列制約の詳細については、SignUp「API リクエストのユーザー名プロパティ」を参照してください。

ログイン属性のカスタマイズ

ユーザープールを作成するときに、ユーザーが e メールアドレスまたは電話番号をユーザー名として使用してサインアップおよびサインインできるようにする場合は、ユーザー名属性を設定できます。また、エイリアス属性を設定してユーザーに選択肢を与えることもできます。サインアップするときに複数の属性を含めて、ユーザー名、優先ユーザー名、E メールアドレス、または電話番号でサインインできます。

> **Important**
> ユーザープールを作成した後は、この設定を変更することはできません。

エイリアス属性とユーザー名属性の選択方法

要件	エイリアス属性	ユーザー名属性
ユーザーには複数のサインイン属性があります	Yes[1]	No[2]
ユーザーは、サインインする前に E メールアドレスまたは電話番号を確認する必要があります	はい	いいえ
重複する E メールアドレスまたは電話番号でユーザーをサインアップし、UsernameE	はい	いいえ

要件	エイリアス属性	ユーザー名属性
xistsException エラーを防ぐ[3]		
複数のユーザーに同じ E メールアドレスまたは電話番号属性値を割り当てることができます	あり[4]	いいえ

[1] 使用可能なサインイン属性は、ユーザー名、E メールアドレス、電話番号、および優先ユーザー名です。

[2] E メールアドレスまたは電話番号を使用してサインインできます。

[3] ユーザーが登録したメールアドレスや電話番号が重複している可能性があるが、ユーザー名がない場合、ユーザープールで UsernameExistsException エラーが生成されません。この動作はユーザー名存在エラーを防ぐとは無関係です。これはサインイン操作には適用されますが、サインアップ操作には適用されません。

[4] 属性を確認した最後のユーザーのみが、その属性を使用してサインインできます。

オプション 1: 複数のログイン属性 (エイリアス属性)

ユーザーがサインイン時にユーザー名やその他の属性値を入力することを選択できるようにするには、エイリアスを有効にすることができます。デフォルトでは、ユーザーは各自のユーザー名とパスワードでサインインします。ユーザー名はユーザーが変更できない固定値です。属性をエイリアスとしてマーキングすると、ユーザーはユーザー名の代わりにその属性を使用してサインインできます。E メールアドレス、電話番号、および任意ユーザー名属性は、エイリアスレコードとしてマーキングできます。例えば、E メールと電話をユーザープールのエイリアスとして選択すると、そのユーザープールのユーザーは、自分のユーザー名、E メールアドレス、または電話番号とパスワードを使ってサインインできます。

エイリアス属性を選択するには、ユーザープールを作成するときに [User name] (ユーザー名) と 1 つ以上の追加のサインインオプションを選択します。

> **Note**
>
> 大文字と小文字を区別しないようにユーザープールを設定すると、ユーザーは小文字または大文字を使用してサインアップ、またはエイリアスを使用してサインインできるようにな

ります。詳細については、CreateUserPoolAmazon Cognito ユーザープール API リファレンス」の「」を参照してください。

エイリアスとして E メールアドレスを選択した場合、Amazon Cognito は有効な E メールアドレス形式と一致するユーザー名を受け入れません。同様に、電話番号をエイリアスとして選択した場合、Amazon Cognito は、有効な電話番号形式に一致するユーザープールのユーザー名を受け入れません。

> **Note**
>
> エイリアス値は、ユーザープール内で一意にする必要があります。エイリアスがメールアドレスまたは電話番号に設定される場合、指定される値は 1 つのアカウントのみで検証状態になります。サインアップ時に E メールアドレスまたは電話番号をエイリアス値として指定し、別のユーザーがその エイリアス値を既に使用している場合、登録は成功します。ただし、ユーザーがこの E メール (または電話番号) を持つアカウントを確認して有効なコードを入力しようとする場合、Amazon Cognito は、`AliasExistsException` エラーを返します。エラーは、この E メールアドレス (または電話番号) を持つアカウントが既に存在していることを示します。この時点で、ユーザーは新しいアカウントの作成を中止して、古いアカウントのパスワードのリセットを試すことができます。ユーザーが新しいアカウントの作成を続行する場合は、アプリは `forceAliasCreation` オプションを使用して `ConfirmSignUp` API を呼び出す必要があります。`forceAliasCreation` を使用する `ConfirmSignUp` は、エイリアスを前のアカウントから新しく作成されたアカウントに移動し、前のアカウントで未確認の属性をマークします。

ユーザーが電話番号および E メールアドレスを確認した後、電話番号および E メールアドレスのみがユーザーのアクティブエイリアスになります。エイリアスとしてこれらを使用する場合は、E メールアドレスと電話番号の自動確認を選択することをお勧めします。

ユーザーがサインアップするときに、E メールアドレスおよび電話番号の属性の `UsernameExistsException` エラーを防ぐために、エイリアス属性を選択します。

`preferred_username` 属性をアクティブにして、`username` 属性値が変更されていないときに、ユーザーがサインインに使用するユーザー名を変更できるようにします。このユーザーエクスペリエンスをセットアップする場合は、新しい `username` 値を `preferred_username` として送信し、エイリアスとして `preferred_username` を選択します。こうすることで、ユーザーは入力した新しい値でサインインできます。`preferred_username` がエイリアスとして選択された場合、ユー

ザーはアカウントが確認されている場合にのみ、値を指定できます。登録時にはこの値は指定できません。

ユーザーがユーザー名を使用してサインアップする場合、ユーザーが次の 1 つ以上のエイリアスを使用してサインインできるかどうかを選択できます。

- 検証済みの E メールアドレス
- 検証済みの電話番号
- 優先ユーザー名

これらのエイリアスはユーザーがサインアップした後に変更できます。

> **Important**
>
> ユーザープールがエイリアスによるサインインをサポートしている場合、ユーザーを認証または検索するときに、サインイン属性のいずれかを使用してユーザーを特定することは避けてください。固定値のユーザー識別子 sub は、ユーザーのアイデンティティを示す唯一の一貫した指標です。

エイリアスを使用してサインインできるように、ユーザープールを作成する場合は、次の手順を含めます。

ユーザーが優先ユーザー名でサインインできるようにユーザープールを設定するには

1. AWS Management Consoleの Amazon Cognito に移動します。コンソールでプロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。
4. [Configure sign-in experience] (サインインエクスペリエンスを構成する) で、ユーザープールに関連付けるアイデンティティの [Provider types] (プロバイダーのタイプ) を選択します。
5. [Cognito user pool sign-in options] (Cognito ユーザープールのサインインオプション) で、[User name] (ユーザー名)、[Email] (E メール)、および [Phone number] (電話番号) のお好きな組み合わせを選択します。
6. [ユーザー名の要件] で、[優先ユーザー名を使用したサインインを許可する] を選択して、ユーザーがサインイン時に代替ユーザー名を設定できるようにします。

7. [Next] (次へ) をクリックし、ウィザードのすべての手順を完了します。

オプション 2: サインイン属性 (ユーザー名属性) としての E メールアドレスまたは電話番号

ユーザー名として E メールアドレスまたは電話番号でユーザーがサインアップする場合、E メールアドレスのみ、電話番号のみ、またはいずれか一方でもサインアップできるようにするかを選択することができます。

ユーザー名属性を選択するには、ユーザープールを作成するときに、サインインオプションとして [ユーザー名] を選択しないでください。

E メールまたは電話番号は一意である必要があり、別のユーザーによって既に使用されていてはなりません。これを検証する必要はありません。ユーザーが E メールアドレスまたは電話番号を使用してサインアップした後、ユーザーは同じ E メールアドレスまたは電話番号を使用して新しいアカウントを作成することはできません。ユーザーは既存のアカウントを (必要に応じてパスワードをリセットして) 再利用できるだけです。ただし、ユーザーは E メールアドレスまたは電話番号を新しい E メールアドレスまたは電話番号に変更できます。E メールアドレスまたは電話番号が既に使用中ではない場合は、それが新しいユーザー名となります。

> **Note**
>
> ユーザーが E メールアドレスをユーザー名として使用してサインアップした場合、ユーザー名を別の E メールアドレスに変更できますが、電話番号に変更することはできません。電話番号でサインアップする場合は、ユーザー名を別の電話番号を変更できますが、E メールアドレスに変更することはできません。

次のステップを使用して、ユーザープール作成のプロセス中に、E メールアドレスまたは電話番号を使用したサインアップおよびサインインをセットアップします。

E メールアドレスまたは電話番号を使用したサインアップとサインインをユーザープールで設定するには

1. AWS Management Consoleの Amazon Cognito に移動します。コンソールでプロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. ページの右上隅にある [Create a user pool] (ユーザープールを作成する) を選択して、ユーザープール作成のウィザードを開始します。

4. [Cognito user pool sign-in options] (Cognito ユーザープールのサインインオプション) で、ユーザーがサインインに使用することを許可する属性を表す [Email] (E メール)、および [Phone number] (電話番号) から任意の組み合わせを選択します。
5. [Next] (次へ) をクリックし、ウィザードの残りのステップを完了します。

> **Note**
>
> E メールアドレスまたは電話番号をユーザープールの必須属性としてマークする必要はありません。

アプリケーションでオプション 2 を実装する

1. `CreateUserPool` API を呼び出してユーザープールを作成します。`UserNameAttributes` パラメータを `phone_number`、`email`、または `phone_number | email` に設定します。
2. `SignUp` API を呼び出して、E メールアドレスまたは電話番号を API の `username` パラメータに渡します。この API によって次の処理が実行されます。
   - [username] 文字列が有効な E メールアドレス形式の場合、ユーザープールは自動的にユーザーの `email` 属性に `username` 値を入力します。
   - [username] 文字列が有効な電話番号形式の場合、ユーザープールは自動的にユーザーの `phone_number` 属性に `username` 値を入力します。
   - `username` 文字列形式が E メールアドレスまたは電話番号の形式ではない場合は、`SignUp` API から例外が返されます。
   - `SignUp` API によってユーザーの永続的な UUID が生成され、内部でイミュータブルなユーザー名属性として使用されます。この UUID は、ユーザー ID トークン内の `sub` クレームと同じ値です。
   - `username` 文字列に含まれている E メールアドレスまたは電話番号が既に使用されている場合、`SignUp` API によって例外が返されます。

`ListUsers` API を除くすべての API で、E メールアドレスまたは電話番号をエイリアスとしてユーザー名の代わりに使用できます。`ListUsers` を呼び出すと、`email` または `phone_number` 属性で検索できます。`username` で検索する場合は、エイリアスではなく、実際のユーザー名を指定する必要があります。

カスタム属性

ユーザープールには、カスタム属性を最大 50 まで追加できます。カスタム属性の最小長または最大長を指定できます。ただし、すべてのカスタム属性の最大長は 2048 文字以下にする必要があります。

各カスタム属性には以下の特徴があります。

- 文字列または数値として定義できます。Amazon Cognito は ID トークンにカスタム属性値を文字列としてのみ書き込みます。
- ユーザーが属性の値を提供することを要求することはできません。
- ユーザープールに追加した後で、削除または変更することはできません。
- 属性名の文字長は、Amazon Cognito が受け入れる制限内です。詳細については、「Amazon Cognito のクォータ」を参照してください。
- 変更可能またはイミュータブルとすることができます。ユーザーを作成するときに、イミュータブル属性に値を書き込むことができます。アプリクライアントがその属性に対する書き込み権限を持っている場合、変更可能な属性の値を変更できます。詳細については、「属性の権限と範囲」を参照してください。

> **Note**
>
> コードおよび ロールベースアクセスコントロールの使用 のルールの設定で、カスタム属性は標準属性と区別するために、`custom:` プレフィックスを必要とします。

ユーザープールを作成するときに、 の `SchemaAttributes`プロパティにデベロッパー属性を追加することもできますCreateUserPool。開発者属性には `dev:` プレフィックスがあります。ユーザーのデベロッパー属性は、 AWS 認証情報でのみ変更できます。開発者属性はレガシー機能であり、Amazon Cognito がアプリケーションクライアントの読み取り/書き込み権限と置き換えました。

以下の手順を使用して、新しくカスタム属性を作成します。

コンソールを使用してカスタム属性を追加する

1. の Amazon Cognito に移動します AWS Management Console。コンソールでプロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。

3. リストから存在するユーザープールを 1 つ選択します。
4. [Sign-up experience] (サインアップエクスペリエンス) タブをクリックして、[Custom attributes] (カスタム属性) セクションで、[Add custom attributes] (カスタム属性を追加する) を選択します。
5. [Add custom attributes] (カスタム属性を追加する) のページで、新しい属性に関する以下の詳細情報を入力します。
   - 名前を入力します。
   - [Type] (タイプ) (文字列または数値) を選択します。
   - 最小文字列の長さまたは数値を入力します。
   - 最大文字列の長さまたは数値を入力します。
   - ユーザーが初期値を設定した後で、カスタム属性の値を変更する権限をユーザーに付与する場合は、[Mutable] (変更可能) を選択します。
6. [変更を保存] を選択します。

属性の権限と範囲

各アプリクライアントで、各ユーザー属性に読み取りと書き込み権限を設定できます。これにより、ユーザー用に保存する各属性の読み取りおよび変更に必要なアプリによるアクセスを制御できます。例えば、ユーザーが有料サービスを利用しているお客様であるかどうかを示すカスタム属性があるとします。アプリでこの属性は表示できますが、直接変更することはできません。代わりに、管理ツールまたはバックグラウンドプロセスを使用してこの属性を更新します。ユーザー属性の権限は、Amazon Cognito コンソール、Amazon Cognito API、または AWS CLIから設定できます。デフォルトでは、新しいカスタム属性を利用する前に、読み取り/書き込みの許可を設定する必要があります。デフォルトでは、新しいアプリケーションクライアントを作成すると、すべての標準属性とカスタム属性に対する読み取りおよび書き込みアクセス許可がアプリケーションに付与されます。アプリケーションが必要とする情報量のみに制限するには、アプリケーションクライアント設定の属性に特定のアクセス権限を割り当てます。

ベストプラクティスとして、アプリケーションクライアントを作成するときに属性の読み取りおよび書き込みアクセス許可を指定します。アプリケーションのオペレーションに必要な最小限のユーザー属性セットへのアクセス権をアプリケーションクライアントに付与します。

> **Note**
>
> DescribeUserPoolClient は、デフォルト以外のアプリケーションクライアントのアクセス許可を設定するWriteAttributes場合にのみ、 ReadAttributesと の値を返します。

属性権限を更新するには (AWS Management Console)

1. の Amazon Cognito に移動します AWS Management Console。コンソールでプロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存するユーザープールを 1 つ選択します。
4. [App integration] (アプリケーションの統合) タブ、[App clients] (アプリケーションクライアント) セクションで、リストからアプリケーションクライアントを選択します。
5. [Attribute read and write permissions] (属性の読み込みおよび書き込みアクセス許可) セクションで、[Edit] (編集) を選択します。
6. [Edit attribute read and write permissions] (属性の読み込みおよび書き込みアクセス権限を編集) ページで、読み取りおよび書き込み権限を設定し、[Save changes] (変更の保存) を選択します。

カスタム属性を使用して、アプリクライアントごとに以上の手順を繰り返します。

アプリごとに、属性を読み取り可能または書き込み可能とマークできます。これは、標準属性とカスタム属性の両方に当てはまります。アプリは、読み取り可能としてマークした属性の値を取得したり、書き込み可能としてマークした属性の値を設定または変更したりできます。書き込み権限のない属性の値をアプリが設定しようとすると、Amazon Cognito が NotAuthorizedException を返します。GetUser リクエストには、アプリケーションクライアントクレームを含むアクセストークンが含まれます。Amazon Cognito は、アプリケーションクライアントが読み取ることができる属性の値のみを返します。アプリからのユーザーの ID トークンには、読み取り可能な属性に対応するクレームのみが含まれます。すべてのアプリケーションクライアントは、ユーザープールの必須属性を書き込むことができます。Amazon Cognito ユーザープール API リクエストで属性の値を設定できるのは、まだ値が設定されていない必須属性の値も指定する場合のみに限られます。

カスタム属性の読み取り書き込みアクセス許可にはそれぞれ異なる機能があります。ユーザープールの変更可能な属性または変更不可能な属性として作成でき、どのアプリクライアントでも読み取りまたは書き込み属性として設定できます。

変更不可能なカスタム属性は、ユーザー作成時に 1 回だけ更新できます。変更不可能な属性には以下のメソッドを入力できます。

- SignUp: ユーザーは、変更不可能なカスタム属性への書き込みアクセス許可を持つアプリクライアントにサインアップします。その属性の値を提供します。
- サードパーティ IdP でのサインイン: ユーザーは、変更不可能なカスタム属性への書き込みアクセス許可を持つアプリクライアントにサインインします。IdP のユーザープール設定には、提供されたクレームを変更不可能な属性にマップするルールがあります。
- AdminCreateUser: 変更不可能な属性の値を指定します。

アプリケーションクライアントに割り当てることができるスコープについては、「スコープ、M2M 、およびリソースサーバーによる API 認証」を参照してください。

ユーザープールを作成した後、属性の権限と範囲を変更できます。

ユーザープールのパスワードの追加要件

強力で複雑なパスワードは、ユーザープールのセキュリティのベストプラクティスです。特にインターネットに公開されているアプリケーションでは、弱いパスワードにより、ユーザーの認証情報がパスワードを推測してデータにアクセスしようとするシステムに公開される可能性があります。パスワードが複雑なほど、推測が困難になります。Amazon Cognito には、アドバンストセキュリティ機能やAWS WAFウェブ ACLs など、セキュリティを重視する管理者向けの追加のツールがありますが、パスワードポリシーはユーザーディレクトリのセキュリティの中心的な要素です。

Amazon Cognito ユーザープールのローカルユーザーのパスワードは、自動的に期限切れになることはありません。ベストプラクティスとして、ユーザーパスワードのリセットの時刻、日付、メタデータを外部システムに記録します。パスワードの有効期間の外部ログを使用すると、アプリケーションまたは Lambda トリガーはユーザーのパスワードの有効期間を検索し、特定の期間後にリセットを要求できます。

セキュリティ標準に準拠したパスワードの複雑さを最小限に抑えるようにユーザープールを設定できます。複雑なパスワードには、8 文字以上の長さが必要です。また、大文字、数字、特殊文字が混在しています。

ユーザープールのパスワードポリシーを設定するには

1. ユーザープールを作成して [セキュリティ要件を設定] ステップに移動するか、既存のユーザープールにアクセスして [サインインエクスペリエンス] タブに移動します。
2. [パスワードポリシー] に移動します。

3. [パスワードポリシーモード] を選択します。[Cognito のデフォルト] では、推奨最小設定でユーザープールを設定します。[カスタム] パスワードポリシーを選択することもできます。
4. [パスワードの最小文字数] を設定します。すべてのユーザーは、この値以上の長さのパスワードでサインアップまたは作成する必要があります。この最小値は 99 まで設定できますが、ユーザーは最大 256 文字のパスワードを設定できます。
5. [パスワードの要件] でパスワードの複雑さのルールを設定します。各ユーザーのパスワードで少なくとも 1 つは必須とする文字タイプ (数字、特殊文字、大文字、小文字) を選択します。

   パスワードには、次の文字のうち少なくとも 1 つを要求できます。Amazon Cognito がパスワードに最低限必要な文字が含まれていることを確認した後、ユーザーのパスワードには、パスワードの最大長までの任意のタイプの追加の文字を含めることができます。

   - 大文字と小文字の基本的なラテン文字
   - 数字
   - 以下の特殊文字。

     ```
     ^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
     ```

   - 先頭でも末尾にもない空白文字。
6. [管理者が設定した一時パスワードの有効期限] に値を設定します。この期間が過ぎると、Amazon Cognito コンソールまたは `AdminCreateUser` で作成した新しいユーザーは、サインインして新しいパスワードを設定できなくなります。一時パスワードでサインインすると、ユーザーアカウントが期限切れになることはありません。Amazon Cognito ユーザープール API のパスワード期間を更新するには、 CreateUserPoolまたは UpdateUserPool API リクエストTemporaryPasswordValidityDays で の値を設定します。

   - 期限切れのユーザーアカウントのアクセス権をリセットするには、以下のいずれかを実行します。

     - ユーザープロファイルを削除し、新しいものを作成します。
     - AdminSetUserPassword API リクエストで新しい永続パスワードを設定します。
     - AdminResetUserPassword API リクエストで新しい確認コードを生成します。

Amazon Cognito ユーザープールの E メール設定

ユーザープールのクライアントアプリでの特定のイベントにより、Amazon Cognito がユーザーに E メールを送信する場合があります。例えば、E メールの検証を必須とするようにユーザープールを設

定している場合、ユーザーがアプリの新しいアカウントにサインアップする、またはパスワードをリセットする場合に Amazon Cognito が E メールを送信します。E メールを開始するアクションに応じて、検証コードまたは仮パスワードが含まれます。

E メール配信を処理するには、次のいずれかのオプションを使用できます。

- Amazon Cognito サービスに組み込まれているデフォルトの E メール設定。
- Amazon Simple Email Service (Amazon SES) の設定。

ユーザープールを作成した後に、配信オプションを変更できます。

Amazon Cognito は、ユーザーが入力できるコードか選択可能な URL リンクのいずれかを含む E メールメッセージをユーザーに送信します。次の表は、E メールメッセージを生成できるイベントを示しています。

メッセージオプション

アクティビティ	API オペレーション	配信オプション	形式オプション	カスタマイズ可能	メッセージテンプレート
パスワードを忘れてしまった	ForgotPassword	E メール、SMS	コード	いいえ	該当なし
招待	AdminCreateUser	E メール、SMS	コード	はい	招待メッセージ
自己登録	SignUp	E メール、SMS	コード、リンク	はい	検証メッセージ
E メールアドレスまたは電話番号の検証	UpdateUserAttributes	E メール、SMS	コード	はい	検証メッセージ
多要素認証 (MFA)	AdminInitiateAuth, InitiateAuth	SMS	コード	Yes[1]	MFA メッセージ

[1] SMS メッセージ用。

Amazon SES はメールメッセージに対して料金を請求します。詳細については、「Amazon SES の料金 」を参照してください。

デフォルトの E メール設定

Amazon Cognito は、デフォルトの E メール設定を使用して E メール配信を処理できます。デフォルトのオプションを使用すると、Amazon Cognito は、ユーザープールに毎日送信するメールの数を制限します。サービスの制限に関する詳細については、「Amazon Cognito のクォータ」を参照してください。一般的な実稼働環境では、デフォルトの E メール制限は必要な配信ボリュームを下回っています。大規模なデリバリーボリュームを有効にするには、Amazon SES E メール設定を使用する必要があります。

デフォルト設定を使用する場合は、 によって AWS 管理される Amazon SES リソースを使用して E メールメッセージを送信します。Amazon SES は、ハードバウンスを返す E メールアドレスをアカウントレベルのサプレッションリストまたはグローバルサプレッションリストに追加します。配信不能な E メールアドレスが後で配信可能になった場合、ユーザープールがデフォルト設定を使用するように設定されている場合、サプレッションリストからの削除を制御できません。E メールアドレスは、 AWSマネージドサプレッションリストに無期限に残すことができます。配信不能な E メールアドレスを管理するには、次のセクションで説明するように、アカウントレベルのサプレッションリストで Amazon SES E メール設定を使用します。

デフォルトの E メール設定を使用すると、オプションでは、次のいずれかの E メールアドレスを FROM アドレスとして使用できます。

- デフォルトの E メールアドレス (no-reply@verificationemail.com)。
- カスタム E メールアドレス。独自の E メールアドレスを使用する前に、それを Amazon SES で検証し、このアドレスを使用するための許可を Amazon Cognito に付与する必要があります。

Amazon SES の E メール設定

アプリケーションは、デフォルトのオプションで利用できるよりも大きいボリュームを必要とする可能性があります。可能なデリバリーボリュームを増やすには、Amazon SES リソースとユーザープールを使用してユーザーに E メールを送信します。独自の Amazon SES 設定で E メールメッセージを送信するときに、E メール送信アクティビティをモニタリングすることもできます。

Amazon SES の設定を使用する前に、1 つ、または複数の E メールアドレスまたはドメインを Amazon SES で検証する必要があります。検証済み E メールアドレスまたは検証済みドメインの

アドレスを、ユーザープールに割り当てる FROM E メールアドレスとして使用します。Amazon Cognito が E メールをユーザーに送信するときは、Amazon SES を呼び出し、その E メールアドレスを使用するようになります。

Amazon SES の設定を使用する場合は、次の条件が適用されます。

- ユーザープールの E メール配信制限が、 AWS アカウントの Amazon SES 検証済み E メールアドレスに適用される制限と同じになります。
- Amazon SES のアカウントレベルのサプレッションリストで、グローバルサプレッションリストを上書きして、配信不能な E メールアドレスへのメッセージを管理することができます。アカウントレベルのサプレッションリストを使用すると、E メールメッセージのバウンスが送信者としてのアカウントの評価に影響します。詳細については、Amazon Simple Email Service デベロッパーガイドの「Amazon SES アカウントレベルのサプレッションリストの使用」を参照してください。

Amazon SES の E メール設定リージョン

ユーザープール AWS リージョン を作成する には、Amazon SES での E メールメッセージの設定に関する 3 つの要件のいずれかがあります。ユーザープールと同じリージョン、同じリージョンを含む複数のリージョン、または 1 つ以上のリモートリージョンで、Amazon SES から E メールメッセージを送信できます。最適なパフォーマンスを得るには、 オプションがある場合は、ユーザープールと同じリージョンで Amazon SES 検証済み ID を使用して E メールメッセージを送信します。

Amazon SES 検証済み ID のリージョン要件のカテゴリ

リージョン内のみ

ユーザープールは、ユーザープール AWS リージョン と同じ で検証済み ID を含む E メールメッセージを送信できます。カスタム E メールアドレスを使用しないデフォルトの FROM E メール設定では、Amazon Cognito は同じリージョンで`no-reply@verificationemail.com`検証済み ID を使用します。

下位互換性

ユーザープールは、同じリージョン AWS リージョン または次のいずれかの代替リージョンで、検証済み ID を含む E メールメッセージを送信できます。

- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)

- 欧州 (アイルランド)

この機能は、サービスの起動時に Amazon Cognito の要件に合わせて作成したユーザープールリソースの継続性をサポートします。その期間のユーザープールは、限られた数の で検証済み ID の E メールメッセージのみを送信できました AWS リージョン。カスタム E メールアドレスを使用しないデフォルトの FROM E メール設定では、Amazon Cognito は同じリージョンでno-reply@verificationemail.com検証済み ID を使用します。

代替リージョン

ユーザープールは、ユーザープールリージョン外の代替 で、検証 AWS リージョン 済み ID を含む E メールメッセージを送信できます。この設定は、Amazon Cognito が利用可能なリージョンで Amazon SES が利用できない場合に発生します。 Amazon Cognito

代替リージョンで検証済み ID の Amazon SES 送信承認ポリシーは、発信元リージョンの Amazon Cognito サービスプリンシパルを信頼する必要があります。詳細については、「デフォルトの E メール設定を使用するアクセス許可を付与するには」を参照してください。

これらのリージョンの一部では、Amazon Cognito は のデフォルトの E メール設定のために、E メールメッセージを 2 つの代替リージョンに分割しますCOGNITO_DEFAULT。このような場合、カスタム FROM E メールアドレスを使用するには、各代替リージョンで検証済み ID の Amazon SES 送信承認ポリシーが、発信元のリージョンの Amazon Cognito サービスプリンシパルを信頼する必要があります。詳細については、「デフォルトの E メール設定を使用するアクセス許可を付与するには」を参照してください。これらのリージョンDEVELOPERの の Amazon SES E メール設定では、最初にリストされたリージョンで検証済み ID を使用し、ユーザープールリージョンの Amazon Cognito サービスプリンシパルを信頼するように設定する必要があります。例えば、中東 (アラブ首長国連邦) のユーザープールで、 を信頼するように欧州 (フランクフルト) で検証済み ID を設定しますcognito-idp.me-central-1.amazonaws.com。カスタム E メールアドレスを使用しないデフォルトの FROM E メール設定では、Amazon Cognito は各リージョンでno-reply@verificationemail.com検証済み ID を使用します。

> **Note**
>
> 以下の条件の組み合わせでは、リージョン要素でワイルドカードEmailConfigurationを使用して の SourceArnパラメータを 形式で指定する必要がありますarn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。これにより、ユーザープールは、両方の の で同じ検証済み ID AWS アカウント の E メールメッセージを送信できます AWS リージョン。

- は EmailSendingAccount ですCOGNITO_DEFAULT。
- カスタムFROMアドレスを使用します。
- ユーザープールは代替リージョン で E メールを送信します。
- ユーザープールには、次の Amazon SES がサポートするリージョンの表で指定されている 2 番目の[1]代替リージョンがあります。 Amazon SES

AWS SDK、Amazon Cognito API または CLI、、または を使用してプログラムでユーザープールを作成する場合、 AWS CloudFormationユーザープールは AWS CDK、 のSourceArnパラメータでユーザープールにEmailConfiguration指定されている Amazon SES ID を含む E メールメッセージを送信します。Amazon SES ID は、サポートされている を占有する必要があります AWS リージョン。EmailSendingAccount が COGNITO_DEFAULT で SourceArn パラメータを指定しない場合、Amazon Cognito はユーザープールを作成したリージョン内のリソースを使用して no-reply@verificationemail.com から E メールメッセージを送信します。

次の表は、Amazon Cognito で Amazon SES ID AWS リージョン を使用できる を示しています。 Amazon Cognito

ユーザープールリージョン	リージョンオプション	Amazon SES でサポートされているリージョン
米国東部 (バージニア北部)	下位互換性	米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
米国東部 (オハイオ)	下位互換性	米国東部 (オハイオ)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
米国西部 (北カリフォルニア)	リージョン内のみ	米国西部 (北カリフォルニア)
米国西部 (オレゴン)	下位互換性	米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)

ユーザープールリージョン	リージョンオプション	Amazon SES でサポートされているリージョン
カナダ (中部)	下位互換性	カナダ (中部)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (東京)	下位互換性	アジアパシフィック (東京)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (ソウル)	下位互換性	アジアパシフィック (ソウル)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (ムンバイ)	下位互換性	アジアパシフィック (ムンバイ)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (ハイデラバード)	代替リージョン	アジアパシフィック (ムンバイ) 、アジアパシフィック (シンガポール) [1]
アジアパシフィック (シンガポール)	下位互換性	アジアパシフィック (シンガポール)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (シドニー)	下位互換性	アジアパシフィック (シドニー)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
アジアパシフィック (大阪)	リージョン内のみ	アジアパシフィック (大阪)

ユーザープールリージョン	リージョンオプション	Amazon SES でサポートされているリージョン
アジアパシフィック (ジャカルタ)	リージョン内のみ	アジアパシフィック (ジャカルタ)
アジアパシフィック (メルボルン)	代替リージョン	アジアパシフィック (シドニー)、アジアパシフィック (シンガポール)[1]
欧州 (アイルランド)	下位互換性	米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
欧州 (ロンドン)	下位互換性	欧州 (ロンドン)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
欧州 (パリ)	リージョン内のみ	欧州 (パリ)
欧州 (フランクフルト)	下位互換性	欧州 (フランクフルト)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)
欧州 (チューリッヒ)	代替リージョン	欧州 (フランクフルト)、欧州 (ロンドン)[1]
欧州 (ストックホルム)	リージョン内のみ	欧州 (ストックホルム)
欧州 (ミラノ)	リージョン内のみ	欧州 (ミラノ)
欧州 (スペイン)	代替リージョン	欧州 (パリ)、欧州 (ストックホルム)[1]
中東 (バーレーン)	リージョン内のみ	中東 (バーレーン)

ユーザープールリージョン	リージョンオプション	Amazon SES でサポートされているリージョン
中東 (アラブ首長国連邦)	代替リージョン	欧州 (フランクフルト)、欧州 (ロンドン) [1]
南米 (サンパウロ)	リージョン内のみ	南米 (サンパウロ)
イスラエル (テルアビブ)	リージョン内のみ	イスラエル (テルアビブ)
アフリカ (ケープタウン)	リージョン内のみ	アフリカ (ケープタウン)

[1] デフォルトの E メール設定のユーザープールで使用されます。Amazon Cognito は、各リージョンで同じ E メールアドレスを持つ検証済み ID 間で E メールメッセージを配信します。カスタムFROMアドレスを使用するには、 形式の SourceArn パラメータEmailConfigurationを使用して を設定しますarn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。

ユーザープールの E メールの設定

ユーザープールの E メール設定を指定するには、以下のステップを完了します。使用する設定に応じて、Amazon SES、 AWS Identity and Access Management (IAM)、および Amazon Cognito での IAM アクセス許可が必要になる場合があります。

> **Note**
>
> これらのステップで作成したリソースを AWS アカウント間で共有することはできません。例えば、あるアカウントのユーザープールを設定してから、別のアカウントの Amazon SES E メールアドレスで使用することはできません。複数のアカウントで Amazon Cognito を使用する場合は、それぞれのアカウントでこれらのステップを繰り返すようにしてください。

ステップ 1: Amazon SES での E メールアドレスを検証する

次のいずれかを行う場合は、ユーザープールを設定する前に、Amazon SES で 1 つ、または複数の E メールアドレスを検証する必要があります。

- FROM アドレスとして独自の E メールアドレスを使用する
- Amazon SES の設定を使用して E メール配信を処理する

E メールアドレスを検証することにより、そのアドレスを所有していることを確認する (不正使用の防止に役立ちます)

Amazon SES での E メールアドレスの検証については、Amazon Simple Email Service デベロッパーガイドの「E メールアドレスの検証」を参照してください。Amazon SES でのドメインの検証の詳細については、「ドメインの検証」を参照してください。

ステップ 2: Amazon SES サンドボックスからアカウントを移動する

デフォルトの Amazon Cognito E メール設定を使用している場合は、このステップを省略します。

で Amazon SES を初めて使用すると AWS リージョン、 はそのリージョン AWS アカウント の Amazon SES サンドボックスに配置されます。Amazon SES は、サンドボックスを使用して不正使用や悪用を防ぎます。Amazon SES 設定を使用して E メール配信を処理している場合は、Amazon Cognito がユーザーに E メールを送信する前に、サンドボックスから AWS アカウント を移動させる必要があります。

サンドボックスでは、Amazon SES が、送信できる E メールの数と、それらの送信先に制限を課します。E メールは、Amazon SES で検証したアドレスとドメインのみに送信できます。または、Amazon SES メールボックスシミュレーターのアドレスにも E メールを送信できます。がサンドボックスに AWS アカウント 残っている間は、本番環境のアプリケーションに Amazon SES 設定を使用しないでください。この状況では、Amazon Cognito がユーザーの E メールアドレスにメッセージを送信することができません。

サンドボックス AWS アカウント から を削除するには、Amazon SES サンドボックス外への移動」を参照してください。

ステップ 3: Amazon Cognito に E メール許可を付与する

Amazon Cognito がユーザーに E メールを送信する前に、特定の許可を Amazon Cognito に付与する必要がある場合があります。付与するアクセス許可と付与に使用するプロセスは、デフォルトの E メール設定を使用しているか、Amazon SES 設定を使用しているかによって異なります。

デフォルトの E メール設定を使用するアクセス許可を付与するには

このステップは、Cognito で E メールを送信するようにユーザープールを設定するか、`EmailSendingAccount`を `COGNITO_DEFAULT`に設定する場合にのみ実行します。

デフォルトの E メール設定では、ユーザープールは次のいずれかのアドレスで E メールメッセージを送信できます。

- デフォルトのアドレス no-reply@verificationemail.com。
- Amazon SES で検証済みの E メールアドレスまたはドメインからのカスタム FROM アドレス。

カスタムアドレスを使用している場合、Amazon Cognito には、そのアドレスからユーザーに E メールを送信するための追加のアクセス許可が必要です。これらのアクセス許可は、Amazon SES のアドレスまたはドメインの送信承認ポリシーによって付与されます。Amazon Cognito コンソールを使用してユーザープールにカスタムアドレスを追加する場合、ポリシーは Amazon SES 検証済み E メールアドレスに自動的にアタッチされます。ただし、 AWS CLI や Amazon Cognito API など、コンソールの外部でユーザープールを設定する場合は、Amazon SES コンソールまたは API を使用してポリシーをアタッチする必要があります。 PutIdentityPolicy

> **Note**
>
> FROM アドレスは、 AWS CLI または Amazon Cognito API を使用して検証済みドメインでのみ設定できます。

送信承認ポリシーは、Amazon Cognito を使用して Amazon SES を呼び出すアカウントリソースに基づいてアクセスを許可、または拒否します。リソースベースのポリシーに関する詳細については、IAM ユーザーガイドを参照してください。リソースベースのポリシーの例についても、Amazon SES デベロッパーガイドで確認いただけます。

Example 送信承認ポリシー

次の送信承認ポリシーの例では、Amazon SES 検証済み ID を使用するための限定された権限が Amazon Cognito に付与されています。Amazon Cognito は、条件 `aws:SourceArn` のユーザープールと条件 `aws:SourceAccount` のアカウントの両方に代わって送信する場合にのみ、E メールメッセージを送信できます。

Regions with Amazon SES

ユーザープールリージョンまたは代替リージョンの送信承認ポリシーでは、Amazon Cognito サービスプリンシパルが E メールメッセージを送信することを許可する必要があります。詳細については、リージョン表を参照してください。ユーザープールリージョン が Amazon SES リージョン の少なくとも 1 つの値と一致する場合は、次の例で グローバルサービスプリンシパルを使用して送信承認ポリシーを設定します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

Opt-in Regions without Amazon SES

Amazon SES は、Amazon Cognito AWS リージョン が利用可能なすべてのオプトインで利用できるわけではありません。中東 (UAE) は一例であり、 は欧州 (フランクフルト) () で検証済みの ID を持つ E メールのみを送信できますeu-central-1。デフォルトの E メール設定を持つユーザープールでは、Amazon Cognito は 2 つのリージョンのそれぞれで検証済み ID を含む E メールメッセージも送信します。中東 (アラブ首長国連邦) の場合、追加のリージョンは欧州 (ロンドン) です。両方のリージョンで送信承認ポリシーを更新する必要があります。

各代替リージョンの送信承認ポリシーでは、ユーザープールオプトインリージョンの Amazon Cognito サービスプリンシパルが E メールメッセージを送信することを許可する必要があります。詳細については、リージョン表を参照してください。リージョンが代替リージョン としてマークされている場合は、次の例のように、リージョンサービスプリンシパルを使用して送信承認ポリシーを設定します。必要に応じて、サンプルリージョン識別子 *me-central-1* を必要なリージョン ID に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

ポリシー構文の詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES 送信承認ポリシー」を参照してください。

その他の例については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES 送信承認ポリシーの例」を参照してください。

Amazon SES の設定を使用する許可を付与する

Amazon SES の設定を使用するようにユーザープールを設定する場合、Amazon Cognito には、ユーザーに E メールを送信するときに代理で Amazon SES を呼び出すための追加の許可が必要です。この承認は、IAM サービスで付与されます。

このオプションでユーザープールを設定すると、Amazon Cognito が AWS アカウントにサービスリンクロール (IAM ロールの一種) を作成します。このロールには、Amazon Cognito が Amazon SES

にアクセスし、お客様のアドレスを使用して E メールを送信できるようにする許可が含まれています。

Amazon Cognito は、設定を設定するユーザーセッションの AWS 認証情報を使用して、サービスにリンクされたロールを作成します。このセッションの IAM 権限には、`iam:CreateServiceLinkedRole` アクションが含まれている必要があります。IAM のアクセス許可の詳細については、IAM ユーザーガイドの「AWS リソースのアクセス管理」を参照してください。

Amazon Cognito が作成するサービスリンクロールの詳細については、「Amazon Cognito のサービスリンクロールの使用」を参照してください。

ステップ 4: ユーザープールを設定する

以下のいずれかを使用してユーザープールを設定する場合は、次のステップを完了します。

- E メール送信者として表示されるカスタムの FROM アドレス
- ユーザーが FROM アドレスに送信するメッセージを受信するカスタムの REPLY-TO アドレス
- Amazon SES の設定

> **Note**
>
> 検証済み ID が E メールアドレスの場合、Amazon Cognito はデフォルトで、その E メールアドレスを FROM と REPLY-TO の E メールアドレスとして設定します。ただし、検証済み ID がドメインの場合は、FROM と REPLY-TO の E メールアドレスに値を指定する必要があります。例えば、検証済みドメインが example.com の場合、FROM と REPLY-TO の E メールアドレスの両方として no-reply@example.com を設定できます。

デフォルトの Amazon Cognito E メール設定とアドレスを使用する場合は、この手順を省略します。

カスタム E メールアドレスを使用するようにユーザープールを設定する方法

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。

4. [Messaging] (メッセージング) タブを選択して、[Email configuration] (E メールの設定) を検索して、[Edit] (編集) をクリックします。
5. [Edit email configuration] (E メールの設定) の編集ページで、[Send email from Amazon SES] (Amazon SES からメールを送信する) または[Send email with Amazon Cognito] (Amazon Cognito で E メールを送信する) を選択します。[Send email from Amazon SES] (Amazon SES からメールを送信する) を選択した場合のみ、[SES Region] (SES リージョン)、[Configuration Set] (設定セット)、および [FROM sender name] (FROM 送信者名) をカスタマイズできます。
6. カスタムの FROM アドレスを使用するには、次の手順を実行します。

   a. [SES Region] (SES リージョン) で、検証済み E メールアドレスが含まれているリージョンを選択します。
   b. [FROM email address] (FROM E メールアドレス) で、E メールアドレスを選択します。Amazon SES で検証済みの E メールアドレスを使用します。
   c. (オプション) [Configuration set] (設定セット) で Amazon SES で使用する設定セットを選択します。この変更を行って保存すると、サービスにリンクされたロールが作成されます。
   d. (オプション) [FROM sender address] (FROM 送信者アドレス) で E メールアドレスを入力します。E メールアドレスだけ、または E メールアドレスと分かりやすい名前を `Jane Doe <janedoe@example.com>` 形式で指定できます。
   e. (オプション) [REPLY-TO email address] (REPLY-TO E メールアドレス) で、ユーザーが FROM アドレスに送信するメッセージを受信する E メールアドレスを入力します。
7. [変更の保存] を選択します。

関連トピック

- E メール検証メッセージのカスタマイズ
- ユーザー招待メッセージのカスタマイズ

Amazon Cognito ユーザープール用の SMS メッセージ設定

ユーザープールの Amazon Cognito イベントには、Amazon Cognito がユーザーに SMS テキストメッセージを送信する結果となるものがあります。例えば、電話番号の検証を必須とするようにユーザープールを設定している場合、ユーザーがアプリケーションの新しいアカウントにサインアップする、またはパスワードをリセットする場合に Amazon Cognito が SMS テキストメッセージを送信します。SMS テキストメッセージを開始するアクションに応じて、メッセージには検証コード、一時的なパスワード、またはウェルカムメッセージが含まれます。

Amazon Cognito は、SMS テキストメッセージの配信に Amazon Simple Notification Service (Amazon SNS) を使用します。Amazon Cognito または Amazon SNS 経由でのテキストメッセージの送信を初めて行う場合は、Amazon SNS によってサンドボックス環境に置かれます。サンドボックス環境では、SMS テキストメッセージのアプリケーションをテストできます。サンドボックスでは、検証済みの電話番号にしかメッセージを送信できません。

Amazon SNS は、SMS テキストメッセージに料金を請求します。詳細については、「Amazon SNS の料金」を参照してください。

> **Note**
>
> 世界中で一方的な SMS トラフィックの量が多いため、一部の政府は SMS メッセージの送信者と受信者の間に障壁を課しています。SMS メッセージを MFA やユーザーアップデートに使用する場合は、メッセージが確実に送られるように追加の手順を踏む必要があります。また、ユーザーが住んでいる可能性のある国の SMS メッセージ関連の規制を監視し、SMS メッセージの設定を最新の状態に保つ必要があります。詳細については、Amazon Simple Notification Service デベロッパーガイドの「Mobile text messaging (SMS)」を参照してください。
>
> SMS メッセージを使用してユーザーを認証および検証することは、セキュリティ上のベストプラクティスではありません。電話番号は所有者が変わる場合があり、ユーザーの MFA の所有している要素を確実に表さないかもしれません。代わりに、アプリまたはサードパーティーの IdP で TOTP MFA を実装してください。また、カスタム認証チャレンジの Lambda トリガー を使って、追加のカスタム認証要素を作成することもできます。

Amazon Cognito は、ユーザーが入力できるコードを含む SMS メッセージをユーザーに送信します。次の表は、SMS メッセージを生成できるイベントを示しています。

メッセージオプション

アクティビティ	API オペレーション	配信オプション	形式オプション	カスタマイズ可能	メッセージテンプレート
パスワードを忘れてしまった	ForgotPassword	E メール、SMS	コード	いいえ	該当なし

アクティビティ	API オペレーション	配信オプション	形式オプション	カスタマイズ可能	メッセージテンプレート
招待	AdminCreateUser	E メール、SMS	コード	はい	招待メッセージ
自己登録	SignUp	E メール、SMS	コード、リンク	はい	検証メッセージ
E メールアドレスまたは電話番号の検証	UpdateUserAttributes	E メール、SMS	コード	はい	検証メッセージ
多要素認証 (MFA)	AdminInitiateAuth, InitiateAuth	SMS、認証アプリ	コード	Yes[1]	MFA メッセージ

[1] SMS メッセージ用。

Amazon Cognito ユーザープールでの SMS メッセージングの初回セットアップ

Amazon Cognito は、ユーザーへの SMS メッセージの送信に Amazon SNS を使用します。カスタム SMS 送信者の Lambda トリガー を使用して、独自のリソースを使用し、SMS メッセージを送信することもできます。特定の で SMS テキストメッセージを送信するように Amazon SNS を初めて設定すると AWS リージョン、Amazon SNS はそのリージョンの AWS アカウント SMS サンドボックスに を配置します。Amazon SNS はサンドボックスを使用して、不正使用や不正使用を防止し、コンプライアンス要件を満たします。 AWS アカウント がサンドボックスにある場合、Amazon SNS はいくつかの制限を課します。例えば、テキストメッセージを送信できるのは Amazon SNS で検証した電話番号 10 個までです。がサンドボックスに AWS アカウント 残っている間は、本番環境のアプリケーションに Amazon SNS 設定を使用しないでください。サンドボックスに置かれている間、Amazon Cognito はユーザーの電話番号にメッセージを送信できません。

SMS テキストメッセージをユーザープールユーザーに送信するには

1. Amazon Cognito が Amazon SNS で SMS メッセージを送信するために使用できる IAM ロールを準備します。
2. Amazon SNS SMS メッセージの AWS リージョン を選択する
3. 米国の電話番号に SMS メッセージを送信するための発信元 ID を取得する
4. SMS サンドボックスに置かれていることを確認する
5. Amazon SNS サンドボックスからアカウントを移動させる
6. Amazon SNS で Amazon Cognito の電話番号を検証する
7. Amazon Cognito でユーザープールのセットアップを完了する

Amazon Cognito が Amazon SNS で SMS メッセージを送信するために使用できる IAM ロールを準備します。

ユーザープールから SMS メッセージを送信すると、Amazon Cognito はアカウントの IAM ロールを引き受けます。Amazon Cognito は、そのロールに割り当てられた `sns:Publish` アクセス許可を使用して、ユーザーに SMS メッセージを送信します。Amazon Cognito コンソールでは、[SMS] のユーザープールの [Messaging] (メッセージング) タブから [IAM role selection] (IAM ロールの選択) を設定するか、ユーザープール作成ウィザードでこの選択を行うことができます。

次の IAM ロール信頼ポリシーの例では、Amazon Cognito ユーザープールに、ロールを引き受ける制限付きの機能を付与しています。Amazon Cognito は、条件 `aws:SourceArn` のユーザープールと条件 `aws:SourceAccount` の AWS アカウント の代わりに行う場合にのみ、そのロールを引き受けることができます。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "<your account number>"
            },
```

```
            "ArnLike": {
                "aws:SourceArn": "<your user pool ARN>"
            }
        }
    }]
}
```

条件 aws:SourceArn の値には、正確なユーザープール ARN またはワイルドカード ARN を指定できます。 AWS Management Console または DescribeUserPool API リクエストを使用して、ユーザープールの ARNs を検索します。

IAM ロールおよび信頼ポリシーの詳細については、「AWS Identity and Access Management ユーザーガイド」の「ロールに関する用語と概念」を参照してください。

Amazon SNS SMS メッセージの AWS リージョン を選択する

一部の では AWS リージョン、Amazon Cognito SMS メッセージに使用する Amazon SNS リソースを含むリージョンを選択できます。 Amazon Cognito アジアパシフィック (ソウル) を除く Amazon Cognito が利用可能なすべての AWS リージョン では、ユーザープール AWS リージョン を作成したで Amazon SNS リソースを使用できます。リージョンを選択したときに SMS メッセージングを高速かつ信頼性の高いものにするには、ユーザープールと同じリージョンの Amazon SNS リソースを使用します。

> **Note**
>
> では AWS Management Console、新しい Amazon Cognito コンソールエクスペリエンスに切り替えた後にのみ、SMS リソースのリージョンを変更できます。

新規ユーザープールウィザードのメッセージ配信の設定ステップで、SMS リソースのリージョンを選択します。既存のユーザープールの [Messaging] (メッセージング) タブの [SMS] で [Edit] (編集) を選択することもできます。

起動時に、一部の について AWS リージョン、Amazon Cognito は代替リージョンで Amazon SNS リソースを含む SMS メッセージを送信しました。任意のリージョンを設定するには、ユーザープールの SmsConfigurationType オブジェクトの SnsRegionパラメータを使用します。次の表から Amazon Cognito リージョン に Amazon Cognito ユーザープールリソースをプログラムによって作成し、SnsRegion パラメータを提供しない場合、ユーザープールはレガシー Amazon SNS 代替リージョンで Amazon SNS リソースを使用して SMS メッセージを送信します。

アジアパシフィック (ソウル) の Amazon Cognito ユーザープール AWS リージョン は、アジアパシフィック (東京) リージョンの Amazon SNS 設定を使用する必要があります。

Amazon SNS は、すべての新しいアカウントに対して 1 か月あたり 1.00 USD で使用クォータを設定します。Amazon Cognito AWS リージョン で使用する で支出制限が引き上げられた可能性があります。 AWS リージョン Amazon SNS SMS メッセージの を変更する前に、 AWS サポートセンターでクォータ引き上げケースを開いて、新しいリージョンの制限を引き上げてください。詳細については、Amazon Simple Notification Service 開発者ガイド の「Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする」を参照してください。

以下の表にある、任意の Amazon Cognito リージョンに対して、対応する Amazon SNS リージョンの Amazon SNS リソースで SMS メッセージを送信することができます。

Amazon Cognito リージョン	Amazon SNS リージョン
米国東部 (オハイオ)	米国東部 (オハイオ)、米国東部 (バージニア北部)
カナダ (中部)	カナダ (中部)、米国東部 (バージニア北部)
欧州 (フランクフルト)	欧州 (フランクフルト)、欧州 (アイルランド)
欧州 (ロンドン)	欧州 (ロンドン)、欧州 (アイルランド)
アジアパシフィック (ソウル)	アジアパシフィック (東京)
米国東部 (バージニア北部)	米国東部 (バージニア北部)
米国西部 (北カリフォルニア)	米国西部 (北カリフォルニア)
米国西部 (オレゴン)	米国西部 (オレゴン)
アジアパシフィック (ムンバイ)	アジアパシフィック (ムンバイ)、アジアパシフィック (シンガポール)
アジアパシフィック (ハイデラバード)	アジアパシフィック (ハイデラバード)
アジアパシフィック (シンガポール)	アジアパシフィック (シンガポール)
アジアパシフィック (シドニー)	アジアパシフィック (シドニー)

Amazon Cognito リージョン	Amazon SNS リージョン
アジアパシフィック (東京)	アジアパシフィック (東京)
アジアパシフィック (ジャカルタ)	アジアパシフィック (ジャカルタ)
アジアパシフィック (大阪)	アジアパシフィック (大阪)
アジアパシフィック (メルボルン)	アジアパシフィック (メルボルン)
欧州 (アイルランド)	欧州 (アイルランド)
欧州 (パリ)	欧州 (パリ)
欧州 (ストックホルム)	欧州 (ストックホルム)
欧州 (ミラノ)	欧州 (ミラノ)
欧州 (スペイン)	欧州 (スペイン)
中東 (バーレーン)	中東 (バーレーン)
南米 (サンパウロ)	南米 (サンパウロ)
イスラエル (テルアビブ)	イスラエル (テルアビブ)
アフリカ (ケープタウン)	アフリカ (ケープタウン)
中東 (アラブ首長国連邦)	中東 (アラブ首長国連邦)
欧州 (チューリッヒ)	欧州 (チューリッヒ)

米国の電話番号に SMS メッセージを送信するための発信元 ID を取得する

米国の電話番号に SMS テキストメッセージを送信する場合は、SMS サンドボックステスト環境または本番環境を構築するかどうかにかかわらず、発信元 ID を取得する必要があります。

米国の通信事業者は 2021 年 6 月 1 日付けで、米国の電話番号へのメッセージの送信に発信元 ID を義務付けました。発信元 ID をまだ取得していない場合は、取得する必要があります。発信元 ID を取得する方法については、Amazon Pinpoint ユーザーガイドの「番号のリクエスト」を参照してください。

次の で運用している場合は AWS リージョン、 AWS Support チケットを開いて発信元 ID を取得する必要があります。この手順については、Amazon Simple Notification Service デベロッパーガイドの「SMS メッセージングのサポートをリクエストする」を参照してください。

- 米国東部 (オハイオ)
- 欧州 (ストックホルム)
- 欧州 (パリ)
- ヨーロッパ (ミラノ)
- 中東 (バーレーン)
- 南米 (サンパウロ)
- 米国西部 (北カリフォルニア)

同じ に複数の発信元 ID がある場合 AWS リージョン、Amazon SNS は、ショートコード、10DLC、通話料無料番号の優先順位で発信元 ID タイプを選択します。この優先順位を変更することはできません。詳細については、「Amazon SNS のよくある質問」を参照してください。

SMS サンドボックスに置かれていることを確認する

次の手順を使用して SMS サンドボックスに移動していることを確認します。本番稼働用 Amazon Cognito ユーザープール AWS リージョン がある各 に対して、同じ手順を繰り返します。

Amazon Cognito コンソールで SMS サンドボックスステータスを確認します。

SMS サンドボックスに置かれていることを確認する

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから存在するユーザープールを 1 つ選択します。
4. [Messaging] (メッセージング) タブを選択します。
5. [SMS configuration] (SMS 設定) セクションで、[Move to Amazon SNS production environment] (Amazon SNS 本番環境に移行する) を展開します。アカウントが SMS サンドボックスに置かれている場合は、以下のメッセージが表示されます。

   ```
   You are currently in the SMS Sandbox and cannot send SMS messages to
   unverified numbers.
   ```

このメッセージが表示されない場合は、アカウント内での SMS メッセージのセットアップが既に実行されています。「Amazon Cognito でユーザープールのセットアップを完了する」へ進んでください。

6. メッセージの Amazon SNS リンクを選択します。これにより、新しいタブで Amazon SNS コンソールが開きます。
7. サンドボックス環境に置かれていることを確認します。コンソールメッセージには AWS リージョン、サンドボックスのステータスと が次のように表示されます。

   ```
   This account is in the SMS sandbox in US East (N. Virginia).
   ```

Amazon SNS サンドボックスからアカウントを移動させる

アプリをテストしていて、管理者が検証できる電話番号にのみ SMS メッセージを送信する必要がある場合は、このステップをスキップしてください。

本番稼働でアプリを使用するには、アカウントを SMS サンドボックスから本番環境に移動します。Amazon Cognito が使用する Amazon SNS リソースを含む で AWS リージョン 発信元 ID を設定したら、 が SMS サンドボックスに AWS アカウント 残っている間、米国の電話番号を確認できます。Amazon SNS 環境が本番環境にある場合、ユーザーに SMS メッセージを送信するために Amazon SNS でユーザーの電話番号を確認する必要はありません。

詳細については、Amazon Simple Notification Service デベロッパーガイドの「SNS サンドボックス外への移動」を参照してください。

Amazon SNS で Amazon Cognito の電話番号を検証する

SMS サンドボックスからアカウントを移動した場合は、このステップをスキップします。

SMS サンドボックスでは、Amazon SNS で検証した任意の電話番号にメッセージを送信できます。

電話番号を確認するには、次の手順を実行します。

1. Amazon SNS コンソールのテキストメッセージング (SMS) セクションに、サンドボックスの送信先電話番号を追加します。
2. 指定した電話番号にコードが入った SMS メッセージを受信します。
3. Amazon SNS コンソールに SMS メッセージに記載されている確認コードを入力します。

詳しい手順については、Amazon Simple Notification Service デベロッパーガイドの「SMS サンドボックスでの電話番号の追加と確認」を参照してください。

> **Note**
>
> Amazon SNS では、SMS サンドボックスで確認できる送信先の電話番号の数が制限されます。Amazon Simple Notification Service デベロッパーガイドの「SMS サンドボックス」を参照してください。

Amazon Cognito でユーザープールのセットアップを完了する

ユーザープールを作成または編集していたブラウザタブに戻ります。手順を完了します。ユーザープールに SMS 設定を正常に追加すると、Amazon Cognito は内部の電話番号にテストメッセージを送信して、設定が機能することを確認します。Amazon SNS は、テスト SMS メッセージごとに料金を請求します。

ユーザープールでのトークンの使用

トークンを使用してユーザーを認証し、リソースへのアクセス権を付与します。トークンのクレームはユーザーに関する情報です。ID トークンには、ユーザー名、苗字、E メールアドレスなど、身元に関するクレームが含まれています。アクセストークンには、認証されたユーザーがサードパーティー API、Amazon Cognito ユーザーのセルフサービス API オペレーション、および UserInfo エンドポイント にアクセスできる `scope` のようなクレームが含まれています。アクセストークンと ID トークンの両方に、ユーザープール内でのユーザーのグループメンバーシップを示す `cognito:groups` クレームが含まれています。ユーザープールグループの詳細については、「ユーザープールにグループを追加する」を参照してください。

Amazon Cognito には、新しいトークンの取得や既存のトークンの取り消しに使用できる更新トークンもあります。トークンを更新して、新しい ID とアクセストークンを取得します。トークンを取り消して、更新トークンによって許可されているユーザーアクセスを取り消します。

Amazon Cognito は、Base 64 でエンコードされた文字列としてトークンを発行します 任意の Amazon Cognito ID またはアクセストークンを base64 からプレーンテキスト JSON にデコードできます。Amazon Cognito の更新トークンは暗号化されており、ユーザープールのユーザーと管理者に対して透過的ではなく、ユーザープールのみが読み取ることができます。

トークンを使用した認証

ユーザーがアプリケーションにサインインすると、Amazon Cognito がログイン情報を検証します。ログインが正常に行われると、Amazon Cognito がセッションを作成し、認証されたユーザーの ID トークン、アクセストークン、および更新トークンを返します。これらのトークンを使用して、ダウンストリームのリソースや Amazon API Gateway などの API へのアクセス権をユーザーに付与できます。または、これらのトークンを他の AWS のサービスにアクセスするための一時的な AWS 認証情報と交換することができます。

トークンの保存

アプリケーションでは、さまざまなサイズのトークンを保存する必要があります。トークンのサイズは、クレームの追加、エンコーディングアルゴリズムの変更、暗号化アルゴリズムの変更による理由 (ただし、これらに限定されない) で変更される場合があります。ユーザープールでトークンの取り消しを有効にした場合も、Amazon Cognito が JSON ウェブトークンにクレームを追加するので、そのトークンのサイズは増加します。アクセストークンと ID トークンに対し、新たなクレーム `origin_jti` および `jti` が追加されます。トークンの取り消しの詳細については、「トークンの取り消し」を参照してください。

> **Important**
>
> ベストプラクティスとして、アプリケーションのコンテキストにおけるすべての転送中と保管中のトークンを保護します。トークンには、アプリケーションのユーザーに関する個人識別情報、およびユーザープールで使用するセキュリティモデルに関する情報を含めることができます。

トークンをカスタマイズする

Amazon Cognito からアプリに渡すアクセストークンと ID トークンをカスタマイズできます。トークン生成前の Lambda トリガーでは、トークンクレームを追加、変更、および抑制できます。トークン生成前トリガーは、Amazon Cognito からデフォルトのクレームセットを送信する先の Lambda 関数です。クレームには、OAuth 2.0 スコープ、ユーザープールグループのメンバーシップ、ユーザー属性などが含まれます。この関数は、必要に応じてランタイムに変更を加え、更新したトークンクレームを Amazon Cognito に返すことができます。

バージョン 2 のイベントによるアクセストークンのカスタマイズには追加料金がかかります。詳細については、「Amazon Cognito の料金」を参照してください。

トピック

ID トークンの使用

ID トークンとは、name、email、および phone_number など、認証されたユーザーのアイデンティティに関するクレームが含まれる、JSON ウェブトークン (JWT) です。この ID 情報はアプリケーション内で使用できます。ID トークンは、リソースサーバーまたはサーバーアプリケーションに対するユーザーの認証にも使用できます。Web API オペレーションを使用して、アプリケーション外で ID トークンを使用することも可能です。このような場合は、ID トークン内のクレームを信頼する前に、ID トークンの署名を検証する必要があります。JSON Web トークンの検証 を参照してください。

ID トークンの有効期限は、5 分から 1 日までの任意の値に設定できます。この値は、アプリケーションのクライアントごとに設定できます。

> **Important**
>
> ユーザーがホストされた UI またはフェデレーション ID プロバイダー (IdP) でサインインすると、Amazon Cognito は 1 時間有効なセッション Cookie を設定します。ホストされた UI またはフェデレーションを使用し、アクセストークンと ID トークンに 1 時間未満の最小期間を指定すると、Cookie の有効期限が切れるまで、ユーザーは有効なセッションを継続できます。1 時間のセッション中に有効期限が切れるトークンがある場合、ユーザーは再認証しなくてもトークンを更新できます。

ID トークンのヘッダー

ヘッダーには 2 つの情報 (キー ID: kid、アルゴリズム: alg) が含まれています。

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

kid

キー ID。その値は、トークンの JSON Web 署名 (JWS) をセキュア化するために使用されたキーを示します。ユーザープール署名キー ID は `jwks_uri` エンドポイントで確認できます。

`kid` パラメータの詳細については、「Key identifier (kid) header parameter」を参照してください。

alg

Amazon Cognito が、アクセストークンをセキュア化するために使用した暗号化アルゴリズム。ユーザープールは、SHA-256 による RSA 署名である RS256 暗号化アルゴリズムを使用します。

`alg` パラメータの詳細については、「Algorithm (alg) header parameter」を参照してください。

ID トークンのデフォルトペイロード

これは ID トークンからのペイロードの例です。認証されたユーザーに関するクレームが含まれています。OpenID Connect (OIDC) 標準クレームの詳細については、OIDC 標準クレームのリストを参照してください。を使用して、独自の設計のクレームを追加できますトークン生成前の Lambda トリガー。

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
        "test-group-a",
        "test-group-b",
        "test-group-c"
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
```

```
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

sub

認証されたユーザーの固有識別子 (UID) または件名。ユーザー名はユーザープール内で一意ではない可能性があります。sub クレームは、特定のユーザーを識別する最良の方法です。

cognito:groups

ユーザーをメンバーとするユーザープールグループの名前の配列。グループは、アプリに提示する識別子として使用したり、アイデンティティプールへの優先 IAM ロールのリクエストの生成に使用したりできます。

cognito:preferred_role

ユーザーの最も優先度の高いユーザープールグループに関連付けた IAM ロールの ARN。ユーザープールがこのロールクレームを選択する方法の詳細については、「グループへの優先順位の値の割り当て」を参照してください。

iss

トークンを発行した ID プロバイダー。クレームは以下のような形式になります。

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

cognito:username

ユーザープール内のユーザーのユーザー名。

nonce

nonce クレームは、OAuth 2.0 の authorize エンドポイントへのリクエストに追加するものと、名前が同じパラメータから取得されます。パラメータを追加すると、Amazon Cognito が発行する ID トークンに nonce クレームが含められます。これは、リプレイ攻撃からの保護のために使用できます。リクエストで nonce 値を指定せずにサードパーティー ID プロバイダーを介した認証を行う場合、Amazon Cognito はノンスを自動的に生成および検証した上で、その値を nonce クレームとして ID トークンに追加します。Amazon Cognito での nonce クレームの実装は、OIDC 標準に基づいています。

origin_jti

ユーザーの更新トークンに関連付けられたトークン失効識別子。Amazon Cognito は、エンドポイントの取り消しまたは RevokeToken API オペレーションでユーザーのトークンを取り消したかどうかをチェックするときに、 origin_jtiクレームを参照します。トークンを取り消すと、Amazon Cognito は同じ origin_jti 値を持つすべてのアクセストークンと ID トークンを無効にします。

cognito:roles

ユーザーのグループに関連付けられた IAM ロールの名前の配列。各ユーザープールグループには、1 つの IAM ロールを関連付けることができます。この配列は、優先順位に関係なく、ユーザーグループのすべての IAM ロールを表します。詳細については、「ユーザープールにグループを追加する」を参照してください。

aud

ユーザーを認証したユーザープールアプリクライアント。Amazon Cognito は、アクセストークン client_id クレームで同じ値をレンダリングします。

identities

ユーザーの identities 属性の内容。この属性には、フェデレーションサインインによって、またはフェデレーションユーザーをローカルプロファイルにリンクすることによってユーザーにリンクした各サードパーティ ID プロバイダープロファイルに関する情報が含まれます。この情報には、プロバイダー名、プロバイダーの固有 ID、およびその他のメタデータが含まれます。

`token_use`

トークンの本来の目的。ID トークンでは、その値は `id` です。

`auth_time`

ユーザーが認証を完了した認証時刻 (Unix の時間形式)。

`exp`

ユーザーのトークンの有効期限が切れる有効期限 (Unix の時間形式)。

`iat`

Amazon Cognito がユーザーのトークンを発行した発行時刻 (Unix の時間形式)。

`jti`

JWT の一意の識別子。

この ID トークンには、「OIDC standard claims」に定義されている、OIDC の標準クレームを含めることができます。また、ユーザープールで定義されたカスタム属性が、この ID トークンに含まれる場合もあります。Amazon Cognito は、属性タイプに関係なく、カスタム属性値を文字列として ID トークンに書き込みます。

> **Note**
>
> ユーザープールのカスタム属性には、常に というプレフィックスが付けられます`custom:`。

ID トークンの署名

ID トークンの署名は、JWT トークンのヘッダーとペイロードに基づいて計算されます。アプリが受け取る ID トークンのクレームを受け入れる前に、トークンの署名を検証してください。詳細については、「JSON Web トークンの検証」を参照してください。JSON Web トークンの検証。

アクセストークンの使用

ユーザープールアクセストークンには、認証されたユーザーに関するクレーム、ユーザーのグループのリスト、およびスコープのリストが含まれます。アクセストークンの目的は、API オペレーションを承認することです。ユーザープールは、ユーザーのセルフサービスオペレーションを許可するアク

セストークンを受け入れます。例えば、アクセストークンを使用して、ユーザーにユーザー属性を追加、変更、または削除するアクセス権を付与することができます。

ユーザープールに追加したカスタムスコープから派生した OAuth 2.0 スコープをアクセストークンに含めると、ユーザーが API から情報を取得することを承認できます。例えば、Amazon API Gateway は、Amazon Cognito のアクセストークンによる認証をサポートします。REST API オーソライザーにユーザープールからの情報を入力することも、Amazon Cognito を HTTP API の JSON ウェブトークン (JWT) オーソライザーとして使用することもできます。カスタムスコープでアクセストークンを生成するには、ユーザープールのパブリックエンドポイントを通じてアクセストークンをリクエストする必要があります。

ユーザーのアクセストークンは、UserInfo エンドポイント からユーザーの属性に関する詳細情報をリクエストするためのアクセス許可です。ユーザーのアクセストークンは、ユーザー属性の読み取りと書き込みのアクセス許可でもあります。アクセストークンによって付与される属性へのアクセスレベルは、アプリクライアントに割り当てるアクセス許可と、トークンで付与するスコープによって異なります。

アクセストークンは、JSON ウェブトークン (JWT) です。アクセストークンのヘッダーは ID トークンと同じ構造になっていますが、Amazon Cognito は、ID トークンに署名するキーとは異なるキーでアクセストークンに署名します。アクセスキー ID (`kid`) クレームの値は、同じユーザーセッションの ID トークンの `kid` クレームの値と一致しません。アプリコードで、ID トークンとアクセストークンを個別に検証します。署名を検証するまでは、アクセストークンのクレームを信用しないでください。詳細については、「JSON Web トークンの検証」を参照してください。アクセストークンの有効期限は、5 分から 1 日までの間で任意の値に設定できます。この値は、アプリケーションのクライアントごとに設定できます。

> **Important**
>
> ホストされた UI を使用する場合、アクセストークンと ID トークンには、1 時間未満の最小値を指定しないでください。Amazon Cognito HostedUI は、有効期間が 1 時間のクッキーを使用します。最小値として 1 時間未満を入力した場合でも、有効期間は 1 時間より短くなりません。

アクセストークンのヘッダー

ヘッダーには 2 つの情報 (キー ID: `kid`、アルゴリズム: `alg`) が含まれています。

```
{
```

```
"kid" : "1234example="
"alg" : "RS256",
}
```

kid

キー ID。その値は、トークンの JSON Web 署名 (JWS) をセキュア化するために使用されたキーを示します。ユーザープール署名キー ID は `jwks_uri` エンドポイントで確認できます。

`kid` パラメータの詳細については、「Key identifier (kid) header parameter」を参照してください。

alg

Amazon Cognito が、アクセストークンをセキュア化するために使用した暗号化アルゴリズム。ユーザープールは、SHA-256 による RSA 署名である RS256 暗号化アルゴリズムを使用します。

`alg` パラメータの詳細については、「Algorithm (alg) header parameter」を参照してください。

アクセストークンのデフォルトペイロード

以下は、アクセストークンのサンプルペイロードです。詳細については、「JWT claims」を参照してください。を使用して、独自の設計のクレームを追加できますトークン生成前の Lambda トリガー。

```
<header>.
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "testgroup"
   ],
   "iss":"https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
   "version":2,
   "client_id":"xxxxxxxxxxxxexample",
   "origin_jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "event_id":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "token_use":"access",
   "scope":"phone openid profile resourceserver.1/appclient2 email",
   "auth_time":1676313851,
   "exp":1676317451,
```

```
   "iat":1676313851,
   "jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "username":"my-test-user"
}
.<token signature>
```

sub

認証されたユーザーの固有識別子 (UID) または件名。ユーザー名はユーザープール内で一意ではない可能性があります。sub クレームは、特定のユーザーを識別する最良の方法です。

cognito:groups

ユーザーをメンバーとするユーザープールグループの名前の配列。

iss

トークンを発行した ID プロバイダー。クレームは以下のような形式になります。

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

client_id

ユーザーを認証したユーザープールアプリクライアント。Amazon Cognito は、ID トークン aud クレームで同じ値をレンダリングします。

origin_jti

ユーザーの更新トークンに関連付けられたトークン失効識別子。Amazon Cognito は、エンドポイントの取り消しまたは RevokeToken API オペレーションでユーザーのトークンを取り消したかどうかをチェックするときに、 origin_jtiクレームを参照します。トークンを取り消すと、Amazon Cognito は同じ origin_jti 値を持つすべてのアクセストークンと ID トークンを無効にします。

token_use

トークンの本来の目的。アクセストークンでは、その値は access です。

scope

トークンが許可するアクセスを定義する OAuth 2.0 スコープのリスト。トークンエンドポイント のトークンには、アプリクライアントがサポートする任意のスコープを含めることができます。Amazon Cognito API サインインからのトークンにはスコープ aws.cognito.signin.user.admin のみが含まれます。

auth_time

ユーザーが認証を完了した認証時刻 (Unix の時間形式)。

exp

ユーザーのトークンの有効期限が切れる有効期限 (Unix の時間形式)。

iat

Amazon Cognito がユーザーのトークンを発行した発行時刻 (Unix の時間形式)。

jti

JWT の一意の識別子。

username

ユーザープール内のユーザーのユーザー名。

アクセストークンの署名

アクセストークンの署名は、JWT トークンのヘッダーおよびペイロードに基づいて計算されます。Web API のアプリケーション外で使用するときは、常にこの署名を検証してからトークンを受け入れる必要があります。詳細については、「JSON Web トークンの検証」を参照してください。

更新トークンの使用

更新トークンを使用して、新しい ID およびアクセストークンを取得できます。更新トークンはデフォルトで、アプリケーションユーザーがユーザープールにサインインしてから 30 日後に有効期限が切れます。ユーザープールのアプリケーションを作成するときは、アプリケーションの更新トークンの有効期限を 60 分から 10 年までの任意の値に設定できます。

Mobile SDK for iOS、Mobile SDK for Android、Amplify for iOS、Amplify for Android、および Amplify for Flutter は、有効な (期限が切れていない) 更新トークンが存在する場合に ID トークンとアクセストークンを自動的に更新します。ID トークンとアクセストークンには、2 分間の最小残存有効期間があります。更新トークンが有効期限切れになっている場合、アプリケーションユーザーはユーザープールに再度サインインすることで再認証される必要があります。アクセストークンと ID トークンの最小値が 5 分に設定されていて、SDK を使用している場合、更新トークンは、新しいアクセストークンと ID トークンを取得するために継続的に使用されます。期待される動作は、最小値を 5 分ではなく 7 分以上を設定することで確認できます。

ユーザーのアカウント自体は、ユーザーが新しいアカウントの UnusedAccountValidityDays 時間制限前に少なくとも 1 回ログインしている限り、その有効期限が切れることはありません。

更新トークンによる新しいアクセストークンと ID トークンの取得

更新トークンの認証を開始するには、API または ホストされた UI を使用します。

更新トークンを使用してユーザープール API で新しい ID トークンとアクセストークンを取得するには、 AdminInitiateAuthまたは InitiateAuth API オペレーションを使用します。AuthFlow パラメータの REFRESH_TOKEN_AUTH を渡します。AuthFlow の AuthParameters プロパティで、ユーザーの更新トークンを "REFRESH_TOKEN" の値として渡します。Amazon Cognito は、API リクエストがすべてのチャレンジを通過した後、新しい ID とアクセストークンを返します。

> **Note**
>
> Amazon Cognito ユーザープール API を使用してホストされた UI ユーザーのトークンを更新するには、InitiateAuth リクエストを生成します。

更新トークンは、ドメインを設定したユーザープール内の トークンエンドポイント に送信することもできます。リクエスト本文には、refresh_token の grant_type 値とユーザーの更新トークンの refresh_token 値を含めます。

更新トークンの取り消し

ユーザーに属する更新トークンを取り消すことができます。トークンの取り消しの詳細については、「トークンの取り消し」を参照してください。

> **Note**
>
> 更新トークンを取り消すと、Amazon Cognito がそのトークンを使用して更新リクエストから発行したすべての ID トークンとアクセストークンが取り消されます。

ユーザーのすべてのトークンを GlobalSignOut および AdminUserGlobalSignOut API オペレーションを使用して取り消すことで、そのユーザーは、現在サインインしているすべてのデバイスからサインアウトできるようになります。ユーザーがサインアウトすると、以下の結果になります。

- ユーザーの新しいトークンの取得にユーザーの更新トークンを使用できない。

- ユーザーのアクセストークンは、トークン認証された API リクエストを行うことができない。
- 新しいトークンを取得するためにユーザーが再認証される必要がある。ホストされた UI セッション cookie は自動的に期限切れにならないため、ユーザーは認証情報の入力を求められることなく、セッション cookie を使用して再認証できます。ホストされた UI ユーザーをサインアウトしたら、ユーザーをログアウトエンドポイントにリダイレクトします。ここで Amazon Cognito はユーザーのセッション cookie を消去します。

更新トークンを使用すると、ユーザーのセッションをアプリケーション内で長期間維持できます。時間が経つに従って、ユーザーはサインインしている一部のデバイスの認証を解除して、セッションを継続的に更新することを希望する場合があります。1 つのデバイスからユーザーをサインアウトするには、ユーザーの更新トークンを取り消します。ユーザーがすべての認証済みセッションからサインアウトする場合は、GlobalSignOutAPI リクエスト を生成します。アプリケーションは、[すべてのデバイスからサインアウト] などのオプションをユーザーに提供できます。`GlobalSignOut` は、ユーザーの有効な (変更されていない、有効期限が切れていない、取り消されていない) アクセストークンを受け入れます。この API はトークン認証されているため、あるユーザーがそれを使用して別のユーザーのサインアウトを開始することはできません。

ただし、 AWS 認証情報を使用して認証する AdminUserGlobalSignOut API リクエストを生成して、すべてのデバイスから任意のユーザーをサインアウトできます。管理者アプリケーションは AWS 、開発者の認証情報を使用してこの API オペレーションを呼び出し、ユーザープール ID とユーザーのユーザー名をパラメータとして渡す必要があります。`AdminUserGlobalSignOut` API は、ユーザープール内の任意のユーザーをサインアウトできます。

AWS 認証情報またはユーザーのアクセストークンのいずれかで認証できるリクエストの詳細については、「」を参照してくださいAmazon Cognito ユーザープールの認証済みおよび未認証の API オペレーション。

トークンの取り消し

AWS API を使用して、ユーザーの更新トークンを取り消すことができます。更新トークンを取り消すと、その更新トークンによってそれまでに発行されたすべてのアクセストークンが無効になります。ユーザーに発行されたその他の更新トークンは影響を受けません。

> **Note**
>
> JWT トークンは、トークンが作成されたときに割り当てられた署名と有効期限を持つ自己完結型トークンです。取り消されたトークンは、トークンを必要とする Amazon Cognito API

コールでは使用できません。ただし、JWT ライブラリを使用してトークンの署名と有効期限が検証された場合は、引き続き使用が可能です。

トークンの取り消しが有効化されているユーザープールクライアント用の更新トークンは、取り消すことが可能です。トークンの取り消しは、新しいユーザープールクライアントの作成時にデフォルトで有効になります。

トークンの取り消しを有効にする

既存のユーザープールクライアントのトークンを取り消す前に、トークンの取り消しを有効にする必要があります。または AWS API を使用して、 AWS CLI 既存のユーザープールクライアントのトークン失効を有効にできます。これを行うには、`aws cognito-idp describe-user-pool-client` CLI コマンドまたは `DescribeUserPoolClient` API 操作を呼び出して、アプリクライアントから現在の設定を取得します。次に、または `UpdateUserPoolClient` CLI コマンド`aws cognito-idp update-user-pool-client` API オペレーションを呼び出します。アプリクライアントの現在の設定を含め、`EnableTokenRevocation` パラメータを `true` に設定します。

AWS Management Console、、 AWS CLIまたは AWS API を使用して新しいユーザープールクライアントを作成すると、トークンの失効がデフォルトで有効になります。

トークンの取り消しを有効にした後は、Amazon Cognito JSON ウェブトークンに新しいクレームが追加され、アクセストークンと ID トークンに `origin_jti` と `jti` クレームが追加されます。これらのクレームにより、アプリケーションクライアントのアクセストークンと ID トークンのサイズが大きくなります。

トークン失効を有効にしてアプリケーションクライアントを作成または変更するには、CreateUserPoolClientまたは UpdateUserPoolClient API リクエストに次のパラメータを含めます。

```
"EnableTokenRevocation": true
```

トークンを取り消す

CLI コマンドなど、RevokeTokenAPI `aws cognito-idp revoke-token` リクエストを使用して更新トークンを取り消すことができます。エンドポイントの取り消し を使用してトークンを取り消すこともできます。このエンドポイントは、ユーザープールにドメインを追加した後で利用可能になります。取り消しエンドポイントは、Amazon Cognito がホストするドメイン、あるいは独自のカスタムドメインの両方で使用できます。

> **Note**
>
> 更新トークンを取り消すリクエストには、そのトークンを取得するために使用したクライアント ID を含める必要があります。

次は、RevokeToken API リクエストの例の本文です。

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

次は、カスタムドメインのユーザープールの /oauth2/revoke エンドポイントに対する cURL リクエストの例です。

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

RevokeToken オペレーションと /oauth2/revoke エンドポイントには、アプリケーションクライアントにクライアントシークレットがある場合を除き、追加の認証は必要ありません。

JSON Web トークンの検証

これらのステップでは、ユーザープール JSON Web トークン (JWT) の検証を説明します。

トピック

- 前提条件
- でトークンを検証する aws-jwt-verify
- トークンの理解と検査

前提条件

このセクションのタスクは、ライブラリ、SDK、またはソフトウェアフレームワークで既に処理されている可能性があります。 AWS SDKsは、アプリケーションで Amazon Cognito ユーザープール

トークンの処理と管理のためのツールを提供します。 AWS Amplify には、Amazon Cognito トークンを取得および更新する関数が含まれています。

詳細については、次のページを参照してください。

- Amazon Cognito の認証と承認を、ウェブアプリケーションとモバイルアプリケーションに統合する
- SDK を使用した Amazon Cognito ID プロバイダーのコード例 AWS SDKs
- Amplify Dev Center の高度なワークフロー

JSON ウェブトークン (JWT) のデコードと検証用として、多数のライブラリが用意されています。サーバー側の API 処理用にトークンを手動で処理する場合、または他のプログラミング言語を使用している場合は、これらのライブラリが役に立ちます。「OpenID foundation の JWT トークンでの作業のためのライブラリのリスト」を参照してください。

でトークンを検証する aws-jwt-verify

Node.js アプリでは、 AWS は、ユーザーがアプリに渡すトークンのパラメータを検証するためにaws-jwt-verifyライブラリを推奨します。`aws-jwt-verify` を使用すると、1 つ以上のユーザープールについて検証したいクレーム値を `CognitoJwtVerifier` に入力できます。確認できる値には次のものがあります。

- アクセストークンや ID トークンの形式が不正ではなく、期限切れでもなく、有効な署名が付いているもの。
- アクセストークンが、正しいユーザープールとアプリクライアントから取得されたもの。
- アクセストークンクレームに、正しい OAuth 2.0 スコープが含まれているもの。
- アクセストークンと ID トークンに署名したキーが、ユーザープールの JWKS URI の `kid` 署名キーと一致していること。

  JWKS URI には、ユーザーのトークンに署名した秘密鍵に関する公開情報が含まれています。ユーザープールの JWKS URI は、`https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json` で確認できます。

Node.js アプリまたは AWS Lambda オーソライザーで使用できる詳細とコード例については、「」のaws-jwt-verify「」を参照してください GitHub。

トークンの理解と検査

トークン検査をアプリに統合する前に、Amazon Cognito が JWT を組み立てる方法を検討してください。ユーザープールからサンプルトークンを取得します。それらをデコードして詳細に調べて特性を理解し、何をいつ検証するかを決定します。例えば、あるシナリオではグループメンバーシップを検証し、別のシナリオではスコープを調べたい可能性があります。

以下のセクションでは、アプリを準備するときに Amazon Cognito JWT を手動で検査するプロセスについて説明します。

JWT の構造を確認します

JSON ウェブトークン (JWT) は、間に . (ドット) 区切り文字がある 3 つのセクションで構成されます。

[Header] (ヘッダー)

Amazon Cognito がトークンの署名に使用したキー ID、`kid`、および RSA アルゴリズム、`alg`。Amazon Cognito は `RS256` の `alg` でトークン署名します。

ペイロード

トークンクレーム。ID トークンでは、クレームには、ユーザー属性とユーザープール、`iss`、およびアプリクライアント、`aud` に関する情報が含まれます。アクセストークンのペイロードには、スコープ、グループメンバーシップ、ユーザープールが `iss` として含まれ、アプリクライアントは `client_id` として含まれます。

署名

署名は、ヘッダーやペイロードのようにデコード可能な base64 ではありません。JWKS URI で確認できる署名キーとパラメータから派生した RSA256 識別子です。

ヘッダーとペイロードは base64 でエンコードされた JSON です。開始文字 `{` にデコードされる最初の文字 `eyJ` で識別できます。ユーザーが base64 でエンコードされた JWT をアプリに提示し、それが形式 `[JSON Header].[JSON Payload].[Signature]` ではない場合、有効な Amazon Cognito トークンではないため破棄できます。

JWT を検証

JWT 署名は、ヘッダーとペイロードのハッシュされた組み合わせです。Amazon Cognito は、ユーザープールごとに 2 組の RSA 暗号化キーを生成します。1 つの秘密鍵がアクセストークンに署名し、もう 1 つが ID トークンに署名します。

JWT トークンの署名を検証する

1. ID トークンを復号化します。

   OpenID Foundation では、JWT トークンでの作業のためのライブラリのリストも維持されています。

   を使用して AWS Lambda ユーザープール JWTsデコードすることもできます。詳細については、「 を使用した Amazon Cognito JWT トークンのデコードと検証 AWS Lambda」を参照してください。

2. ローカルキー ID (kid) とパブリック kid を比較します。

   a. ユーザープール用に、対応するパブリック JSON Web キー (JWK) をダウンロードして保存します。これは、JSON Web キーセット (JWKS) の一部として提供されており、環境に合わせて次のように jwks_uri URL を構築することで、その場所を特定できます。

   ```
   https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
   ```

   JWK と JWK セットの詳細については、「JSON Web Key (JWK)」を参照してください。

   > **Note**
   >
   > Amazon Cognito は、ユーザープール内で署名キーをローテーションする可能性があります。ベストプラクティスとして、kid をキャッシュキーとして使用して、アプリに公開鍵をキャッシュし、定期的にキャッシュを更新してください。アプリが受け取るトークンの kid をキャッシュと比較します。
   > 正しい発行者で kid が異なるトークンを受け取った場合、Amazon Cognito が署名キーをローテーションした可能性があります。ユーザープール jwks_uri エンドポイントのキャッシュを更新します。

   以下は、サンプル jwks.json ファイルです。

   ```
   {
    "keys": [{
     "kid": "1234example=",
     "alg": "RS256",
     "kty": "RSA",
     "e": "AQAB",
   ```

```
  "n": "1234567890",
  "use": "sig"
 }, {
  "kid": "5678example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "987654321",
  "use": "sig"
 }]
}
```

Key ID (**kid**)

kid は、トークンの JSON ウェブ署名 (JWS) をセキュア化するために使用されたキーを示すヒントです。

Algorithm (**alg**)

alg ヘッダーパラメータは、ID トークンをセキュア化するために使用される暗号化アルゴリズムを表します。ユーザープールは、SHA-256 による RSA 署名である RS256 暗号化アルゴリズムを使用します。RSA の詳細については、「RSA cryptography」を参照してください。

Key type (**kty**)

kty パラメータは、この例の「RSA」など、キーで使用される暗号化アルゴリズムファミリーを特定します。

RSA exponent (**e**)

e パラメータには、RSA パブリックキーの指数値が含まれます。これは、Base64urlUInt でエンコードされた値として表されます。

RSA modulus (**n**)

n パラメータには、RSA パブリックキーのモジュラス値が含まれます。これは、Base64urlUInt でエンコードされた値として表されます。

Use (**use**)

use パラメータは、パブリックキーの用途を表します。この例では、use 値の sig が署名を表しています。

b. JWT の kid に一致する kid のパブリック JSON ウェブキーを検索します。

3. JWT ライブラリを使用して、発行者の署名をトークン内の署名と比較します。発行者の署名は、トークンと一致する jwks.json の kid の公開キー (RSA モジュラス "n") から取得されます。kid。まず、JWK を PEM 形式に変換する必要がある場合があります。次の例には JWT と JWK があり、Node.js ライブラリの jsonwebtoken を使用して JWT 署名を検証します。

   Node.js

```
var jwt = require('jsonwebtoken');
var jwkToPem = require('jwk-to-pem');
var pem = jwkToPem(jwk);
jwt.verify(token, pem, { algorithms: ['RS256'] }, function(err, decodedToken) {
});
```

クレームを検証する

JWT クレームを検証する

1. 次のいずれかの方法で、トークンの有効期限が切れていないことを確認します。

   a. トークンをデコードし、exp クレームを現在の時刻と比較します。

   b. アクセストークンに aws.cognito.signin.user.adminクレームが含まれている場合は、 などの API にリクエストを送信しますGetUser。アクセストークンで承認する API リクエストは、トークンの有効期限が切れているとエラーを返します。

   c. UserInfo エンドポイント へのリクエストでアクセストークンを提示します。トークンの有効期限が切れている場合、リクエストはエラーを返します。

2. ID トークンの aud クレームとアクセストークンの client_id クレームは、Amazon Cognito ユーザープールで作成されたアプリクライアント ID と一致する必要があります。

3. 発行者 (iss) のクレームは、ユーザープールと一致する必要があります。例えば、us-east-1 リージョンで作成されたユーザープールには、以下の iss 値があります。

   https://cognito-idp.us-east-1.amazonaws.com/*<userpoolID>*.

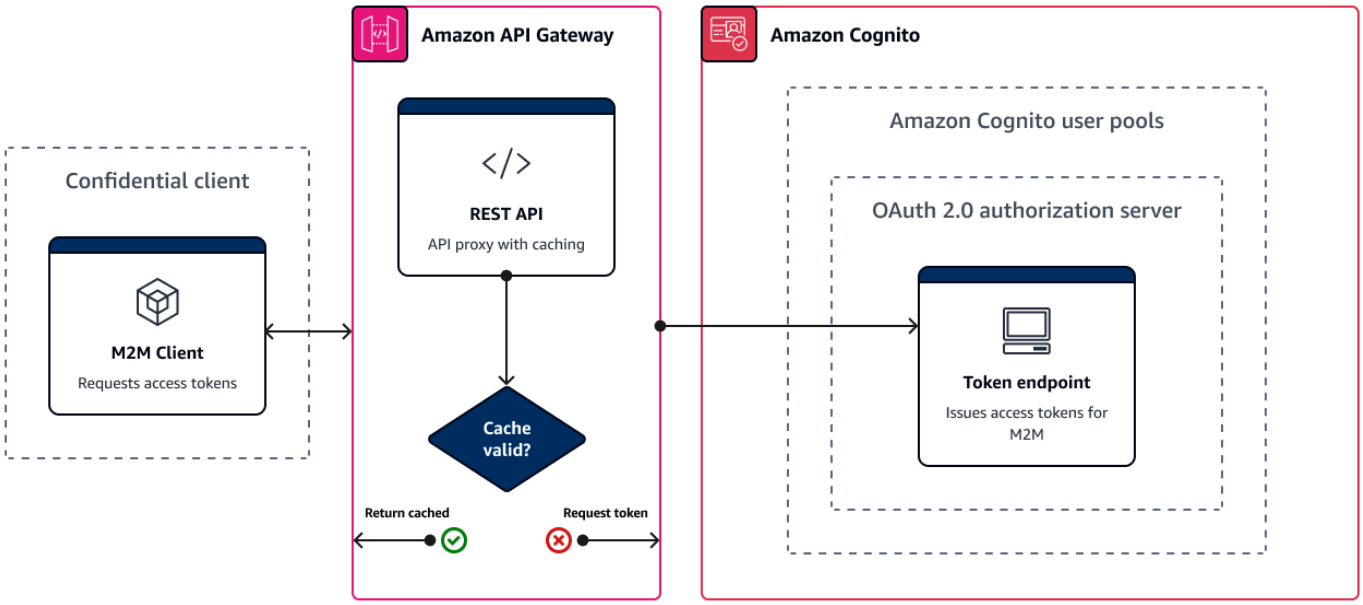
4. token_use クレームをチェックします。

   - Web API オペレーションでアクセストークンのみを受け入れている場合は、その値を access にする必要があります。
   - ID トークンのみを使用している場合、その値は id にする必要があります。

- ID とアクセストークンのいずれも使用している場合、token_use クレームは、id または access になります。

これで、トークン内のクレームを信頼できるようになりました。

キャッシュトークン



新しい JSON ウェブトークン (JWT) を取得するたびに、アプリは次のいずれかのリクエストを正常に完了する必要があります。

- トークンエンドポイント からクライアント認証情報または認証コードの付与をリクエストする。
- ホストされている UI から暗黙的付与をリクエストする。
- などの Amazon Cognito API リクエストでローカルユーザーを認証しますInitiateAuth。

トークンの有効期限が分単位、時間単位、または日単位になるようにユーザープールを設定できます。アプリのパフォーマンスと可用性を確保するには、有効期限が切れるまで Amazon Cognito トークンを使用してから、新しいトークンを取得してください。アプリ用に構築したキャッシュソリューションはトークンを利用可能な状態に保ち、リクエストレートが高すぎる場合に Amazon Cognito がリクエストを拒否するのを防ぎます。クライアント側のアプリは、トークンをメモリキャッシュに保存する必要があります。サーバー側アプリでは、暗号化されたキャッシュメカニズムを追加してトークンを保存できます。

ユーザープールが大量のユーザーまたは machine-to-machine アクティビティを生成すると、Amazon Cognito がトークンのリクエスト数に設定できる制限が発生する可能性があります。Amazon Cognito エンドポイントへのリクエスト数を減らすには、認証データを安全に保存して再利用するか、エクスポネンシャルバックオフと再試行を実装することができます。

認証データは 2 つのクラスのエンドポイントから取得されます。Amazon Cognito OAuth 2.0 エンドポイントには、クライアントの認証情報とホストされた UI 認証コードのリクエストを処理するトークンエンドポイントが含まれています。サービスエンドポイントは、`InitiateAuth` や `RespondToAuthChallenge` のようなユーザープール API リクエストに応答します。各タイプのリクエストには独自の制限があります。制限事項の詳細については、「Amazon Cognito のクォータ」を参照してください。

Amazon API Gateway を使用した machine-to-machine アクセストークンのキャッシュ

API Gateway トークンキャッシュを使用すると、Amazon Cognito OAuth エンドポイントのデフォルトのリクエストレートクォータを超えるイベントに応じてアプリをスケールさせることができます。

アクセストークンをキャッシュして、キャッシュされたトークンの有効期限が切れた場合にのみ、アプリが新しいアクセストークンをリクエストするようにできます。それ以外の場合、キャッシュエンドポイントはキャッシュからトークンを返します。これにより、Amazon Cognito API エンドポイントへの追加呼び出しを防ぐことができます。Amazon API Gateway を トークンエンドポイント へのプロキシとして使用する場合、API はリクエストクォータに影響するリクエストの大半に応答し、レート制限によるリクエストの失敗を防ぎます。

次の API Gateway ベースのソリューションでは、トークンキャッシュの低レイテンシー、ローコード、ノーコード実装が提供されます。API Gateway API は転送中に暗号化され、オプションで保存時にも暗号化されます。API Gateway キャッシュは、OAuth 2.0 クライアント認証情報付与 に最適です。これは、 machine-to-machine およびマイクロサービスセッションを承認するためのアクセストークンを生成する、頻繁に大量の付与タイプです。マイクロサービスが水平方向にスケールするトラフィックの急増などのイベントでは、ユーザープールまたはアプリクライアントの AWS リクエストレート制限を超えるボリュームで同じクライアント認証情報を使用する多くのシステムが発生する可能性があります。このようなシナリオでは、アプリの可用性と低レイテンシーを維持するために、キャッシュソリューションがベストプラクティスです。

このソリューションでは、API にキャッシュを定義して、アプリでリクエストする OAuth スコープとアプリクライアントの組み合わせごとに個別のアクセストークンを保存します。アプリがキャッシュキーと一致するリクエストを行うと、API はキャッシュキーと一致する最初のリクエストに対して Amazon Cognito が発行したアクセストークンを返します。キャッシュキーの有効期限が切れる

と、API はリクエストをトークンエンドポイントに転送し、新しいアクセストークンをキャッシュします。

> **Note**
>
> キャッシュキーの有効期間は、アプリクライアントのアクセストークン有効期間よりも短くする必要があります。

キャッシュキーは、`scope` URL パラメータでリクエストした OAuth スコープとリクエストの `Authorization` ヘッダーの組み合わせです。`Authorization` ヘッダーには、アプリのクライアント ID とクライアントシークレットが含まれます。このソリューションを実装するために、アプリに追加のロジックを実装する必要はありません。ユーザープールトークンエンドポイントへのパスを変更するには、設定を更新するだけで済みます。

ElastiCache for Redis を使用してトークンキャッシュを実装することもできます。 AWS Identity and Access Management (IAM) ポリシーによるきめ細かなコントロールが必要な場合は、Amazon DynamoDB キャッシュをご検討ください。

> **Note**
>
> API Gateway でのキャッシュには追加料金がかかります。詳細については、料金表を参照してください。

API Gateway でキャッシュプロキシをセットアップする方法

1. API Gateway コンソールを開き、REST API を作成します。
2. [Resources] (リソース) で、POST メソッドを作成します。
   a. HTTP [Integration type] (統合タイプ) を選択してください。
   b. [Use HTTP proxy integration] (HTTP プロキシ統合の使用) を選択します。
   c. `https://<your user pool domain>/oauth2/token` の [Endpoint URL] (エンドポイント URL) を入力します。
3. [Resources] (リソース) で、キャッシュキーを設定します。
   a. POST メソッドの [Method request] (メソッドリクエスト) を編集します。
   b. `scope` パラメータと `Authorization` ヘッダーをキャッシュキーとして設定します。

i. クエリ文字列を [URL query string parameters] (URL クエリ文字列パラメータ) に追加し、`scope` 文字列の [Caching] (キャッシュ) を選択します。

ii. [HTTP request headers] (HTTP リクエストヘッダー) にヘッダーを追加し、`Authorization` ヘッダーの [Caching] (キャッシュ) を選択します。

4. [Stages] (ステージ) で、キャッシュを設定します。

   a. 変更するステージを選択します。

   b. [Settings] (設定) で、[Enable API cache] (API キャッシュを有効にする) を選択します。

   c. [Cache capacity] (キャッシュ容量) を選択します。

   d. 3600 秒以上のキャッシュ time-to-live (TTL) を選択します。

   e. [承認を必須にする] チェックボックスをオフにします。

5. [Stages] (ステージ) で、[Invoke URL] (URL を呼び出す) に注目します。

6. ユーザープールの `/oauth2/token` エンドポイントの代わりに、API の Invoke URL (URL を呼び出す) にトークンリクエストを POST するようにアプリを更新します。

ユーザープール認証に成功した後のリソースへのアクセス

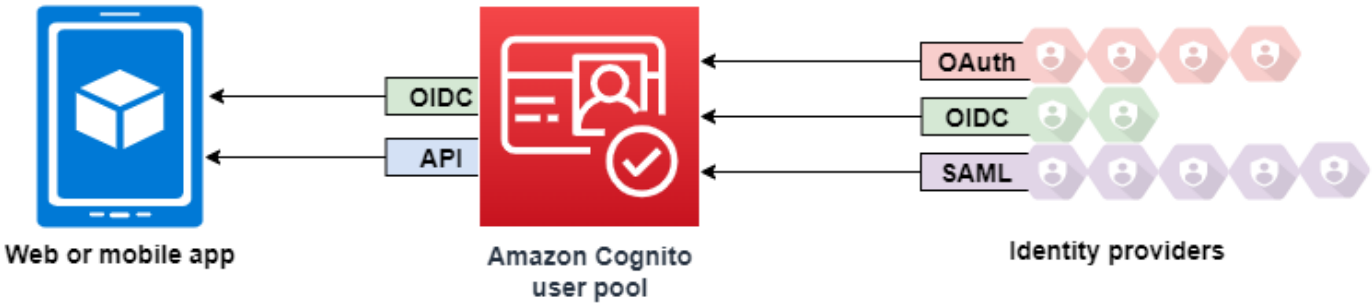
アプリユーザーは、ユーザープールから直接サインインすることも、サードパーティー ID プロバイダー (IdP) を介してフェデレーションすることもできます。ユーザープールは、Facebook、Google、Amazon、Apple を介したソーシャルサインイン、および OpenID Connect (OIDC) と SAML から返されるトークンの処理のオーバーヘッドを管理します IdPs。詳細については、「ユーザープールでのトークンの使用」を参照してください。

認証が正常に行われると、アプリケーションが Amazon Cognito からユーザープールトークンを受け取ります。ユーザープールトークンを使用して、次のことができます。

- Amazon DynamoDB や Amazon S3 AWS のサービス などの のアプリケーションリソースのリクエストを許可する AWS 認証情報を取得します。
- 一時的な取り消し可能な認証証明を提供します。
- アプリのユーザープロファイルに ID データを入力します。
- ユーザープールディレクトリでサインインしたユーザーのプロファイルへの変更を承認します。
- アクセストークンを使用してユーザー情報に対するリクエストを承認します。
- アクセストークンを使用して、アクセスで保護された外部 APIs背後にあるデータへのリクエストを承認します。

- Amazon Verified Permissions を使用して、クライアントまたはサーバーに保存されているアプリケーションアセットへのアクセスを許可します。

詳細については、「ユーザープール認証フロー」および「ユーザープールでのトークンの使用」を参照してください。



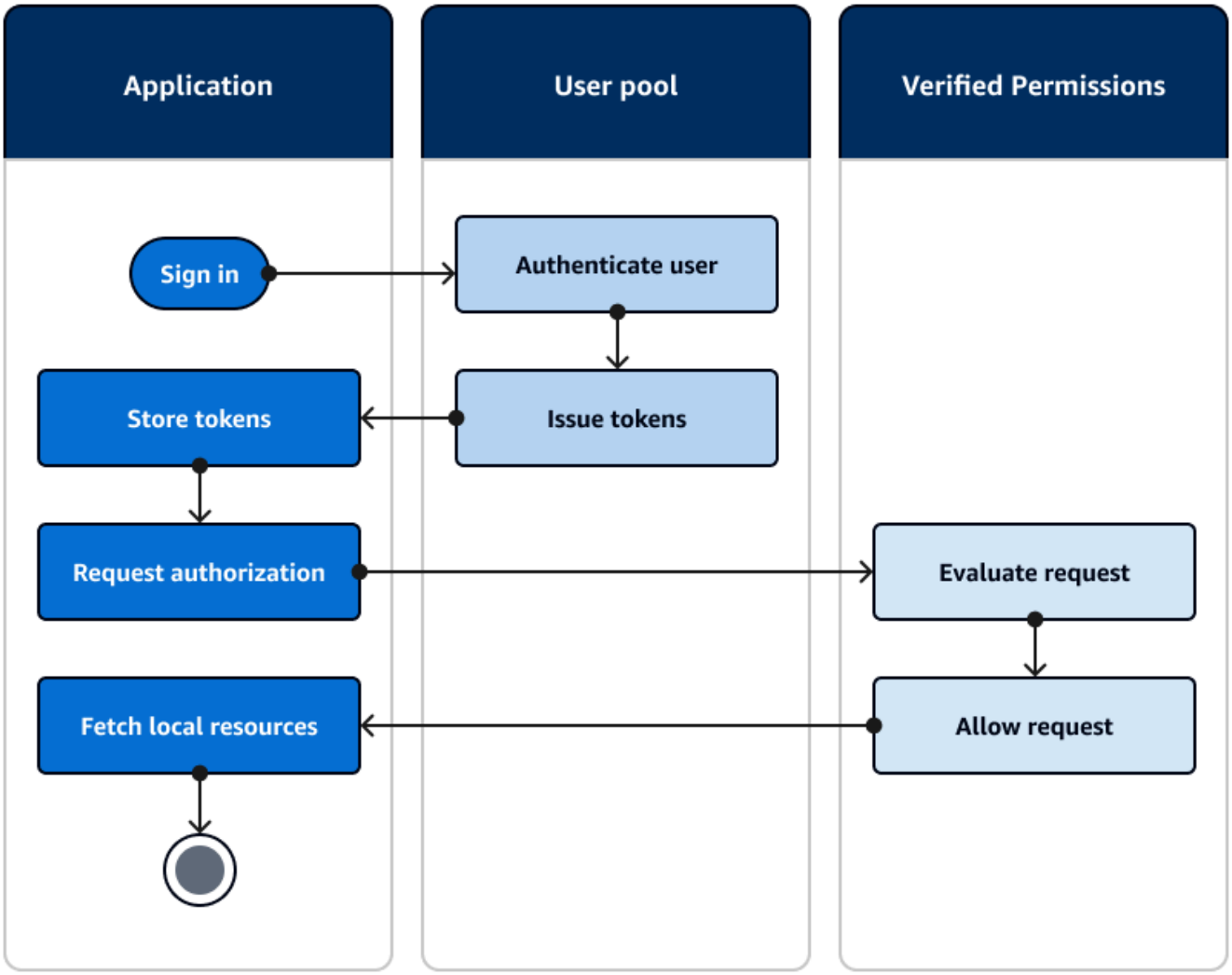
トピック

- Amazon Verified Permissions によるクライアントまたはサーバーリソースへのアクセスの許可
- サインイン後に API Gateway でリソースにアクセスする
- サインイン後の ID プール AWS のサービス を使用した へのアクセス

Amazon Verified Permissions によるクライアントまたはサーバーリソースへのアクセスの許可

アプリは、サインインしたユーザーから Amazon Verified Permissions にトークンを渡すことができます。Verified Permissions は、構築したカスタムアプリケーション向けのスケーラブルできめ細かなアクセス許可の管理および承認サービスです。Amazon Cognito ユーザープールは、Verified Permissions ポリシーストアの ID ソースにすることができます。Verified Permissions は、プリンシパルからのリクエストされたアクションとリソース`premium_badge.png`、およびユーザープールトークンの属性`GetPhoto`について、承認の決定を行います。

次の図は、承認リクエストでユーザーのトークンを Verified Permissions に渡す方法を示しています。



Amazon Verified Permissions の使用を開始する

ユーザープールを Verified Permissions と統合すると、すべての Amazon Cognito アプリに対するきめ細かな認証を一元的に取得できます。これにより、すべてのアプリケーション間でコーディングとレプリケートを行う必要があるきめ細かなセキュリティロジックが不要になります。Verified Permissions による認可の詳細については、「」を参照してくださいAmazon Verified Permissions による承認。

Verified Permissions 認証リクエストには AWS 認証情報が必要です。認証リクエストに認証情報を安全に適用するには、以下の手法の一部を実装できます。

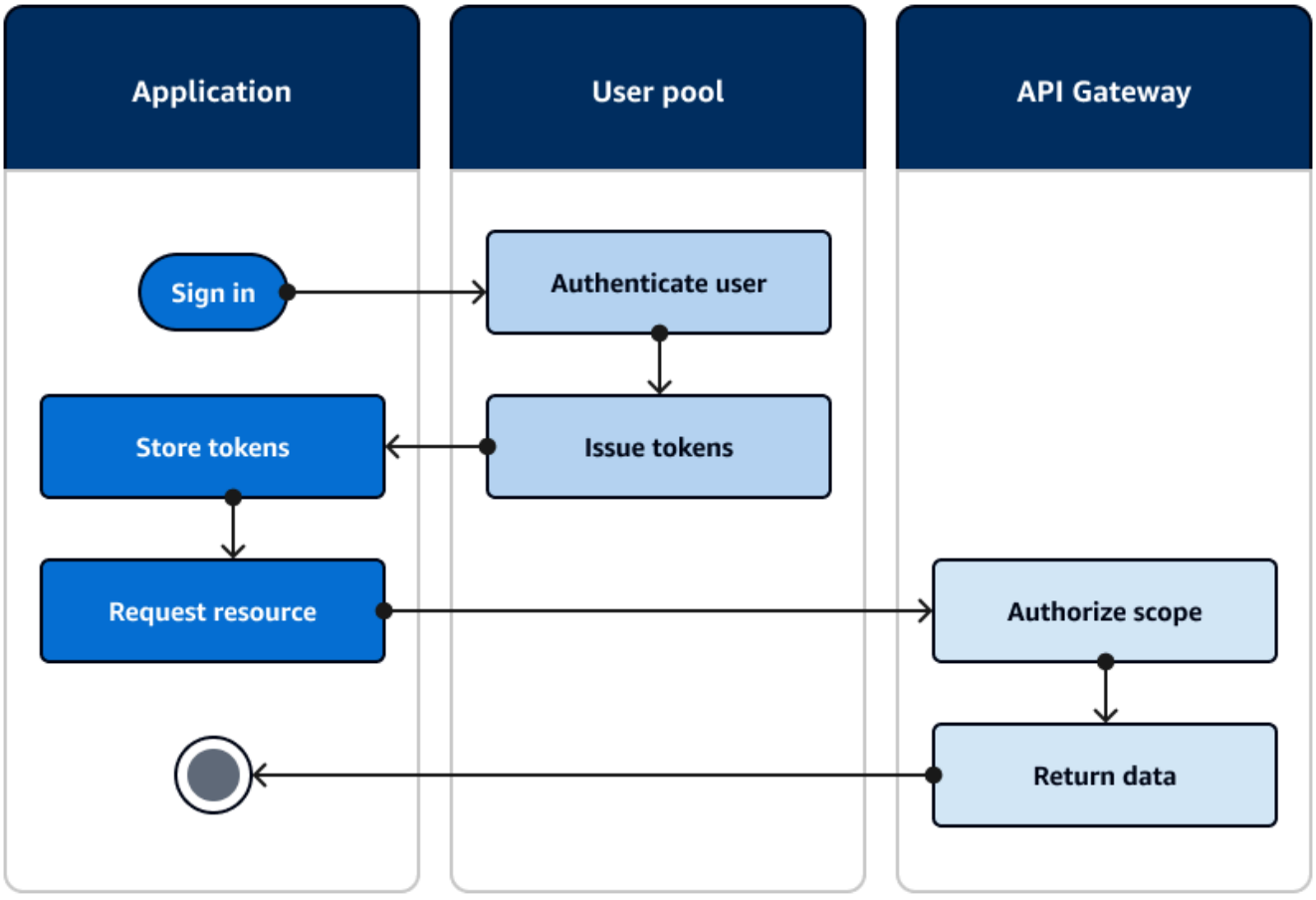
- サーバーのバックエンドにシークレットを保存できるウェブアプリケーションを運用します。

- 認証された ID プールの認証情報を取得します。
- access-token-authorized API を使用してユーザーリクエストをプロキシし、リクエストに認証情報を追加します AWS 。

サインイン後に API Gateway でリソースにアクセスする

Amazon Cognito ユーザープールトークンの一般的な用途は、API Gateway REST API へのリクエストを承認することです。アクセストークンの OAuth 2.0 スコープは、 HTTP GETのようなメソッドとパスを承認できます/app_assets。ID トークンは API への汎用認証として機能し、ユーザー属性をバックエンドサービスに渡すことができます。API Gateway には、HTTP APIs 用の JWT オーソライザーや、より詳細なロジックを適用できる Lambda オーソライザーなどの追加のカスタム認証オプションがあります。

次の図は、アクセストークンの OAuth 2.0 スコープを使用して REST API にアクセスしようとしているアプリケーションを示しています。



アプリは、認証されたセッションからトークンを収集し、ベアラートークンとしてリクエストの `Authorization`ヘッダーに追加する必要があります。API、パス、およびメソッド用に設定したオーソライザーを設定して、トークンの内容を評価します。API Gateway は、リクエストがオーソライザーに設定した条件に一致する場合にのみデータを返します。

API Gateway API がアプリケーションからのアクセスを承認する方法には、次のようなものがあります。

- アクセストークンには、正しい OAuth 2.0 スコープが含まれています。REST API の Amazon Cognito ユーザープールオーソライザーは、エントリに対する障壁が低い一般的な実装です。このタイプのオーソライザーへのリクエストの本文、クエリ文字列パラメータ、ヘッダーを評価することもできます。
- ID トークンは有効であり、有効期限が切れていません。ID トークンを Amazon Cognito オーソライザーに渡すと、アプリケーションサーバーで ID トークンの内容をさらに検証できます。
- アクセストークンまたは ID トークンのグループ、クレーム、属性、またはロールは、Lambda 関数で定義した要件を満たします。Lambda オーソライザーは、リクエストヘッダー内のトークンを解析し、承認の決定について評価します。関数でカスタムロジックを構築するか、Amazon Verified Permissions に API リクエストを行うことができます。

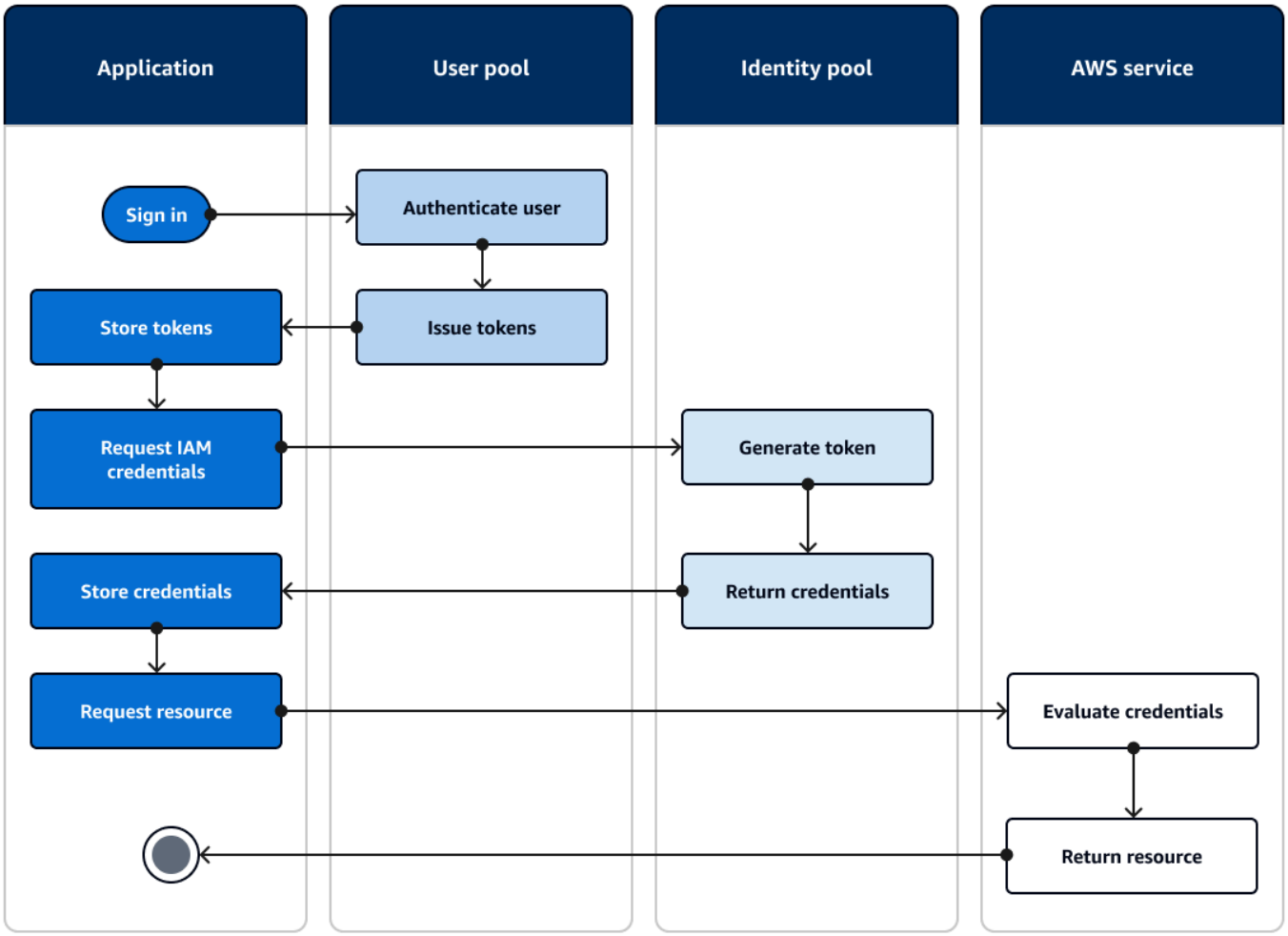
ユーザープールからのトークンを使用して AWS AppSync GraphQL API へのリクエストを承認することもできます。

サインイン後の ID プール AWS のサービス を使用した へのアクセス

ユーザーがユーザープールでサインインすると、ID プールから発行された一時的な API 認証情報 AWS のサービス を使用して にアクセスできます。

ウェブまたはモバイルアプリは、ユーザープールからトークンを受け取ります。ユーザープールを ID プールの ID プロバイダーとして設定すると、ID プールはトークンを一時的な AWS 認証情報と交換します。これらの認証情報の範囲は、IAM ロールとそのポリシーに限定され、ユーザーが限定された AWS リソースセットにアクセスできるようにします。詳細については、「ID プール (フェデレーティッドアイデンティティ) の認証フロー」を参照してください。

次の図は、アプリケーションがユーザープールでサインインし、ID プールの認証情報を取得し、 からアセットをリクエストする方法を示しています AWS のサービス。



ID プールの認証情報を使用して、次のことができます。

- ユーザー自身の認証情報を使用して、Amazon Verified Permissions にきめ細かな認証リクエストを行います。
- IAM との接続を許可する Amazon API Gateway REST API または AWS AppSync GraphQL API に接続します。
- IAM との接続を許可する Amazon DynamoDB や Amazon RDS などのデータベースバックエンドに接続します。
- Amazon S3 バケットからアプリケーションアセットを取得します。
- Amazon WorkSpaces 仮想デスクトップでセッションを開始します。

ID プールは、ユーザープールとの認証されたセッション内でのみ動作するわけではありません。また、サードパーティーの ID プロバイダーから直接認証を受け入れ、認証されていないゲストユーザーの認証情報を生成することもできます。

ID プールとユーザープールグループを使用して AWS リソースへのアクセスを制御する方法の詳細については、ユーザープールにグループを追加する「」および「」を参照してくださいロールベースアクセスコントロールの使用。また、ID プールと の詳細については AWS Identity and Access Management、「」を参照してくださいID プールの概念。

を使用したユーザープールのセットアップ AWS Management Console

Amazon Cognito ユーザープールを作成し、クライアントアプリごとに [User Pool ID] (ユーザープール ID) と [App Client ID] (アプリクライアント ID) をメモしておきます。ユーザープール作成の詳細については、「ユーザープールの開始方法」を参照してください。

を使用した ID プールのセットアップ AWS Management Console

次の手順では、 を使用して ID プールを 1 つ以上のユーザープールおよびクライアントアプリケーションと AWS Management Console 統合する方法について説明します。

Amazon Cognito のユーザープール ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Amazon Cognito ユーザープール] を選択します。
5. ユーザープール ID とアプリクライアント ID を入力します。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

    a. 認証済みロール を設定したときに設定したデフォルトロールをその IdP のユーザーに付与することも、ルール でロールを選択することもできます。Amazon Cognito ユーザープール IdP では、トークンの優先ロールクレームを持つロールを選択することもできます。`cognito:preferred_role` クレームの詳細については、「グループへの優先順位の値の割り当て」を参照してください。

        i. ルールを使用してロールを選択 を選択した場合は、ユーザーの認証からソースクレーム、クレームをルールと比較するために使用するオペレーター、このロールの選択に一

致する値、ロールの割り当てが一致するときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

ii. トークン で preferred_role クレームを持つロールを選択を選択した場合、Amazon Cognito はユーザーの`cognito:preferred_role`クレームでロールの認証情報を発行します。優先ロールクレームが存在しない場合、Amazon Cognito はロール解決に基づいて認証情報を発行します。

b. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

- プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
- sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。
- プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

ユーザープールを ID プールと統合する

アプリユーザーが認証されると、認証情報プロバイダーのログインマップにユーザーのアイデンティティトークンを追加します。プロバイダー名は、Amazon Cognito ユーザープール ID に応じて異なります。次の構造になります。

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

<region> の値は、ユーザープール ID から取得できます。例えば、ユーザープール ID が の場合`us-east-1_EXAMPLE1`、*<region>* は ですus-east-1。ユーザープール ID が の場合`us-west-2_EXAMPLE2`、*<region>* は ですus-west-2。

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
```

```
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');

   // Add the User's Id Token to the Cognito credentials login map.
   AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    }
   });
  }
 });
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
```

```
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

Amazon Cognito ユーザープールのセキュリティ機能を使用する

多要素認証 (MFA) をユーザープールに追加して、ユーザーの ID を保護することができます。MFA は、ユーザープールがユーザー名とパスワードのみに依存しないように 2 番目の認証要素を追加します。ユーザーにサインインする際の第 2 の要素として、SMS テキストメッセージまたは時間ベースのワンタイムパスワード (TOTP) を使用することができます。リスクベースのモデルでアダプティブ認証を使用して、別の認証要素が必要なときを予測することもできます。ユーザープールアドバンストセキュリティ機能には、アダプティブ認証および侵害された認証情報に対する保護が含まれます。

トピック

- ユーザープールに MFA を追加します
- ユーザープールにアドバンストセキュリティを追加する
- AWS WAF ウェブ ACL とユーザープールの関連付け
- ユーザープールの大文字と小文字の区別
- ユーザープールの削除保護
- ユーザー存在エラー応答の管理

ユーザープールに MFA を追加します

多要素認証 (MFA) により、アプリのセキュリティが向上します。これは、ユーザー名とパスワードという知っていることの認証要素に、持っていることの認証要素を追加するものです。ユーザーにサインインするときの 2 番目の要素としては、SMS テキストメッセージまたは時間ベースのワンタイムパスワード (TOTP) を選択することができます。

> **Note**
>
> 新しいユーザーがアプリに初めてサインインするときに、Amazon Cognito は OAuth 2.0 トークンを発行します。これは、ユーザープールに MFA が必要な場合でも同様です。ユーザーが初めてサインインするときの 2 番目の認証要因は、Amazon Cognito が送信する検証メッセージの確認です。ユーザープールに MFA が必要な場合、Amazon Cognito は、最初のログイン試行後の各サインイン試行で使用する追加のサインイン係数を登録するようユーザーに要求します。

アダプティブ認証では、リスクレベルの上昇に対応して、2 番目の要素認証を要求するようにユーザープールを設定できます。アダプティブ認証をユーザープールに追加するには、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

ユーザープールの MFA が `required` に設定されている場合は、すべてのユーザーはサインインするために MFA を完了する必要があります。サインインするためには、各ユーザーには、少なくとも 1 つの MFA 要素 (SMS や TOTP のセットアップなど) が必要です。MFA を `required` に設定した場合は、ユーザープールがサインインを許可するように、ユーザーオンボーディングに MFA セットアップを含める必要があります。

SMS を MFA 要素として有効化する場合は、ユーザーに電話番号の提供を求め、それらの検証をサインアップ時に行うことができます。MFA を `required` に設定し、要素として SMS のみをサポートする場合は、ユーザーに電話番号が必要になります。電話番号がないユーザーは、サインインする前に、プロファイルに電話番号を追加するためのサポートが必要です。SMS MFA には、未検証の電話番号を使用できます。これらの数字は、MFA が成功した後、検証済みステータスを受け取ります。

MFA を必須に設定し、サポートされている検証方法として SMS と TOTP を有効化した場合、Amazon Cognito は電話番号のない新規ユーザーに TOTP MFA を設定するよう促します。MFA を必須に設定し、有効化した唯一の MFA メソッドが TOTP の場合、Amazon Cognito は、2 回目のサインイン時に、すべての新規ユーザーに TOTP MFA を設定するようにプロンプトを表示し

ます。Amazon Cognito は、 InitiateAuthおよび AdminInitiateAuth API オペレーションに応答して TOTP MFA を設定するチャレンジを生成します。

MFA を必須に設定すると、ホスト UI はユーザーに MFA の設定を求めるプロンプトを表示します。ユーザープールで MFA をオプションに設定すると、ホスト UI でユーザーにプロンプトは表示されません。オプションの MFA を使用するには、ユーザーに MFA を設定するかどうかを選択するようプロンプトを表示し、API 入力を案内して追加のサインイン要素を確認するインターフェースをアプリに構築する必要があります。

MFA コードの提示に 5 回失敗すると、Amazon Cognito は ユーザープール認証フロー で説明した指数関数的タイムアウトロックアウトプロセスを開始します。

トピック

- 前提条件
- 多要素認証の設定
- SMS テキストメッセージ MFA
- TOTP ソフトウェアトークン MFA

前提条件

MFA を設定する前に、次の点を考慮します。

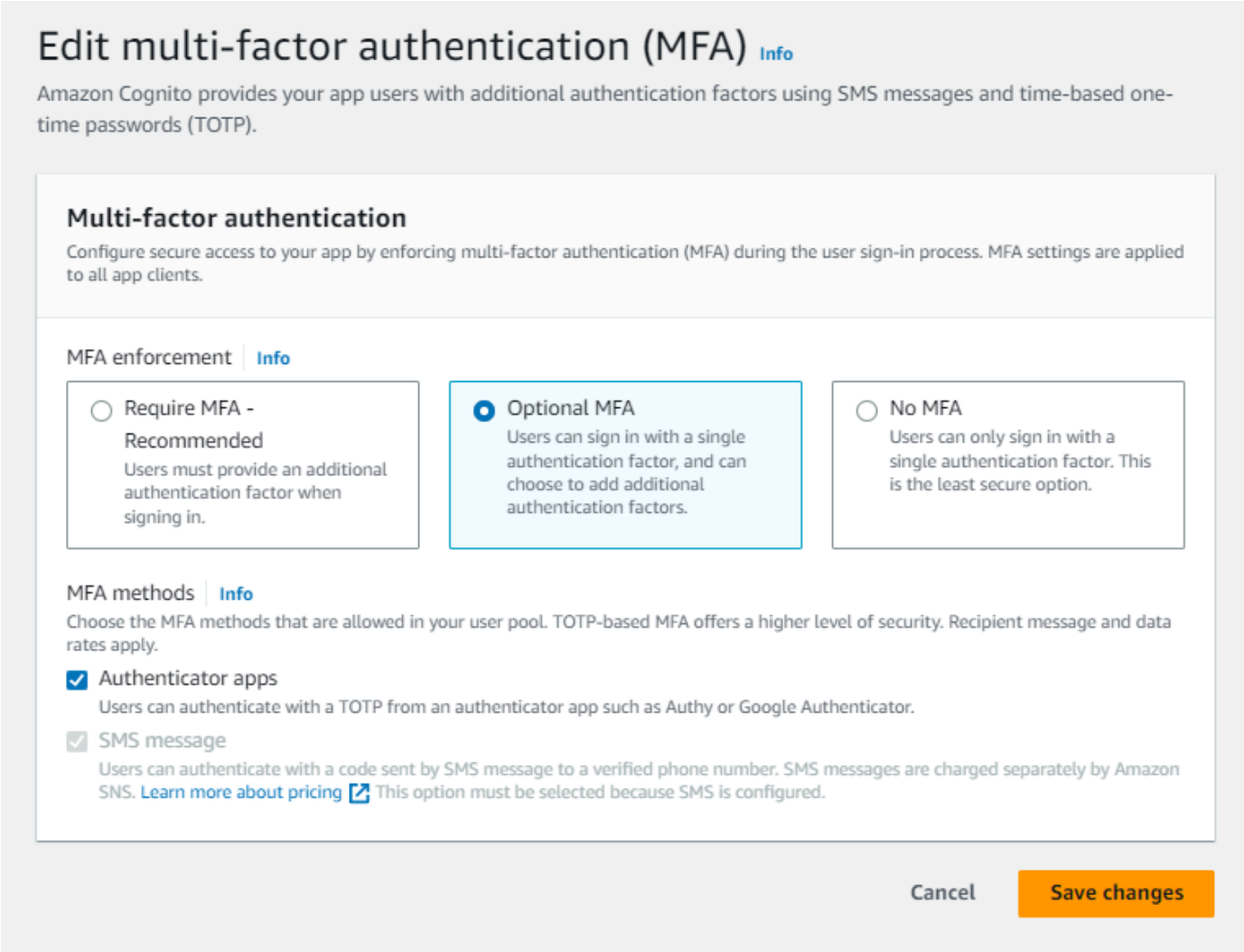
- ユーザープールで MFA を有効化し、SMS テキストメッセージを第 2 要素として選択する場合は、Amazon Cognito で検証していない電話番号属性に SMS メッセージを送信できます。ユーザーが SMS MFA を完了すると、Amazon Cognito は、`phone_number_verified` 属性 を `true` に設定します。
- アカウントがユーザープールの Amazon Simple Notification Service (Amazon SNS) リソース AWS リージョン を含む の SMS サンドボックスにある場合は、SMS メッセージを送信する前に Amazon SNS で電話番号を確認する必要があります。詳細については、「Amazon Cognito ユーザープール用の SMS メッセージ設定」を参照してください。
- アドバンストセキュリティ機能には MFA が有効化され、Amazon Cognito ユーザープールコンソールでオプションとして設定されている必要があります。詳細については、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

多要素認証の設定

MFA は、Amazon Cognito コンソールで設定できます。

Amazon Cognito コンソールで MFA を設定する

1. Amazon Cognito コンソールにサインインします。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [Sign-in experience] (サインインエクスペリエンス) タブを選択します。[Multi-factor authentication] (多要素認証) を探し、[Edit] (編集) を選択します。
5. ユーザープールで使用する [MFA enforcement] (MFA 強制実行) 方法を選択します。



   a. MFA が必要。ユーザープール内のすべてのユーザーは、追加の SMS コードまたはタイムベースドワンタイムパスワード (TOTP) 要素でサインインする必要があります。
   b. MFA のオプション - ユーザーに追加のサインイン要素を登録するオプションを与えても、MFA を設定していないユーザーにはサインインを許可することができます。アダプティブ認証を使用している場合は、このオプションを選択してください。アダプティブ認証

の詳細については、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

c. MFA なし。ユーザーは、サインイン要素を追加登録することはできません。

6. アプリケーションでサポートする [MFA methods] (MFA メソッド) を選択します。2 番目の要素として SMS メッセージまたは TOTP 生成認証システムアプリケーションを設定できます。アカウントの回復が SMS メッセージを使用できるように、TOTP ベースの MFA を実装することをお勧めします。
7. 2 番目の要素として SMS テキストメッセージを使用していて、SMS メッセージ用の Amazon Simple Notification Service (Amazon SNS) で使用する IAM ロールを設定していない場合は、コンソールで作成できます。ユーザープールの [Messaging] (メッセージング) タブで、[SMS] を見つけ、[Edit] (編集) を選択します。Amazon Cognito がユーザーに SMS メッセージを送信することを許可する既存のロールを使用することもできます。詳細については、「IAM ロール」を参照してください。
8. [Save changes] (変更の保存) をクリックします。

SMS テキストメッセージ MFA

MFA が有効になった状態でユーザーがサインインするときは、最初にそのユーザー名とパスワードを入力して送信します。クライアントのアプリケーションは、認証コードがどこに送信されたか示す `getMFA` 応答を受信します。クライアントアプリは、コードがどこにあるか (コードが送信された電話番号など) をユーザーに示す必要があります。次に、コードを入力するためのフォームが提供されます。最後に、クライアントアプリがコードを送信して、サインインプロセスを完了します。送信先はマスクされ、電話番号の末尾 4 桁を除くすべてが非表示になります。アプリケーションが Amazon Cognito でホストされる UI を使用している場合は、ユーザーが MFA コードを入力するためのページを表示します。

SMS テキストメッセージの認証コードは、アプリクライアントに設定された認証フローセッション持続期間の間有効です。

[App clients and analytics] (アプリクライアントと分析) でアプリクライアントを変更するときに、[App integration] (アプリ統合) タブの Amazon Cognito コンソールで認証フローセッションの期間を設定します。`CreateUserPoolClient` または `UpdateUserPoolClient` API リクエストで認証フローセッション持続期間を設定することも可能です。詳細については、「ユーザープール認証フロー」を参照してください。

ユーザーが、SMS テキストメッセージの MFA コードが送られた自身のデバイスにアクセスできない場合、カスタマーサービスのオフィスからのサポートをリクエストする必要があります。必要な

AWS アカウント アクセス許可を持つ管理者は、ユーザーの電話番号を変更できますが、 AWS CLI または API を介してのみ変更できます。

ユーザーが SMS テキストメッセージ MFA フローで正常に処理されると、その電話番号も検証済みとしてマークされます。

> **Note**
>
> MFA の SMS は、個別に課金されます。(メールアドレスへの検証コードの送信には料金はかかりません。) Amazon SNS の料金については、「Worldwide SMS Pricing」を参照してください。SMS メッセージを利用可能な国の最新のリストについては、「サポートされるリージョンと国」を参照してください。

> **Important**
>
> 電話番号の検証と SMS テキストメッセージ MFA のために SMS メッセージが送信されることを確実するには、Amazon SNS の使用制限の引き上げをリクエストする必要があります。Amazon Cognito は、ユーザーへの SMS メッセージの送信に Amazon SNS を使用します。Amazon SNS が配信する SMS メッセージの数は、使用制限の対象になります。支出制限は AWS アカウントと個々のメッセージに対して指定でき、この制限は SMS メッセージを送信するコストにのみ適用されます。
> アカウントごとのデフォルトの使用制限 (指定しない場合) は、1.00 USD/月です。制限を引き上げる場合は、 AWS Support センターで SNS 制限引き上げケースを送信してください。[New limit value] (新しい制限値) に、必要な月ごとの使用制限を入力します。[Use Case Description] (ユースケースの説明) フィールドで、月ごとの SMS 使用制限の引き上げをリクエストしていることを説明します。

MFA をユーザープールに追加するには、「ユーザープールに MFA を追加します」を参照してください。ユーザープール内の Amazon SNS を使用した SMS メッセージの詳細については、「」を参照してくださいAmazon Cognito ユーザープール用の SMS メッセージ設定。

TOTP ソフトウェアトークン MFA

ユーザープールで TOTP ソフトウェアトークン MFA を設定すると、ユーザーはユーザー名とパスワードでサインインし、TOTP を使用して認証を完了します。ユーザーがユーザー名とパスワードを設定して検証した後、MFA の TOTP ソフトウェアトークンを有効化できます。アプリケーション

でユーザーのサインインに Amazon Cognito でホストされた UI を使用する場合は、ユーザーはユーザー名とパスワードを送信し、追加のサインインページで TOTP パスワードを送信します。

ユーザープールの TOTP MFA は、Amazon Cognito コンソールでアクティブ化することも、Amazon Cognito API オペレーションを使用してもかまいません。ユーザープールレベルでは、SetUserPoolMfaConfigを呼び出して MFA を設定し、TOTP MFA を有効にできます。

> **Note**
>
> TOTP ソフトウェアトークン MFA がユーザープールに対して有効になっていない場合は、Amazon Cognito で、ユーザーをトークンに関連付けたり、トークンで検証することができません。この場合、ユーザーは`SoftwareTokenMFANotFoundException`説明による例外`Software Token MFA has not been enabled by the userPool`を受診します。ソフトウェアトークン MFA が後にユーザープールに対して非アクティブ化された場合、以前に TOTP トークンを関連付けて検証したユーザーは、引き続き MFA で使用することができます。

ユーザーの TOTP の設定には複数のステップが伴います。ユーザーはシークレットコードを受け取り、ワンタイムパスワードを入力して、このコードを検証します。次に、ユーザーの TOTP MFA を有効にするか、ユーザーの優先 MFA メソッドとして TOTP を設定することができます。

TOTP MFA を要求するようにユーザープールを設定し、ユーザーがホストされた UI でアプリにサインアップすると、Amazon Cognito はユーザープロセスを自動化します。Amazon Cognito は、ユーザーに MFA メソッドの選択を促し、認証アプリを設定するための QR コードを表示し、MFA 登録を確認します。ユーザーが SMS と TOTP MFA のどちらかを選択できるようにしたユーザープールでは、Amazon Cognito もユーザーに選択肢を提示します。ホストされた UI サインアップのエクスペリエンスについて詳しくは、「Amazon Cognito でホストされた UI での新規アカウントのサインアップの方法」を参照してください。

> **Important**
>
> ユーザープールに関連付けられた AWS WAF ウェブ ACL があり、ウェブ ACL のルールが CAPTCHA を提示すると、ホストされた UI TOTP 登録で回復不可能なエラーが発生する可能性があります。CAPTCHA アクションを含み、ホストされた UI TOTP には影響しないルールを作成するには、「ホストされた UI TOTP MFA の AWS WAF ウェブ ACL の設定」を参照してください。 AWS WAF ウェブ ACLs「」を参照してくださいAWS WAF ウェブ ACL とユーザープールの関連付け。 Amazon Cognito

Amazon Cognito API を使用するカスタム UI で TOTP MFA を実装するには、「Amazon Cognito ユーザープール API でユーザーの MFA を設定する」を参照してください。

MFA をユーザープールに追加するには、「ユーザープールに MFA を追加します」を参照してください。

TOTP MFA 考慮事項と制約事項

1. Amazon Cognito は、TOTP コードを生成する認証システムアプリケーションを介してソフトウェアトークン MFA をサポートします。Amazon Cognito はハードウェアベースの MFA をサポートしていません。
2. TOTP を設定していないユーザーに対してユーザープールが TOTP を必要とする場合、ユーザーはワンタイムアクセストークンを受け取り、アプリはそれを使ってユーザーの TOTP MFA を有効化することができます。後続のサインイン試行は、ユーザーが追加の TOTP サインイン要素を登録するまで失敗します。
   - `SignUp` API オペレーションまたはホストされた UI を介してユーザープールにサインアップしたユーザーが、サインアップを完了すると 1 回限りのトークンを受け取ります。
   - ユーザーを作成し、ユーザーが初期パスワードを設定すると、Amazon Cognito はホストされた UI からユーザーに 1 回限りのトークンを発行します。ユーザーに永続的なパスワードを設定すると、ユーザーが最初にサインインしたときに Amazon Cognito が 1 回限りのトークンを発行します。
   - Amazon Cognito は、 InitiateAuthまたは AdminInitiateAuth API オペレーションでサインインする管理者作成ユーザーに 1 回限りのトークンを発行しません。ユーザーが初期パスワードの設定のチャレンジに成功した後、またはユーザーに永続的なパスワードを設定すると、Amazon Cognito はすぐに MFA の設定をユーザーに要求します。
3. MFA を必要とするユーザープール内のユーザーがワンタイムアクセストークンをすでに受け取っていても TOTP MFA を設定していない場合、ユーザーは MFA を設定するまでホストされた UI でサインインできません。アクセストークンの代わりに、 への`MFA_SETUP`チャレンジInitiateAuthまたは AssociateSoftwareTokenリクエストAdminInitiateAuthの`session`レスポンス値を使用できます。
4. ユーザーが TOTP を設定した場合は、TOTP が後でユーザープールに対して無効にされた場合でも、その TOTP を MFA に使用できます。
5. Amazon Cognito は、SHA-1 ハッシュ関数を使用してコードを生成する認証アプリからの TOTP のみを受け入れます。SHA-256 ハッシュで生成されたコードは `Code mismatch` エラーを返します。

Amazon Cognito ユーザープール API でユーザーの MFA を設定する

ユーザーが最初にサインインすると、アプリはワンタイムアクセストークンを使用して TOTP プライベートキーを生成し、テキスト形式または QR コード形式でユーザーに提示します。ユーザーは認証システムアプリケーションを設定し、その後のサインイン試行の TOTP を提供します。アプリまたはホストされた UI は、MFA チャレンジレスポンスで TOTP を Amazon Cognito に提示します。

トピック

- TOTP ソフトウェアトークンを関連付ける
- TOTP トークンを検証
- TOTP MFA でのサインイン
- TOTP トークンを削除

TOTP ソフトウェアトークンを関連付ける

TOTP トークンを関連付けるには、ワンタイムパスワードで検証する必要のあるシークレットコードをユーザーに送信する必要があります。トークンの関連付けには 3 つのステップが必要です。

1. ユーザーが TOTP ソフトウェアトークン MFA を選択したら、 AssociateSoftwareTokenを呼び出して、ユーザーアカウント用に一意に生成された共有シークレットキーコードを返します。アクセストークンまたはセッション文字列 AssociateSoftwareToken を使用して を承認できます。
2. アプリは、プライベートキーまたはプライベートキーから生成した QR コードをユーザーに提示します。ユーザーは、キーを Google Authenticator などの TOTP 生成アプリに入力する必要があります。libqrencode を使用して、QR コードを生成できます。
3. ユーザーがキーを入力するか、Google Authenticator などの認証システムアプリケーションに QR コードをスキャンすると、アプリがコードの生成を開始します。

TOTP トークンを検証

次に、TOTP トークンを検証します。以下のように、ユーザーからサンプルコードをリクエストし、Amazon Cognito サービスに提供して、ユーザーが TOTP コードを正常に生成していることを確認します。

1. アプリは、ユーザーが認証システムアプリケーションを適切に設定したことを示すコードの入力をユーザーに促します。

2. ユーザーの認証システムアプリケーションは、一時的なパスワードを表示します。認証システムアプリケーションは、ユーザーに与えたシークレットキーに基づいてパスワードを作成します。
3. ユーザーは一時パスワードを入力します。アプリは、VerifySoftwareToken API リクエストで一時パスワードを Amazon Cognito に渡します。
4. Amazon Cognito は、ユーザーに関連付けられたシークレットキーを保持し、TOTP を生成し、ユーザーが指定したシークレットキーと比較します。一致した場合は、VerifySoftwareToken は SUCCESS レスポンスを返します。
5. Amazon Cognito は TOTP 要素をユーザーに関連付けます。
6. VerifySoftwareToken オペレーションが ERROR レスポンスを返した場合は、ユーザーのクロックが正しいこと、およびリトライの最大回数を超えていないことを確認します。Amazon Cognito は、試行の前後 30 秒以内の TOTP トークンを受け入れ、マイナークロックスキューを考慮します。問題を解決したら、 VerifySoftwareToken オペレーションを再試行してください。

TOTP MFA でのサインイン

この時点で、ユーザーは時間ベースのワンタイムパスワードを使用したサインインを行います。以下はその手順です。

1. ユーザーはユーザー名とパスワードを入力してクライアントアプリにサインインします。
2. TOTP MFA チャレンジが呼び出され、アプリが一時パスワードを入力するプロンプトをユーザーに表示します。
3. ユーザーは、関連付けられた TOTP 生成アプリから一時パスワードを取得します。
4. ユーザーが TOTP コードをクライアントアプリに入力します。アプリは、コードを検証するよう Amazon Cognito サービスに通知します。サインインごとに、 を呼び出して、新しい TOTP 認証チャレンジに対する応答を取得RespondToAuthChallengeする必要があります。
5. Amazon Cognito によってトークンが検証されると、サインインが成功し、ユーザーは認証フローを続行します。

TOTP トークンを削除

最後に、アプリは TOTP 設定を非アクティブ化することをユーザーに許可する必要があります。現在、ユーザーの TOTP ソフトウェアトークンを削除することはできません。ユーザーのソフトウェアトークンを置き換えるには、新しいソフトウェアトークンを関連付けて検証します。ユーザーの TOTP MFA を無効にするには、SetUserMFAPreference を呼び出して、MFA を使用しないか、SMS MFA のみを使用するようにユーザーを変更します。

1. MFA をリセットしたいユーザーのためのインターフェイスをアプリケーション内に作成します。このインターフェイスでユーザーにパスワードの入力を求めます。
2. Amazon Cognito が TOTP MFA チャレンジを返す場合は、SetUserMFAPreference を使用してユーザーの MFA 設定を更新します。
3. アプリで、MFA を非アクティブ化したことをユーザーに伝え、再度サインインするよう促します。

ホストされた UI TOTP MFA の AWS WAF ウェブ ACL の設定

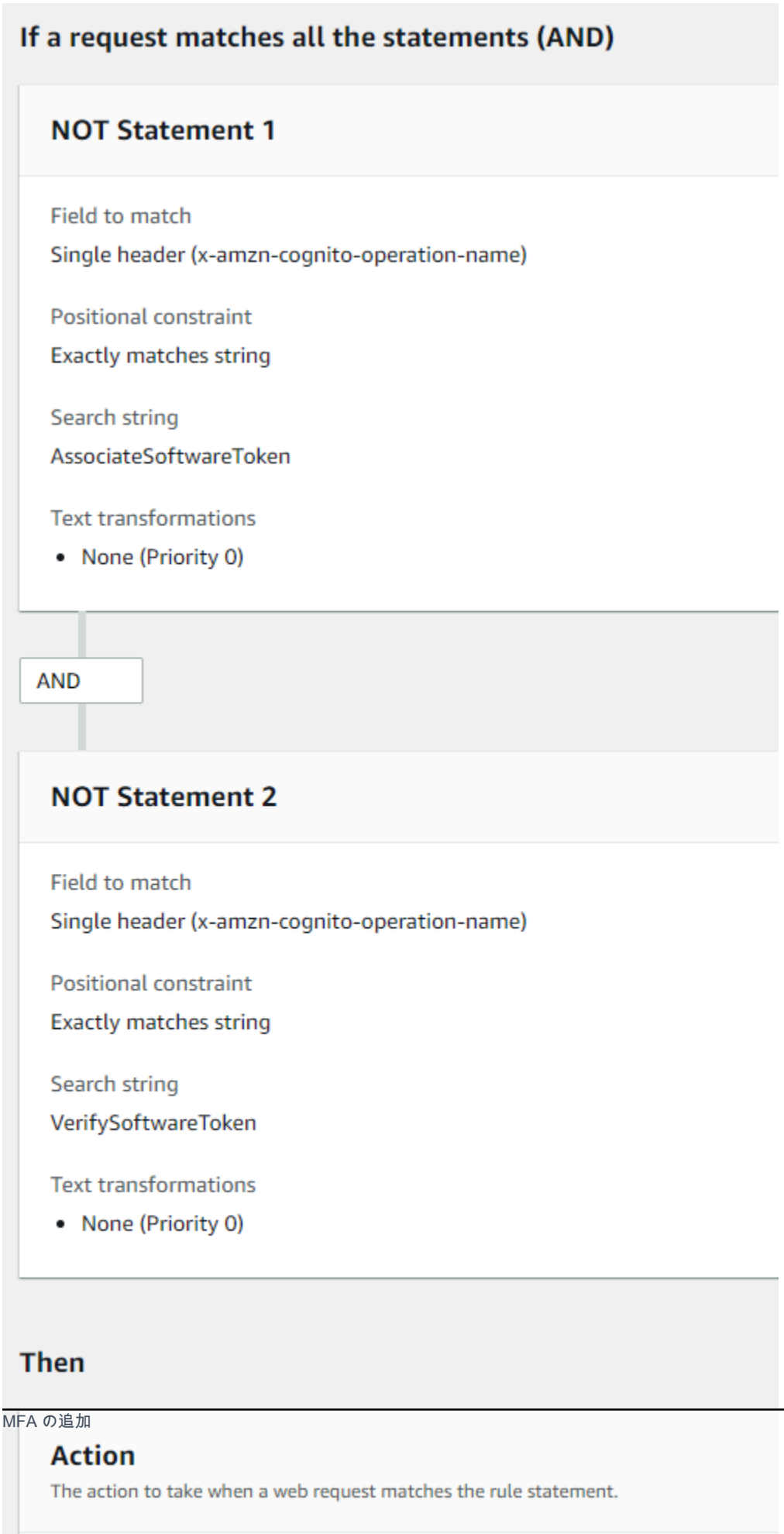
ユーザープールに関連付けられた AWS WAF ウェブ ACL があり、ウェブ ACL のルールが CAPTCHA を提示している場合、ホストされた UI TOTP 登録で回復不可能なエラーが発生する可能性があります。 AWS WAF CAPTCHA ルールは、ホストされた UI の TOTP MFA にのみこの方法で影響します。SMS MFA は影響を受けません。

CAPTCHA ルールにより、ユーザーが TOTP MFA の設定を完了できない場合、Amazon Cognito は次のエラーを表示します。

Request not allowed due to WAF captcha. (WAF captcha によりリクエストは許可されていません。)

このエラーは、 がユーザープールがバックグラウンドで行う AssociateSoftwareTokenおよび VerifySoftwareToken API リクエストに応答して CAPTCHA を AWS WAF プロンプトする場合に発生します。CAPTCHA アクションを含む、ホストされた UI TOTP に影響しないルールを作成するには、ルール内の CAPTCHA アクションから `AssociateSoftwareToken` と `VerifySoftwareToken` の `x-amzn-cognito-operation-name` のヘッダー値を除外します。

次のスクリーンショットは、`x-amzn-cognito-operation-name`ヘッダー値が `AssociateSoftwareToken`または でないすべてのリクエストに CAPTCHA アクションを適用する AWS WAF ルールの例を示しています`VerifySoftwareToken`。


If a request matches all the statements (AND)
NOT Statement 1
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
AssociateSoftwareToken
Text transformations
None (Priority 0)
AND
NOT Statement 2
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
VerifySoftwareToken
Text transformations
None (Priority 0)
Then
Action
The action to take when a web request matches the rule statement.

AWS WAF ウェブ ACLs「」を参照してくださいAWS WAF ウェブ ACL とユーザープールの関連付け。 Amazon Cognito

ユーザープールにアドバンストセキュリティを追加する

ユーザープールを作成すると、Amazon Cognito コンソールのナビゲーションバーの [Advanced security] (アドバンストセキュリティ) にアクセスできるようになります。ユーザープールのアドバンストセキュリティ機能をオンにし、さまざまなリスクに対応して実行されるアクションをカスタマイズすることができます。また、監査モードを使用して、セキュリティ緩和策を適用せずに、検出されたリスクに関するメトリクスを収集できます。監査モードでは、高度なセキュリティ機能がメトリクスを Amazon に発行します CloudWatch。Amazon Cognito が最初の高度なセキュリティイベントを生成した後に、高度なセキュリティメトリクスを確認できます。アドバンストセキュリティのメトリクスの表示 を参照してください。

高度なセキュリティ機能には、漏えいした認証情報の検出とアダプティブ認証が含まれます。

漏えいした認証情報

ユーザーは複数のユーザーアカウントのパスワードを再利用します。Amazon Cognito の認証情報漏えい機能では、ユーザー名とパスワードの公開漏えいに関するデータを収集し、ユーザーの認証情報を漏えいした認証情報のリストと比較します。漏えいした認証情報の検出では、よく推測されるパスワードもチェックされます。

漏えいした認証情報の確認を促すユーザーアクションと、それに応じて Amazon Cognito に実行してほしいアクションを選択できます。サインイン、サインアップ、パスワード変更のイベントでは、Amazon Cognito はサインインをブロックするか、サインインを許可することができます。どちらの場合も、Amazon Cognito はユーザーアクティビティログを生成し、そこでイベントに関する詳細情報を確認できます。

アダプティブ認証

Amazon Cognito は、ユーザーのサインインリクエストから位置情報とデバイス情報を確認し、自動応答を適用してユーザープールのユーザーアカウントを疑わしいアクティビティから保護します。

高度なセキュリティを有効にすると、Amazon Cognito はユーザーアクティビティにリスクスコアを割り当てます。疑わしいアクティビティには自動応答を割り当てることができます。たとえば、MFA を義務付けたり、ログインをブロックしたり、アクティビティの詳細とリスクスコアを記録したりできます。また、疑わしいアクティビティをユーザーに通知する E メールメッセージを自動的に送信して、ユーザーがパスワードのリセットやその他の自発的なアクションを実行できるようにすることもできます。

アクセストークンのカスタマイズ

アドバンストセキュリティ機能を有効にすると、バージョン 2 の Lambda トリガーイベントへのレスポンスを受け入れるようにユーザープールを設定できます。バージョン 2 では、アクセストークンのスコープやその他のクレームをカスタマイズできます。これにより、ユーザーの認証時に柔軟な承認結果を作成する能力が高まります。詳細については、「アクセストークンのカスタマイズ」を参照してください。

トピック

- 考慮事項と制約事項
- 前提条件
- アドバンストセキュリティ機能を設定する
- 侵害された認証情報の確認
- アダプティブ認証の使用
- アドバンストセキュリティのメトリクスの表示
- アプリからユーザープールのアドバンストセキュリティを有効化する

考慮事項と制約事項

- Amazon Cognito のアドバンストセキュリティ機能には追加料金が適用されます。Amazon Cognito の料金ページを参照してください。
- Amazon Cognito は、、USER_PASSWORD_AUTH、および の標準認証フローでアダプティブ認証ADMIN_USER_PASSWORD_AUTHと侵害された認証情報の検出をサポートしますUSER_SRP_AUTH。CUSTOM_AUTH フローと カスタム認証チャレンジの Lambda トリガー、またはフェデレーションサインインでアドバンストセキュリティ機能を使用することはできません。
- 完全な機能モードでの Amazon Cognito のアドバンストセキュリティ機能を使用して、IP アドレス 常にブロック および 常に許可 例外を作成することができます。常にブロック 例外リストの IP アドレスからのセッションは、アダプティブ認証によってリスクレベルが割り当てられておらず、ユーザープールにサインインできません。
- ユーザープールの 常にブロック 例外リストの IP アドレスからのブロックされたリクエストは、ユーザープールのリクエストレートクォータ に貢献します。Amazon Cognito のアドバンスドセキュリティ機能は、分散型サービス妨害 (DDoS) 攻撃を防ぐものではありません。ユーザープールにボリューメトリック攻撃に対する防御を実装するには、 AWS WAF ウェブ ACLs を追加しま

す。詳細については、「AWS WAF ウェブ ACL とユーザープールの関連付け」を参照してください。

- クライアント認証情報の付与は、ユーザーアカウントに接続しない machine-to-machine (M2M) 認証を目的としています。高度なセキュリティ機能は、ユーザープール内のユーザーアカウントとパスワードのみをモニタリングします。M2M アクティビティでセキュリティ機能を実装するには、リクエストレートとコンテンツをモニタリング AWS WAF するための の機能を検討してください。詳細については、「AWS WAF ウェブ ACL とユーザープールの関連付け」を参照してください。

前提条件

開始するには、以下が必要です。

- アプリクライアントを持つユーザープール。詳細については、「ユーザープールの開始方法」を参照してください。
- Amazon Cognito コンソールで多要素認証 (MFA) を [Optional] (オプション) に設定して、リスクに基づくアダプティブ認証機能を使用する。詳細については、「ユーザープールに MFA を追加します」を参照してください。
- 通知に E メールを使用している場合は、Amazon SES コンソールに移動して、通知 E メールで使用する E メールアドレスまたはドメインの設定と検証を実行します。Amazon SES の詳細については、「Amazon SES での ID の検証」を参照してください。

アドバンストセキュリティ機能を設定する

Amazon Cognito のアドバンストセキュリティ機能は、 AWS Management Consoleで設定できます。

ユーザープールにアドバンストセキュリティを設定するには

1. Amazon Cognito コンソールに移動します。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [App integration] (アプリケーションの統合) タブを選択します。アドバンストセキュリティを探し、[Enable] (有効) を選択します。アドバンスセキュリティを既に有効にしている場合は、[Edit] (編集) を選択します。

5. [Full function] (完全な機能) を選択して、侵害されたクレデンシャルとアダプティブ認証に対する高度なセキュリティレスポンスを設定します。監査のみを選択して情報を収集し、ユーザープールデータを に送信します CloudWatch。アドバンストセキュリティ料金は、[Audit only] (監査のみ) および [Full function] (完全な機能) モードの両方で適用されます。詳細については、「Amazon Cognito の料金」を参照してください。

   アクションを有効にする前に、2 週間、アドバンストセキュリティ機能を監査モードにしておくことをお勧めします。この間、Amazon Cognito がユーザーアプリの使用パターンを学習できるようになります。

6. [Audit only] (監査のみ) を選択した場合、[Save changes] (変更の保存) を選択してください。[Full function] (完全な機能) を選択した場合:

   a. 侵害された認証情報への応答は、 [Custom] (カスタム) アクションの実行または使用、または [Cognito defaults] (Cognito デフォルト) を選択します。Cognito デフォルトでは、

      i. サインイン,サインアップ, およびパスワードの変更時に、侵害された認証情報を検出します。

      ii. 侵害された認証情報は、[Block sign-in] (サインインをブロックする) で応答します。

   b. [Compromised credentials] (侵害された認証情報) で [Custom] (カスタム) アクションを選択した場合、Amazon Cognito が [Event detection] (イベント検出) で使用するユーザープールアクション、および、Amazon Cognito に実行させたい [Compromised credentials responses] (侵害された認証情報への応答) を選択します。侵害された認証情報によるサインインをブロックまたはサインインを許可することができます。

   c. [Adaptive authentication] (アダプティブ認証) で、悪意のあるサインイン試行への応答方法を選択します。[Custom] (カスタム) アクションを実行または使用、または悪意のあるアクティビティの疑いに対応するために [Cognito defaults] (Cognito デフォルト) を選択します。[Cognito defaults] (Cognito デフォルト) を選択した場合、Amazon Cognito はすべてのリスクレベルでサインインをブロックし、ユーザーに通知しません。

   d. [Adaptive authentication] (アダプティブ認証) で、 [Custom] (カスタム) アクションを選択した場合、重要度レベルに基づいて検出されたリスクに対して、Amazon Cognito で実行する [Automatic risk response] (自動リスク対応) アクションを選択します。リスクのレベルに応じて対応を割り当てる場合、より高いレベルのリスクに対して、より制限の少ない対応を割り当てることはできません。リスクレベルには、次の対応を割り当てることができます。

      i. サインインを許可する - 予防策をとりません。

      ii. MFA のオプション - ユーザーが MFA を設定している場合、Amazon Cognito はサインイン時に常に SMS またはタイムベースドワンタイムパスワード (TOTP) の追加要素を

提供するようユーザーに要求します。ユーザーに MFA が設定されていない場合は、通常どおりサインインを続行できます。

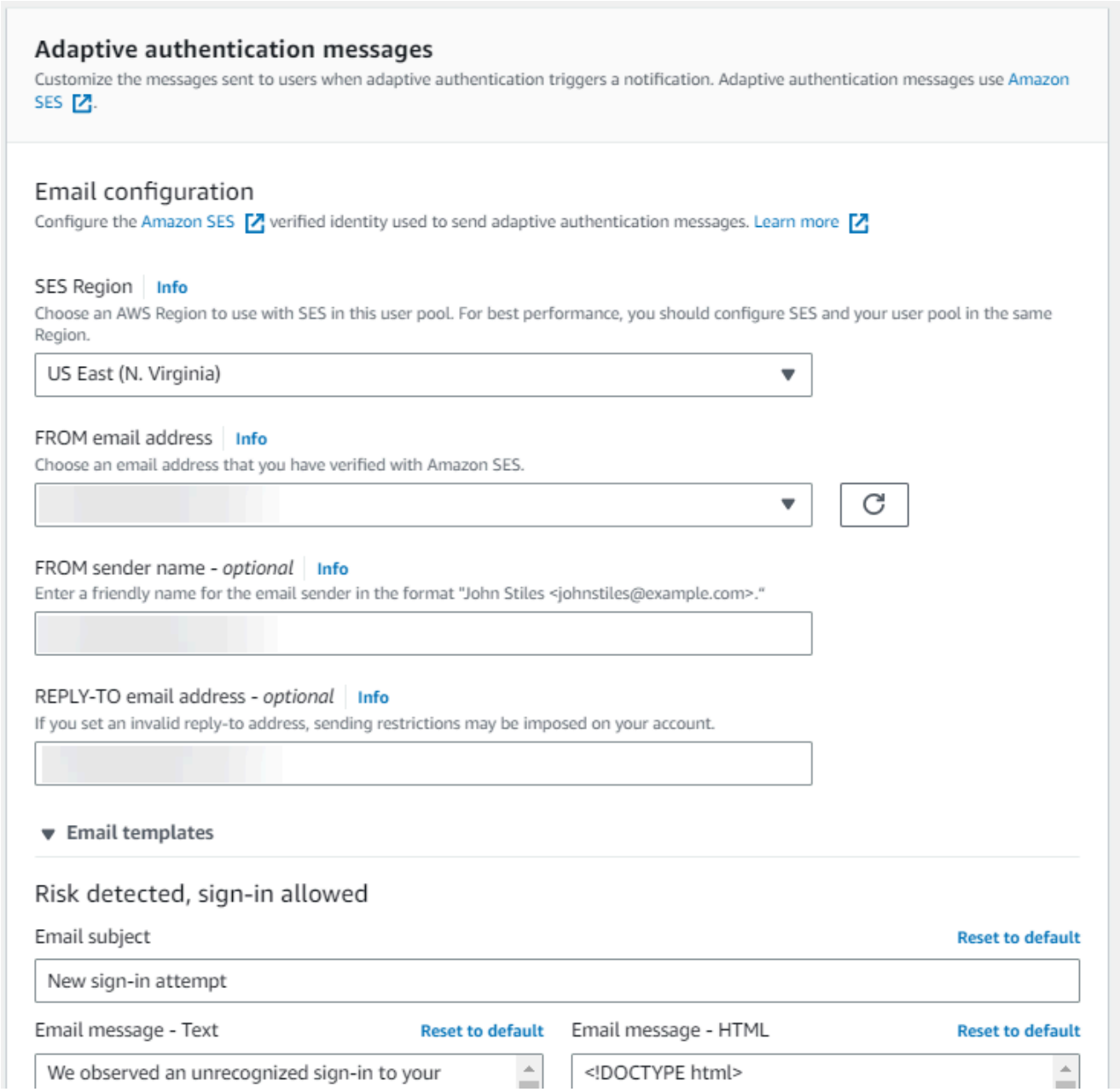
iii. MFA を要求 - ユーザーが MFA を設定している場合、Amazon Cognito はサインイン時に常に SMS または TOTP ファクターの追加要素の提供を要求します。ユーザーに MFA が設定されていない場合、Amazon Cognito は MFA を設定するよう促します。ユーザーに MFA を自動要求する前に、SMS MFA の電話番号をキャプチャするメカニズム、または TOTP MFA の認証アプリケーションを登録するメカニズムを、お客様のアプリケーションに設定してください。

iv. サインインをブロックする - ユーザーがサインインできないようにします。

v. ユーザーに通知する - Amazon Cognito が検出したリスクと応答に関する情報を E メールでユーザーに送信します。送信するメッセージの E メールメッセージテンプレートをカスタマイズできます。

7. 前の手順で [Notify user] (ユーザーに通知する) を選択した場合、アダプティブ認証に使用する E メール配信設定と E メールメッセージテンプレートをカスタマイズできます。

a. [Email configuration] (E メール設定) で、アダプティブ認証で使用する [SES Region] (SES リージョン)、[FROM email address] (FROM E メールアドレス)、[FROM sender name] (FROM 送信者名)、および [REPLY-TO email address] (REPLY-TO E メールアドレス) を選択します。ユーザープールの E メールメッセージを Amazon Simple Email Service の統合方法の詳細については、「Amazon Cognito ユーザープールに使用する E メールの設定」を参照してください。



b. [Email templates] (E メールテンプレート) を展開して、E メールメッセージの HTML とプレーンテキストバージョンの両方でアダプティブ認証通知をカスタマイズします。E メールメッセージテンプレートの詳細については、「メッセージテンプレート」を参照してください。

8. IP アドレスの例外を拡張して、アドバンストセキュリティリスク評価に関係なく、常に許可またはブロックされる IPv4 または IPv6 アドレス範囲に対する常時許可または常時ブロックのリストを作成します。CIDR 表記 で IP アドレスの範囲を指定します (例: 192.168.100.0/24)。

9. [Save changes] (変更の保存) をクリックします。

侵害された認証情報の確認

Amazon Cognito は、ユーザーのユーザー名とパスワードが他の場所で侵害されたかどうかを検出できます。これは、ユーザーが複数のサイトで認証情報を再利用したり、安全でないパスワードを使用したりするときに発生します。Amazon Cognito は、ホストされた UI と Amazon Cognito API で、ユーザー名とパスワードでサインインするローカルユーザーをチェックします。ローカルユーザーは、外部 IdP を介したフェデレーションなしに、ユーザープールディレクトリにのみ存在します。

Amazon Cognito コンソールの [App integration] (アプリの統合) タブの[Advanced security](アドバンストセキュリティ) から、[Compromised credentials] (侵害された認証情報) を設定できます。[Event detection] (イベント検出) を設定して、侵害された認証情報を監視するユーザーイベントを選択します。[Compromised credentials responses] (侵害された認証情報の応答) を設定し、侵害された認証情報が検出された場合にユーザーを許可するかブロックするかを選択します。Amazon Cognito は、サインイン時、サインアップ時、パスワード変更時に侵害された認証情報をチェックすることができます。

サインインを許可する を選択すると、Amazon CloudWatch Logs を確認して、Amazon Cognito がユーザーイベントに対して行う評価をモニタリングできます。詳細については、「アドバンストセキュリティのメトリクスの表示」を参照してください。[Block sign-in] (サインインをブロックする) を選択する場合、Amazon Cognito は、侵害された認証情報を使用するユーザーによるサインインを防止します。Amazon Cognito がユーザーのサインインをブロックすると、ユーザーの `UserStatus` が `RESET_REQUIRED` に設定されます。`RESET_REQUIRED` ステータスのユーザーは、再度サインインする前にパスワードを変更する必要があります。

> **Note**
>
> 現在 Amazon Cognito では、Secure Remote Password (SRP) フローでのサインイン操作に対する侵害された認証情報のチェックが行われません。SRP はサインイン時にハッシュ化されたパスワード証明書を送信します。Amazon Cognito は内部でパスワードにアクセスできないため、クライアントがプレーンテキストで渡したパスワードのみを評価できます。Amazon Cognito は、フローで AdminInitiateAuth API を使用し、`ADMIN_USER_PASSWORD_AUTH`フローで InitiateAuth API を使用するサインインで`USER_PASSWORD_AUTH`、認証情報が侵害されていないかをチェックします。

漏洩した認証情報の保護をユーザープールに追加するには、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

アダプティブ認証の使用

アダプティブ認証では、リスクレベルの上昇に対応して、疑わしいサインインをブロックする、または 2 番目の要素認証を追加するようにユーザープールを設定できます。Amazon Cognito はサインインの試行ごとに、サインインリクエストが侵害されたソースからのものである可能性についてリスクスコアを生成します。このリスクスコアは、デバイスとユーザー情報を含む要因に基づいています。アダプティブ認証は、Amazon Cognito がユーザーのセッションでリスクを検出し、ユーザーがまだ MFA メソッドを選択していない場合に、ユーザープール内のユーザーに対して多要素認証 (MFA) を有効にする、または多要素認証 (MFA) を要求することができます。ユーザーに MFA を有効にすると、アダプティブ認証の設定方法にかかわらず、認証中に必ず 2 つ目の要素を提供または設定するように求めるチャレンジが常に表示されます。ユーザーの観点から見ると、アプリは MFA の設定を支援し、また、オプションとして、Amazon Cognito では、ユーザーが追加の要素を設定するまで、ユーザーが再度サインインできないようにすることもできます。

Amazon Cognito は、サインイン試行、リスクレベル、失敗したチャレンジを Amazon に発行します CloudWatch。詳細については、「アドバンストセキュリティのメトリクスの表示」を参照してください。

アダプティブ認証をユーザープールに追加するには、「ユーザープールにアドバンストセキュリティを追加する」を参照してください。

トピック

- アダプティブ認証の概要
- API リクエストへのユーザーデバイスおよびセッションデータの追加
- ユーザーイベント履歴の表示
- イベントフィードバックを提供します
- 通知メッセージの送信

アダプティブ認証の概要

アダプティブ認証の設定は、[App integration] (アプリケーションの統合) タブの [Advanced security] (アドバンスドセキュリティ) から選択でき、これには、異なるリスクレベルで実行するアクションの設定や、ユーザーへの通知メッセージのカスタマイズなどが含まれます。すべてのアプリクライアントにグローバルな高度なセキュリティ設定を割り当てることができますが、個々のアプリクライアントにはクライアントレベルの設定を適用できます。

Amazon Cognito アダプティブ認証は、各ユーザーセッションに「高」、「中」、「低」、「リスクなし」のいずれかのリスクレベルを割り当てます。

[Enforcement method] (強制実行メソッド) を [Audit-only] (監査専用) から [Full-function] (完全な機能) に変更する場合は、選択肢を慎重に検討してください。リスクレベルに適用する自動応答は、Amazon Cognito が同じ特性を持つ後続のユーザーセッションに割り当てるリスクレベルに影響します。例えば、Amazon Cognito が最初にリスクが高いと評価したユーザーセッションに何もしないか、[Allow] (許可) を選択すると、Amazon Cognito では同様のセッションのリスクが低いと見なします。

リスクレベルごとに、以下のオプションから選択できます。

オプション	アクション
許可	ユーザーは、追加要素なしでサインインできます。
オプションの MFA	第 2 要素を設定しているユーザーは、サインインするために第 2 要素のチャレンジを完了する必要があります。SMS の電話番号と TOTP ソフトウェアトークンが利用可能な第 2 要素です。2 番目の要素が設定されていないユーザーは、1 つの認証情報のみでサインインできます。
MFA が必要	第 2 要素を設定しているユーザーは、サインインするために第 2 要素のチャレンジを完了する必要があります。Amazon Cognito は、第 2 要素が設定されていないユーザーのサインインをブロックします。
ブロック	Amazon Cognito は、指定されたリスクレベルですべてのサインイン試行をブロックします。

> **Note**
>
> SMS を 2 番目の認証要素として使用するために電話番号を検証する必要はありません。

API リクエストへのユーザーデバイスおよびセッションデータの追加

API を使用してサインアップ、サインイン、パスワードのリセットを行う際に、ユーザーのセッションに関する情報を収集し、Amazon Cognito のアドバンストセキュリティに渡すことができます。この情報には、ユーザーの IP アドレスと一意のデバイス識別子が含まれます。

ユーザーと Amazon Cognito の間に、プロキシサービスやアプリケーションサーバーなど、中間ネットワークデバイスがある場合があります。ユーザーのコンテキストデータを収集して Amazon Cognito に渡すことで、アダプティブ認証がサーバーやプロキシではなく、ユーザーエンドポイントの特性に基づいてリスクを計算できます。クライアント側アプリが Amazon Cognito API オペレーションを直接呼び出す場合、アダプティブ認証は送信元 IP アドレスを自動的に記録します。ただし、デバイスのフィンガープリントも収集しない限り、`user-agent` などの他のデバイス情報は記録されません。

Amazon Cognito コンテキストデータ収集ライブラリを使用してこのデータを生成し、 ContextData および UserContextDataパラメータを使用して Amazon Cognito の高度なセキュリティに送信します。コンテキストデータ収集ライブラリは AWS SDKsに含まれています。詳細については、「Amazon Cognito のウェブアプリケーションとモバイルアプリケーションとの統合」を参照してください。ユーザープールでアドバンストセキュリティ機能をアクティブ化している場合、`ContextData` を送信できます。詳細については、「アドバンストセキュリティ機能を設定する」を参照してください。

アプリケーションサーバーから以下の Amazon Cognito 認証 API オペレーションを呼び出す場合、`ContextData` パラメータにユーザーのデバイスの IP を渡します。また、サーバー名、サーバーパス、およびエンコードされたデバイスフィンガープリントデータを渡します。

- AdminInitiateAuth
- AdminRespondToAuthChallenge

Amazon Cognito の認証されていない API オペレーションを呼び出すと、Amazon Cognito のアドバンスドセキュリティ機能に `UserContextData` を送信できます。このデータには、`EncodedData` パラメータのデバイスフィンガープリントが含まれます。また、次の条件を満たす場合、`UserContextData` に `IpAddress` パラメータを送信できます。

- ユーザープールでアドバンストセキュリティ機能をアクティブ化しました。詳細については、「アドバンストセキュリティ機能を設定する」を参照してください。
- アプリクライアントにはクライアントシークレットがあります。詳細については、「ユーザープールのアプリクライアントの設定」を参照してください。

- アプリクライアントで [Accept additional user context data] (追加のユーザーコンテキストデータを受け入れる) をアクティブ化しました。詳細については、「追加のユーザーコンテキストデータの受け入れ (AWS Management Console)」を参照してください。

アプリは、以下の Amazon Cognito の認証されていない API オペレーション で、`UserContextData` パラメータにエンコードされたデバイスフィンガープリントデータとユーザーのデバイスの IP アドレスを入力できます。

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword
- ConfirmForgotPassword
- ResendConfirmationCode

追加のユーザーコンテキストデータの受け入れ (AWS Management Console)

ユーザープールは [Accept additional user context data] (追加のユーザーコンテキストデータを受け入れる) 機能をアクティブ化すると、`UserContextData` パラメータで IP アドレスを受け入れます。次の場合は、この機能をアクティブ化する必要はありません。

- ユーザーは などの認証された API オペレーションでのみサインインしAdminInitiateAuth 、`ContextData`パラメータを使用します。
- 認証されていない API オペレーションから IP アドレスではなく、デバイスのフィンガープリントのみを Amazon Cognito のアドバンストセキュリティ機能に送信します。

Amazon Cognito コンソールでアプリクライアントを次のように更新し、追加のユーザーコンテキストデータのサポートを追加します。

1. Amazon Cognito コンソールにサインインします。
2. ナビゲーションペインで [ユーザープールの管理] を選択してから、編集するユーザープールを選択します。
3. [App integration] (アプリケーションの統合) タブを選択します。

4. [App clients and analytics] (アプリクライアントと分析) で、アプリクライアントを選択または作成します。詳細については、「ユーザープールのアプリクライアントの設定」を参照してください。
5. [App client information] (アプリのクライアント情報) コンテナから [Edit] (編集) を選択します。
6. アプリクライアントの [Advanced authentication settings] (詳細な認証設定) で、[Accept additional user context data] (追加のユーザーコンテキストデータを受け入れる) を選択します。
7. [変更を保存] を選択します。

Amazon Cognito API でユーザーコンテキストデータを受け入れるようにアプリクライアントを設定するには、 CreateUserPoolClientまたは UpdateUserPoolClientリクエストtrueで EnablePropagateAdditionalUserContextData を に設定します。ウェブまたはモバイルアプリからアドバンストセキュリティをアクティブ化する方法の詳細については、「アプリからユーザープールのアドバンストセキュリティを有効化する」を参照してください。アプリがサーバーから Amazon Cognito を呼び出すときに、クライアント側からユーザーコンテキストデータを収集します。 JavaScript SDK メソッド を使用する例を次に示しますgetData。

```
var encodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

アダプティブ認証を使用するようにアプリを設計する場合は、最新の Amazon Cognito SDK をアプリに組み込むことをお勧めします。SDK の最新バージョンでは、デバイス ID、モデル、およびタイムゾーンなどのデバイスフィンガープリント情報を収集します。Amazon Cognito SDK の詳細については、「ユーザープール SDK のインストール」を参照してください。Amazon Cognito アドバンストセキュリティでは、アプリが正しい形式で送信したイベントにのみ保存され、リスクスコアが割り当てられます。Amazon Cognito がエラーレスポンスを返す場合は、リクエストに有効なシークレットハッシュが含まれていることと、IPaddress パラメータが、有効な IPv4 または IPv6 アドレスであることを確認します。

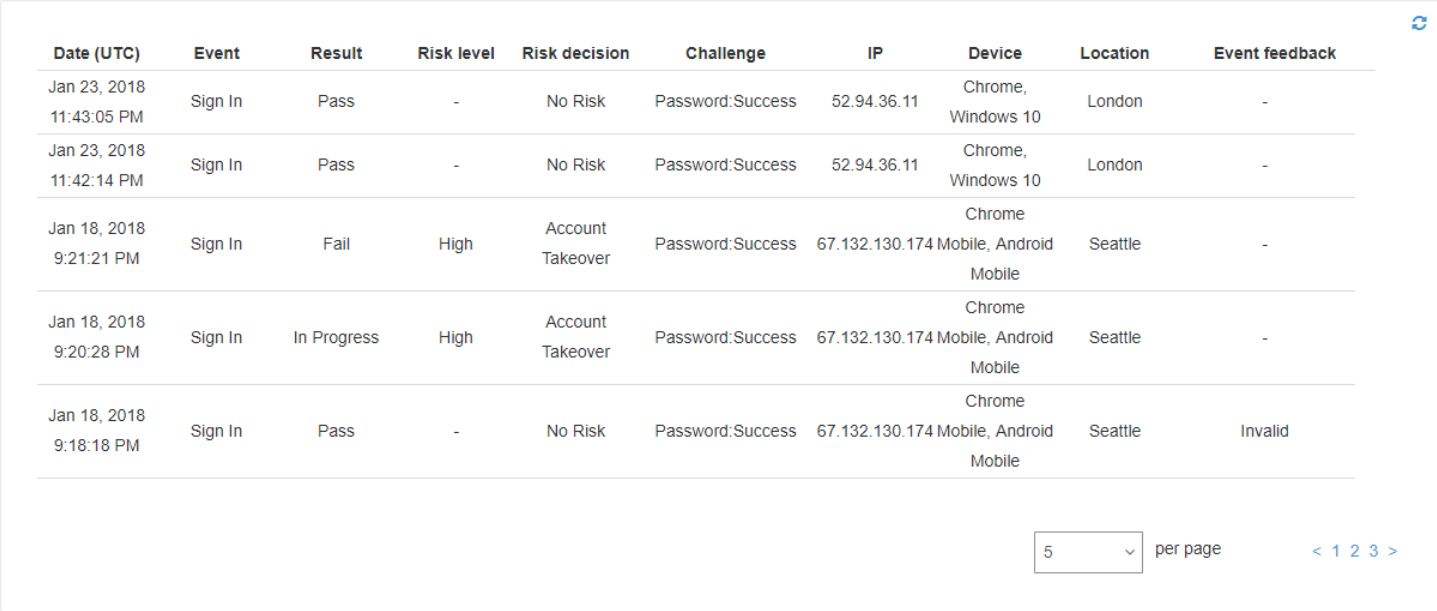
ContextData および UserContextData リソース

- AWS Amplify Android 用 SDK: GetUserContextData
- AWS Amplify SDK for iOS: userContextData
- JavaScript: amazon-cognito-advanced-security-data.min.js

ユーザーイベント履歴の表示

> **Note**
>
> 新しい Amazon Cognito コンソールでは、Users (ユーザー) タブよりイベント履歴が表示できます。

ユーザーのサインイン履歴を表示するには、Amazon Cognito コンソールの [Users] (ユーザー) タブでユーザーを選択できます。Amazon Cognito は、ユーザーイベント履歴を 2 年間保持します。



Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

5 per page < 1 2 3 >

各サインインイベントにはイベント ID があります。イベントには、場所、デバイスの詳細、およびリスク検出結果など、対応するコンテキストデータもあります。Amazon Cognito API オペレーション AdminListUserAuthEvents または -admin-list-user-authevents の AWS Command Line Interface (AWS CLI) を使用して、ユーザーイベント履歴をクエリできます。

また、イベント ID を Amazon Cognito がイベントを記録した時点で発行したトークンと関連付けることができます。ID とアクセストークンには、ペイロードにこのイベント ID が含まれます。Amazon Cognito はまた、更新トークンの使用を元のイベント ID に関連付けます。元のイベント ID は、Amazon Cognito トークンの発行につながったサインインイベントのイベント ID まで追跡できます。システム内のトークンの使用は、特定の認証イベントまで追跡できます。詳細については、「ユーザープールでのトークンの使用」を参照してください。

イベントフィードバックを提供します

イベントフィードバックは、リアルタイムでリスクの評価に反映され、リスク評価アルゴリズムを経時的に向上させます。ユーザーは、Amazon Cognito コンソールおよび API を使用して、サインイン試行の妥当性に関するフィードバックを提供できます。

> **Note**
>
> イベントフィードバックは、Amazon Cognito が同じ特性を持つ後続のユーザーセッションに割り当てるリスクレベルに影響します。

Amazon Cognito コンソールで、[Users] (ユーザー) タブからユーザーを選択し、[Provide event feedback] (イベントフィードバックを送信) を選択します。イベントの詳細を確認して、[Set as valid] (有効として設定) または [Set as invalid] (無効として設定) できます。

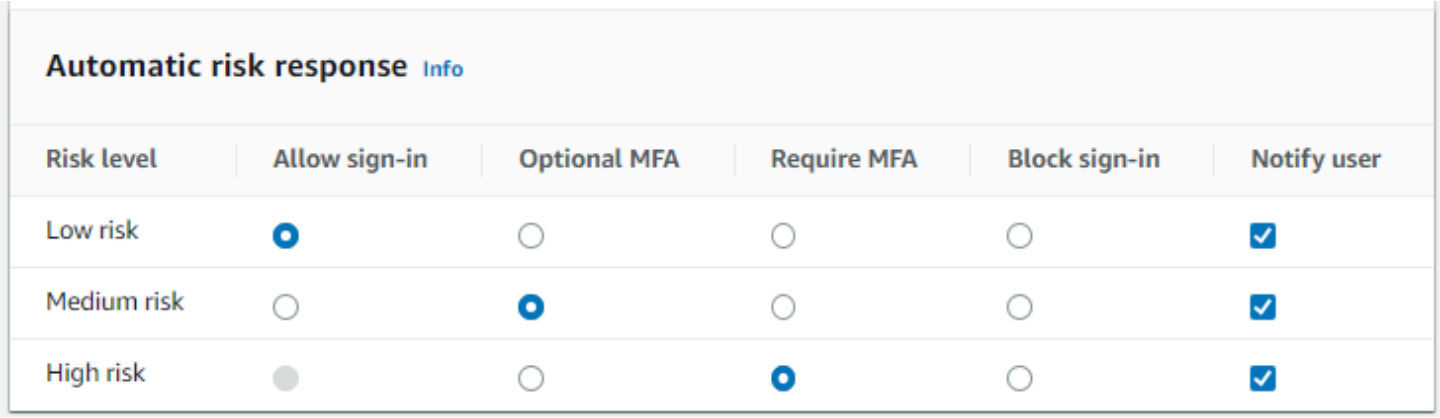
コンソールでは、[Users and groups] (ユーザーとグループ) タブにサインイン履歴が表示されます。エントリを選択すると、イベントを有効または無効としてマークできます。また、ユーザープール API オペレーション AdminUpdateAuthEventFeedbackおよび AWS CLI コマンド admin-update-auth-event-feedback を通じてフィードバックを提供することもできます。

Amazon Cognito コンソールで [Set as valid] (有効として設定) を選択するか、API で `valid` の `FeedbackValue` 値を指定するとき、Amazon Cognito がある程度のリスクを評価したユーザーセッションを信頼することを Amazon Cognito に伝えます。Amazon Cognito コンソールで [Set as invalid] (無効として設定) を選択するか、API で `invalid` の `FeedbackValue` 値を指定するとき、ユーザーセッションを信頼しないこと、または Amazon Cognito が十分に高いリスクレベルを評価したとは考えないことを Amazon Cognito に伝えます。

通知メッセージの送信

アドバンスドセキュリティ保護により、Amazon Cognito はユーザーにリスクの高いサインイン試行を通知できます。Amazon Cognito では、サインインが有効か無効かを示すリンクを選択するようユーザーに促すこともできます。Amazon Cognito はこのフィードバックを使用して、ユーザープールのリスク検出精度を向上させます。

[Automatic risk response] (自動リスク対応) セクションで、低、中、高のリスクケースに応じて [Notify Users] (ユーザーに通知) を選択します。



Amazon Cognito は、ユーザーが E メールアドレスを検証したかどうかに関係なく、E メール通知をユーザーに送信します。

通知 E メールメッセージをカスタマイズして、これらのメッセージのプレーンテキストと HTML の両バージョンを提供できます。E メール通知をカスタマイズするには、高度なセキュリティ設定の[Adaptive authentication messages] (アダプティブ認証メッセージ) から E メールテンプレートを開きます。E メールテンプレートの詳細については、「メッセージテンプレート」を参照してください。

アドバンストセキュリティのメトリクスの表示

Amazon Cognito は、高度なセキュリティ機能のメトリクスを Amazon のアカウントに公開します CloudWatch。Amazon Cognito はアドバンストセキュリティのメトリクスを、リスクレベル別およびリクエストレベル別にグループ化します。

CloudWatch コンソールでメトリクスを表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで [Metrics] (メトリクス) を選択します。
3. Amazon Cognito を選択します。
4. [By Risk Classification] (リスク分類別) などの集約されたメトリクスのグループを選択します。
5. [All metrics] (すべてのメトリクス) タブに、選択したグループのメトリクスが一覧表示されます。以下の操作を行うことができます。

   - テーブルを並べ替えるには、列見出しを使用します。
   - メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。すべてのメトリクスを選択するには、テーブルの見出し行にあるチェックボックスを選択します。

- リソースでフィルターするには、リソース ID を選択し、[Add to search] (検索に追加) を選択します。
- メトリクスでフィルターするには、メトリクス名を選択し、[Add to search] (検索に追加) を選択します。

メトリクス	説明	メトリクスディメンション
CompromisedCredentialRisk	Amazon Cognito が侵害された認証情報を検知したリクエスト。	操作: ディメンションは PasswordChange 、SignIn、または SignUp タイプの操作のみです。 UserPoolId: ユーザープールの識別子。 RiskLevel: high (デフォルト)、medium、または low。
AccountTakeoverRisk	Amazon Cognito がアカウントの乗っ取りリスクを検知したリクエスト。	操作: ディメンションは PasswordChange 、SignIn、または SignUp タイプの操作のみです。 UserPoolId: ユーザープールの識別子。 RiskLevel: high、medium、または low。
OverrideBlock	Amazon Cognito がデベロッパーから提供された設定を理由にブロックしたリクエスト。	操作: ディメンションは PasswordChange 、SignIn、または SignUp タイプの操作のみです。

メトリクス	説明	メトリクスディメンション
		UserPoolId: ユーザープールの識別子。 RiskLevel: high、medium、または low。
Risk	Amazon Cognito がリスクありとしてマークしたリクエスト。	操作: `PasswordChange` 、`SignIn`、または `SignUp` などの操作タイプ。 UserPoolId: ユーザープールの識別子。
NoRisk	Amazon Cognito がリスクを識別しなかったリクエスト。	操作: `PasswordChange` 、`SignIn`、または `SignUp` などの操作タイプ。 UserPoolId: ユーザープールの識別子。

Amazon Cognito では、 ですぐに分析できるように、事前定義された 2 つのメトリクスグループが用意されています CloudWatch。By Risk Classification は、リスクがあるとして Amazon Cognito が識別したリクエストのリスクレベルの詳細を示します。By Request Classification リクエストレベルで集計されたメトリクスを反映します。

集約されたメトリクスグループ	説明
リスク分類別	Amazon Cognito でリスクありとして識別されたリクエスト。
リクエスト分類別	リクエスト別に集約されたメトリクス。

アプリからユーザープールのアドバンストセキュリティを有効化する

ユーザープールのアドバンストセキュリティ機能を設定した後、ウェブまたはモバイルアプリで有効化する必要があります。

での高度なセキュリティの使用 JavaScript

1. Amazon Amazon Cognito Identity SDK for JavaScript をアプリケーションに追加します。
2. CognitoUserPool.js で、 AdvancedSecurityDataCollectionFlagを に設定しますtrue。UserPoolId をユーザープール ID に設定します。
3. このソースリファレンスをアプリケーションの JavaScript ファイルに追加します。を、、us-east-1、us-east-2、、us-west-2、eu-west-2または のリストから eu-west-1<region>に置き換え AWS リージョン ますeu-central-1。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-
cognito-advanced-security-data.min.js"></script>
```

Android でアドバンストセキュリティを使用する

1. AWS Amplify for Android でアプリを作成します。詳細については、AWS Amplify Dev Center の「プロジェクトセットアップ」を参照してください。
2. userContextDataProvider を使用して、認証リクエストにユーザーとデバイスの情報を含めます。

レガシーの Android SDK にユーザーコンテキストデータを追加する方法については、aws-android-sdk-cognitoidentityprovider-asf を参照してください。

iOS でアドバンストセキュリティを使用する

1. AWS Amplify for Swift または Flutter を使用してアプリを作成します。詳細については、AWS Amplify Dev Center の Swift プロジェクトセットアップと Flutter プロジェクトセットアップを参照してください。
2. 認証リクエストにユーザーとデバイスの情報を含めます。InitiateAuth API オペレーションで使用する例については、 の userContextData InitiateAuthInput+Amplify.swift の を参照してください GitHub。

レガシー iOS SDK にユーザーコンテキストデータを追加する方法については、「AWSCognitoIdentityProviderASF」を参照してください。

AWS WAF ウェブ ACL とユーザープールの関連付け

AWS WAF はウェブアプリケーションファイアウォールです。 AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) を使用すると、ホストされた UI および Amazon Cognito API サービスエンドポイントへの不要なリクエストからユーザープールを保護できます。ウェブ ACL を使用すると、ユーザープールが応答するすべての HTTPS ウェブリクエストをきめ細かく制御できます。AWS WAF ウェブ ACLs「 AWS WAF デベロッパーガイド」の「ウェブアクセスコントロールリスト (ウェブ ACL) の管理と使用」を参照してください。

ユーザープールに関連付けられた AWS WAF ウェブ ACL がある場合、Amazon Cognito はユーザーからのリクエストの選択された非機密ヘッダーとコンテンツを に転送します AWS WAF。 はリクエストの内容を AWS WAF 検査し、ウェブ ACL で指定したルールと比較し、Amazon Cognito にレスポンスを返します。

AWS WAF ウェブ ACLs と Amazon Cognito について知っておくべきこと

- によってブロックされたリクエスト AWS WAF は、どのリクエストタイプのリクエストレートクォータにもカウントされません。 AWS WAF ハンドラーは、API レベルのスロットリングハンドラーの前に呼び出されます。
- ウェブ ACL を作成すると、ウェブ ACL が完全に伝達されて Amazon Cognito で使用できるようになるまでに少し時間がかかります。伝達時間は数秒から数分までです。 は、ウェブ ACL が完全に伝達される前に関連付けようと`WAFUnavailableEntityException`すると、 AWS WAF を返します。
- ユーザープールには 1 つのウェブ ACL を関連付けることができます。
- リクエストは、 AWS WAF が検査できる範囲を超えたペイロードになる可能性があります が Amazon Cognito からのオーバーサイズリクエストを処理する方法を設定する方法については、「 デベロッパーガイド」の「オーバーサイズリクエストコンポーネントの処理」を参照してください。 AWS WAF AWS WAF
- AWS WAF Fraud Control アカウント乗っ取り防止 (ATP) を使用するウェブ ACL を Amazon Cognito ユーザープールに関連付けることはできません。`AWS-AWSManagedRulesATPRuleSet` マネージドルールグループを追加するときに、ATP 機能を実装します。ユーザープールに関連付ける前に、ウェブ ACL がこのマネージドルールグループを使用していないことを確認してください。
- ユーザープールに関連付けられた AWS WAF ウェブ ACL があり、ウェブ ACL のルールが CAPTCHA を提示すると、ホストされた UI TOTP 登録で回復不可能なエラーが発生する可能性があります。CAPTCHA アクションを含み、ホストされた UI TOTP には影響しないルールを作成するには、「ホストされた UI TOTP MFA の AWS WAF ウェブ ACL の設定」を参照してください。

AWS WAF は、次のエンドポイントへのリクエストを検査します。

ホストされた UI

ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス 内のすべてのエンドポイントへのリクエスト。

公開 API オペレーション

認証に AWS 認証情報を使用しない Amazon Cognito API へのアプリケーションからのリクエスト。これには、InitiateAuth、、 などの API オペレーションが含まれますRespondToAuthChallengeGetUser。の範囲内にある API オペレーションには、 AWS 認証情報による認証 AWS WAF は必要ありません。これらは、認証されていないか、セッション文字列またはアクセストークンで承認されています。詳細については、「Amazon Cognito ユーザープールの認証済みおよび未認証の API オペレーション」を参照してください。

ウェブ ACL のルールを設定するには、ルールに一致するリクエストに応じてカウント、許可、ブロック、CAPTCHA の表示を行うルールアクションを使用できます。詳細については、AWS WAF デベロッパーガイドの「AWS WAF のルール」を参照してください。ルールアクションに応じて、Amazon Cognito からユーザーに返す応答をカスタマイズできます。

> **Important**
>
> エラー応答をカスタマイズするオプションは、API リクエストを行う方法によって異なります。
>
> - ホストされた UI リクエストのエラーコードと応答本文をカスタマイズできます。ホストされた UI では、ユーザーが解決する CAPTCHA のみを表示できます。
> - Amazon Cognito ユーザープール API を使用して行うリクエストの場合は、ブロック応答を受信するリクエストの応答本文をカスタマイズできます。カスタムエラーコードを 400 ~ 499 の範囲で指定することもできます。
> - AWS Command Line Interface （ AWS CLI) と AWS SDKs は、ブロックまたは CAPTCHA レスポンスを生成するリクエストに`ForbiddenException`エラーを返します。

ウェブ ACL をユーザープールに関連付ける

ユーザープールでウェブ ACL を使用するには、 AWS Identity and Access Management (IAM) プリンシパルに次の Amazon Cognito アクセス許可が必要です。アクセス AWS WAF 許可の詳細につ

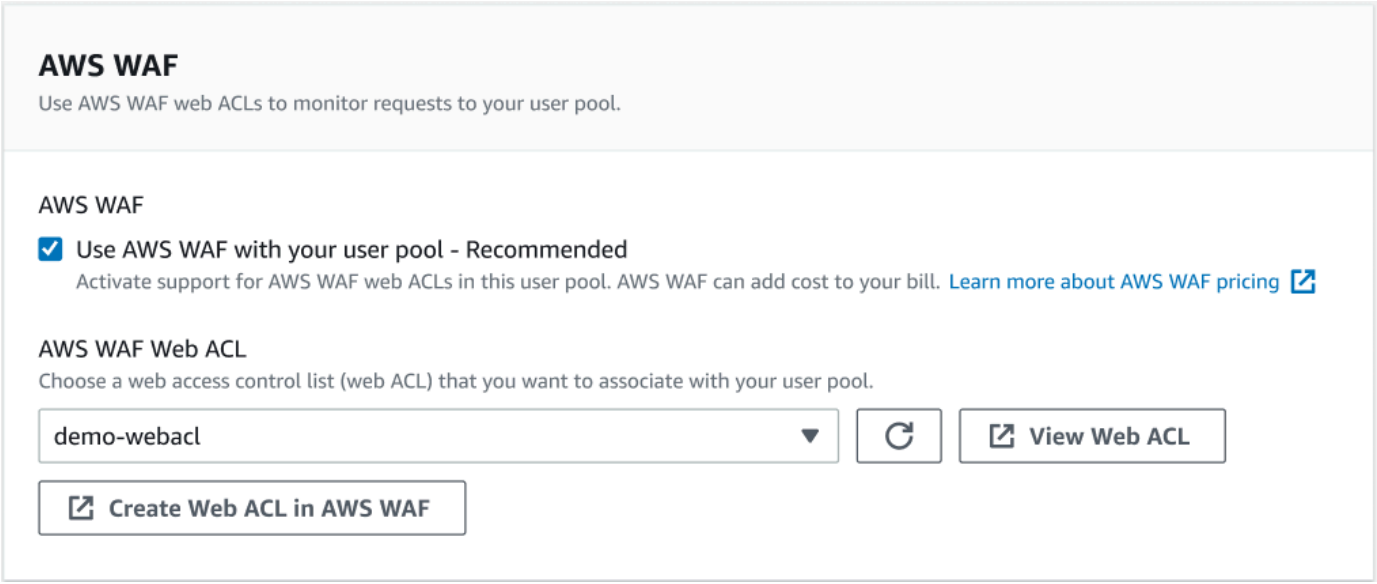
いては、「 AWS WAF デベロッパーガイド」のAWS WAF 「API アクセス許可」を参照してください。

- `cognito-idp:AssociateWebACL`
- `cognito-idp:DisassociateWebACL`
- `cognito-idp:GetWebACLForResource`
- `cognito-idp:ListResourcesForWebACL`

IAM アクセス権限を付与する必要がありますが、リストされているアクションは権限のみで、API オペレーションには対応していません。

ユーザープール AWS WAF に対して をアクティブ化し、ウェブ ACL を関連付けるには

1. Amazon Cognito コンソールにサインインします。
2. ナビゲーションペインで [User Pools] (ユーザープール) を選択してから、編集するユーザープールを選択します。
3. [User pool properties] (ユーザープールのプロパティ) タブを選択します。
4. [AWS WAF] の隣にある [Edit] (編集) を選択します。
5. でAWS WAF、ユーザープール AWS WAF で を使用する を選択します。



6. 既に作成したAWS WAF ウェブ ACL を選択するか、「 でウェブ ACL AWS WAF を作成する」を選択して、 で新しい AWS WAF セッションに作成します AWS Management Console。
7. [変更を保存] を選択します。

ウェブ ACL を AWS Command Line Interface または SDK のユーザープールにプログラムで関連付けるには、 AWS WAF API の AssociateWebACL を使用します。Amazon Cognito には、ウェブ ACL を関連付ける個別の API オペレーションはありません。

AWS WAF ウェブ ACLsテストとログ記録

ウェブ ACL でルールアクションをカウントに設定すると、 はルールに一致するリクエストの数にリクエスト AWS WAF を追加します。ユーザープールでウェブ ACL をテストするには、ルールアクションをカウントに設定し、各ルールに一致するリクエストの量を考慮します。例えば、ルールをブロックアクションに設定した場合に、通常のユーザートラフィックと判断される多数のリクエストと一致するときは、ルールの再設定が必要になる場合があります。詳細については、「 デベロッパーガイドAWS WAF 」の「保護のテストとチューニング」を参照してください。 AWS WAF

リクエストヘッダー AWS WAF を Amazon CloudWatch Logs ロググループ、Amazon Simple Storage Service (Amazon S3) バケット、または Amazon Data Firehose に記録するように を設定することもできます。ユーザープール API で行った Amazon Cognito リクエストは、`x-amzn-cognito-client-id` と `x-amzn-cognito-operation-name` で特定できます ホストされた UI リクエストには、`x-amzn-cognito-client-id` ヘッダーのみが含まれます。詳細については、AWS WAF デベロッパーガイドの「ウェブ ACL トラフィックのログ記録」を参照してください。

AWS WAF ウェブ ACLs には、Amazon Cognito の高度なセキュリティ機能 の料金は適用されません。のセキュリティ機能はAmazon Cognito の高度なセキュリティ機能を AWS WAF 補完します。ユーザープールリクエストの検査のために、ユーザー pool. AWS WAF bills の両機能を個別にアクティブ化できます。詳細については、「AWS WAF の料金」を参照してください。

ログ記録 AWS WAF リクエストデータには、ログをターゲットとするサービスによる追加料金がかかります。詳細については、AWS WAF デベロッパーガイドの「ウェブ ACL トラフィックのログ記録の料金に関する情報」を参照してください。

ユーザープールの大文字と小文字の区別

で作成した Amazon Cognito ユーザープール AWS Management Console では、デフォルトで大文字と小文字は区別されません。ユーザープールが大文字と小文字を区別しない場合、user@example.com と User@example.com は同じユーザーを参照します。ユーザープール内のユーザー名が大文字と小文字が区別しない場合、`preferred_username` と `email` 属性でも大文字と小文字が区別されません。

ユーザープールの大文字と小文字の区別の設定を考慮するには、代替のユーザー属性に基づいてアプリコードでユーザーを識別します。ユーザー名、優先ユーザー名、または E メールアドレス属性の

大文字と小文字はユーザープロファイルによって異なる場合があるため、代わりに sub 属性を参照してください。また、ユーザープールにイミュータブルカスタム属性を作成し、各新規ユーザープロファイルの属性に独自の一意の識別子値を割り当てることもできます。最初にユーザーを作成するときに、作成したイミュータブルカスタム属性に値を書き込むことができます。

> **Note**
>
> ユーザープールの大文字と小文字を区別する設定に関係なく、Amazon Cognito では、SAML または OIDC ID プロバイダー (IdP) のフェデレーティッドユーザーが、大文字と小文字を区別する一意の `NameId` または `sub` クレームを渡す必要があります。一意の識別子の大文字と小文字の区別と SAML の詳細については IdPs、「」を参照してくださいSP 開始 SAML サインインの使用。

大文字と小文字を区別するユーザープールの作成

AWS Command Line Interface（ AWS CLI) および などの API オペレーションを使用してリソースを作成する場合はCreateUserPool、ブール`CaseSensitive`パラメータを に設定する必要がありま`false`す。この設定では、大文字と小文字を区別しないユーザープールが作成されます。値を指定しないと、`CaseSensitive` はデフォルトで `true` に従います。このデフォルトは、AWS Management Consoleで作成するユーザープールのデフォルト動作とは反対です。2020 年 2 月 12 日以前のユーザープールでは、プラットフォームに関係なく、大文字と小文字が区別されていました。

AWS Management Console または DescribeUserPool API オペレーションのサインインエクスペリエンスタブを使用して、アカウント内の各ユーザープールの大文字と小文字の区別設定を確認できます。

新しいユーザープールに移行する

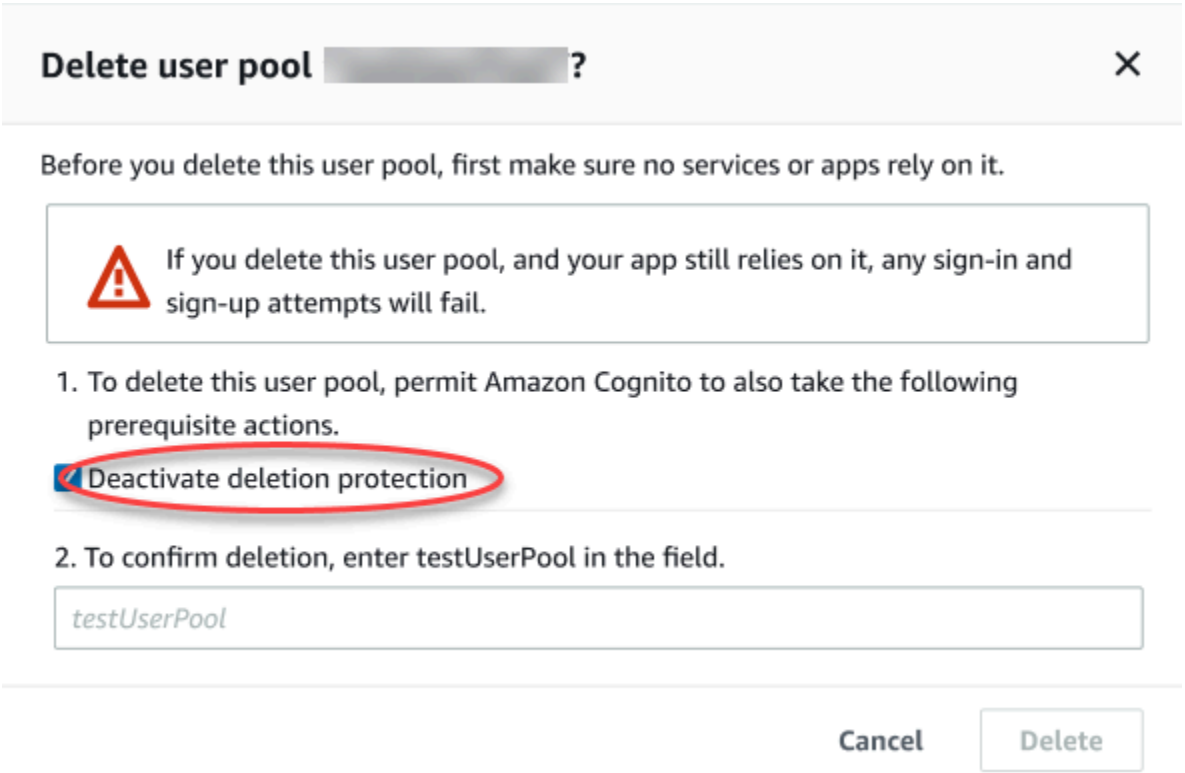
ユーザープロファイル間で競合が発生する可能性があるため、Amazon Cognito ユーザープールで大文字と小文字を区別するから大文字と小文字を区別しないに変更することはできません。代わりに、ユーザーを新しいユーザープールに移行します。ケース関連の競合を解決するには、移行コードをビルドする必要があります。このコードは、競合を検出したときに一意の新しいユーザーを返すか、サインイン試行を拒否する必要があります。大文字と小文字を区別しない新しいユーザープールで、ユーザー移行の Lambda トリガー を割り当てます。 AWS Lambda 関数は、大文字と小文字を区別しない新しいユーザープールにユーザーを作成できます。ユーザーが大文字小文字を区別しないユーザープールでサインインに失敗した場合、Lambda 関数は大文字小文

字を区別するユーザープールからユーザーを見つけて、複製します。ForgotPassword イベントでユーザーの Lambda 移行トリガーをアクティブ化することもできます。Amazon Cognito は、サインインやパスワード回復アクションからユーザー情報とイベントメタデータを Lambda 関数に渡します。関数が大文字と小文字を区別しないユーザープールに新しいユーザーを作成する際、イベントデータを使用してユーザー名とメールアドレス間で発生する競合を管理することができます。これらの競合は、大文字小文字を区別しないユーザープールでは一意ですが、大文字小文字を区別するユーザープールでは同一のユーザー名と E メールアドレスの間で発生する競合です。

Amazon Cognito ユーザープール間でユーザー移行 Lambda トリガーを使用する方法の詳細については、 AWS ブログの「Migrating Users to Amazon Cognito user pools」を参照してください。

ユーザープールの削除保護

管理者が誤ってユーザープールを削除しないようにするには、削除保護を有効にします。削除保護が有効になっている場合は、ユーザープールを削除する前に削除の確認を行う必要があります。AWS Management Console でユーザープールを削除すると、同時に削除保護を無効化できます。次の画像に示すように、削除保護を無効にするプロンプトに応じて削除の意思を確認すると、Amazon Cognito がユーザープールを削除します。



Amazon Cognito API リクエストを使用してユーザープールを削除する場合は、まず UpdateUserPool リクエストで DeletionProtection を Inactive に変更する必要があります。削除保護を無効にしないと、Amazon Cognito は InvalidParameterException エラーを返します。削除保護を無効にすると、DeleteUserPool リクエストでユーザープールを削除できます。

Amazon Cognito は、AWS Management Console で新しいユーザープールの作成時に、デフォルトで削除保護を有効にします。CreateUserPool API を使用してユーザープールを作成すると、削除保護はデフォルトで無効になります。AWS CLI または AWS SDK で作成したユーザープールでこの機能を使用するには、DeletionProtection パラメータを True に設定します。

Amazon Cognito コンソールの [User pool settings] (ユーザープール設定) タブにある [Deletion protection] (削除保護) コンテナで、削除保護ステータスを有効または無効にすることができます。

削除保護を設定するには

1. Amazon Cognito コンソールに移動します。AWS 認証情報を求められる場合があります。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [User pool settings] (ユーザープール設定) タブを選択します。[Deletion Protection] (削除保護) を見つけ、[Activate] (有効化) または [Deactivate] (無効化) を選択します
5. 次のダイアログで選択を確定します。

ユーザー存在エラー応答の管理

Amazon Cognito は、ユーザープールから返されるエラーレスポンスのカスタマイズをサポートしています。カスタムエラーレスポンスは、ユーザーの作成と認証、パスワードリカバリ、および確認操作に利用できます。

ユーザープールアプリクライアントの PreventUserExistenceErrors 設定を使用して、ユーザーの存在に関連するエラーを有効または無効にします。Amazon Cognito ユーザープール API を使用して新しいアプリケーションクライアントを作成する場合、 PreventUserExistenceErrorsはデフォルトで であるかLEGACY、無効になっています。Amazon Cognito コンソールでは、デフォルトでユーザー存在エラーの防止 - ENABLED の の設定 PreventUserExistenceErrors- オプションが選択されています。PreventUserExistenceErrors 設定を更新するには、次のいずれかを実行します。

- UpdateUserPoolClient API リクエストLEGACYで ENABLED から PreventUserExistenceErrorsまでの値を変更します。

- Amazon Cognito コンソールでアプリクライアントを編集し、選択した (ENABLED) と選択解除された () の間でユーザー存在エラーの防止の状態を変更しますLEGACY。

このプロパティの値が の場合LEGACY、ユーザーがユーザープールに存在しないユーザー名でサインインしようとすると、アプリクライアントはUserNotFoundExceptionエラーレスポンスを返します。

このプロパティの値が の場合ENABLED、アプリクライアントはユーザープールにユーザーアカウントが存在しないことをUserNotFoundExceptionエラーで公開しません。PreventUserExistenceErrors の設定ENABLEDには次の効果があります。

- Amazon Cognito は、API リクエストに対して非固有の情報で応答します。それ以外の場合、その応答によって有効なユーザーが存在することが開示される可能性があります。
- Amazon Cognito サインイン APIsが返されます。エラーレスポンスは、ユーザー名またはパスワードが正しくないことを伝えます。
- Amazon Cognito アカウント確認およびパスワード復旧 APIsは、ユーザーの連絡先情報の一部を表すのではなく、シミュレートされた配信メディアにコードが送信されたことを示すレスポンスを返します。

次の情報は、 が PreventUserExistenceErrorsに設定されている場合のユーザープールオペレーションの動作の詳細を示していますENABLED。

認証およびユーザー作成オペレーション

エラーレスポンスは、Username-password 認証と Secure Remote Password (SRP) 認証の両方で設定できます。カスタム認証で返すエラーをカスタマイズすることもできます。次の APIsこれらの認証オペレーションを実行します。

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- InitiateAuth
- RespondToAuthChallenge

次のリストは、ユーザー認証オペレーションのエラーレスポンスをカスタマイズする方法を示しています。

ユーザー名およびパスワード認証

ADMIN_USER_PASSWORD_AUTH および USER_PASSWORD_AUTH でユーザーをサインインするには、ユーザー名とパスワードを AdminInitiateAuth または InitiateAuth API リクエストに含めます。Amazon Cognito は、ユーザー名またはパスワードが正しくない場合、汎用 NotAuthorizedException エラーを返します。

セキュアリモートパスワード (SRP) ベースの認証

USER_SRP_AUTH でユーザーをサインインするには、AdminInitiateAuth または InitiateAuth API リクエストにユーザー名と SRP_A パラメータを含めます。それに応じて、Amazon Cognito はユーザーの SRP_Bと ソルトを返します。ユーザーが見つからない場合、RFC 5054 で説明されているように、Amazon Cognito は最初のステップでシミュレートされたレスポンスを返します。Amazon Cognito は、同じユーザー名とユーザープールの組み合わせに対して、同じ salt と汎用一意識別子 (UUID) 形式の内部ユーザー ID を返します。パスワードの証明を含む RespondToAuthChallenge API リクエストを送信すると、Amazon Cognito はユーザー名またはパスワードのいずれかが正しくない場合に一般的な NotAuthorizedException エラーを返します。

> **Note**
>
> 検証ベースのエイリアス属性を使用していて、変更不可能なユーザー名の形式が UUID でない場合は、ユーザー名とパスワードの認証で汎用レスポンスをシミュレートできます。

カスタム認証チャレンジの Lambda トリガー

カスタム認証チャレンジの Lambda トリガーを使用しており、エラーレスポンスを有効にする場合は、LambdaChallenge が UserNotFound という名前のブールパラメータを返します。その後、それが DefineAuthChallenge、VerifyAuthChallenge、および CreateAuthChallenge Lambda トリガーのリクエストに渡されます。このトリガーを使用して、存在しないユーザーのカスタム認証チャレンジをシミュレートできます。存在しないユーザーに対して事前認証の Lambda トリガーを呼び出す場合、Amazon Cognito は UserNotFound を返します。

次のリストは、ユーザー作成オペレーションのエラーレスポンスをカスタマイズする方法を示しています。

SignUp

ユーザー名が既に取得UsernameExistsExceptionされている場合、 SignUpオペレーションは常に を返します。アプリケーションでユーザーをサインアップするときに Amazon Cognito が E メールアドレスと電話番号の UsernameExistsException エラーを返さないようにするには、検証ベースのエイリアス属性を使用してください。エイリアスの詳細については、「ログイン属性のカスタマイズ」を参照してください。

Amazon Cognito がSignUp API リクエストを使用してユーザープール内のユーザーを検出できないようにする方法の例については、「サインアップ時のメールアドレスと電話番号の UsernameExistsException エラーの防止」を参照してください。

インポート済みユーザー

PreventUserExistenceErrors が有効になっている場合は、インポートされたユーザーの認証中、PasswordResetRequiredException を返す代わりに、ユーザー名またはパスワードが正しくなかったことを示す NotAuthorizedException エラーが返されます。詳細については、「インポートされたユーザーに対するパスワードのリセットの要求」を参照してください。

ユーザー移行の Lambda トリガー

Lambda トリガーによって元のイベントコンテキストに空のレスポンスが設定された場合、Amazon Cognito は存在しないユーザーについてシミュレートされたレスポンスを返します。詳細については、「ユーザー移行の Lambda トリガー」を参照してください。

サインアップ時のメールアドレスと電話番号の **UsernameExistsException** エラーの防止

次の例は、ユーザープールでエイリアス属性を設定するときに、重複する E メールアドレスと電話番号が SignUp API リクエストに応答して UsernameExistsException エラーを生成しないようにする方法を示しています。E メールアドレスまたは電話番号をエイリアス属性として使用してユーザープールを作成しておく必要があります。詳細については、ユーザープール属性の「エイリアス」セクションを参照してください。

1. Jie は新しいユーザー名にサインアップし、E メールアドレス jie@example.com も提供します。Amazon Cognito がユーザーの E メールアドレスにコードを送信します。

   AWS CLI コマンドの例

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

レスポンスの例

```
{
    "UserConfirmed": false,
    "UserSub": "<subId>",
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "Destination": "j****@e****",
        "DeliveryMedium": "EMAIL"
    }
}
```

2. Jie は、E メールアドレスの所有権を確認するために送信されたコードを提供します。これで、ユーザーとしての登録は完了です。

   AWS CLI コマンドの例

```
aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
confirmation-code xxxxxx
```

3. Shirley は新しいユーザーアカウントを登録し、E メールアドレス `jie@example.com` を提供します。Amazon Cognito は `UsernameExistsException` エラーを返さず、確認コードを Jie の E メールアドレスに送信します。

   AWS CLI コマンドの例

```
aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
 PASSWORD --user-attributes Name="email",Value="jie@example.com"
```

   レスポンスの例

```
{
    "UserConfirmed": false,
    "UserSub": "<new subId>",
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "Destination": "j****@e****",
        "DeliveryMedium": "EMAIL"
    }
}
```

4. 別のシナリオでは、Shirley が jie@example.com の所有権を持っています。Shirley は Amazon Cognito が Jie の E メールアドレスに送信したコードを取得し、アカウントの確認を試みます。

   AWS CLI コマンドの例

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   レスポンスの例

   ```
   An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
    account with the email already exists.
   ```

jie@example.com が既存のユーザーに割り当てられているにもかかわらず、Amazon Cognito は Shirley の aws cognito-idp sign-up リクエストにエラーを返しません。Amazon Cognito がエラーレスポンスを返す前に、Shirley は E メールアドレスの所有権を証明する必要があります。エイリアス属性を持つユーザープールでは、この動作により、パブリック SignUp API を使用して、特定の E メールアドレスまたは電話番号を持つユーザーが存在するかどうかを確認できなくなります。

この動作は、次の例に示すように、Amazon Cognito が既存のユーザー名の SignUp リクエストに対して返すレスポンスとは異なります。Shirley はこのレスポンスから、そのユーザー名 jie を持つユーザーが既に存在することを知っていますが、そのユーザーに関連する E メールアドレスや電話番号については知りません。

CLI コマンドの例

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

レスポンスの例

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User
 already exists
```

パスワードのリセットオペレーション

ユーザー存在エラーを防ぐと、Amazon Cognito は、ユーザーパスワードのリセット操作に対して以下の応答を返します。

ForgotPassword

ユーザーが見つからない、非アクティブ化されている、またはパスワードを回復するための検証済みの配信メカニズムがない場合、Amazon Cognito はそのユーザーについてシミュレート済みの配信ミディアムを用いた `CodeDeliveryDetails` を返します。シミュレートされた配信メディアは、入力ユーザー名形式とユーザープールの検証設定によって決まります。

ConfirmForgotPassword

Amazon Cognito は、存在しない、または無効になっているユーザーについて `CodeMismatchException` エラーを返します。`ForgotPassword` の使用時にコードが要求されない場合、Amazon Cognito は `ExpiredCodeException` エラーを返します。

確認オペレーション

ユーザー存在エラーを防ぐと、Amazon Cognito は、ユーザーの確認および検証の操作に対して以下の応答を返します。

ResendConfirmationCode

Amazon Cognito は、無効化されたユーザー、または存在しないユーザーについて `CodeDeliveryDetails` を返します。Amazon Cognito は、既存ユーザーの E メールまたは電話番号に確認コードを送信します。

ConfirmSignUp

コードの有効期限が切れている場合は、`ExpiredCodeException` が返されます。Amazon Cognito は、ユーザーが承認されていない場合に `NotAuthorizedException` を返します。コードがサーバーが期待するものと一致しない場合、Amazon Cognito は `CodeMismatchException` を返します。

Amazon Cognito アイデンティティプール

Amazon Cognito アイデンティティプールは、 AWS 認証情報と交換できるフェデレーションアイデンティティのディレクトリです。ID プールは、サインインしているか、まだ識別していないかにかかわらず、アプリケーションのユーザーに一時的な AWS 認証情報を生成します。 AWS Identity and Access Management (IAM) ロールとポリシーを使用すると、ユーザーに付与するアクセス許可のレベルを選択できます。ユーザーはゲストとしてスタートし、 AWS のサービス で保管しているアセットを取得できます。その後、サードパーティー ID プロバイダーにサインインして、登録メンバーに提供したアセットへのアクセスをロック解除できます。サードパーティーの ID プロバイダーには、Apple や Google などのコンシューマー (ソーシャル) OAuth 2.0 プロバイダー、カスタム SAML または OIDC ID プロバイダー、またはお客様が独自に設計したカスタム認証スキーム (開発者プロバイダーとも呼ばれます) があります。

Amazon Cognito アイデンティティプールの機能

のリクエストに署名する AWS のサービス

Amazon Simple Storage Service (Amazon S3) や Amazon DynamoDB AWS のサービス などの に API リクエストに署名します。Amazon Pinpoint や Amazon などのサービスでユーザーアクティビティを分析します CloudWatch。

リソースベースのポリシーを使用してリクエストをフィルタリングする

リソースへのユーザーアクセスをきめ細かに制御します。ユーザークレームを IAMセッションタグに変換し、ユーザーの個別のサブセットにリソースアクセスを許可する IAM ポリシーを構築します。

ゲストアクセスを割り当てる

まだサインインしていないユーザーの場合は、アクセス範囲の狭い AWS 認証情報を生成するようにアイデンティティプールを設定します。シングルサインオンプロバイダーを通じてユーザーを認証し、アクセスを強化します。

ユーザーの特性に基づいて IAM ロールを割り当てる

認証されたすべてのユーザーに 1 つの IAM ロールを割り当てるか、各ユーザーのリクエストに基づいてロールを選択します。

さまざまな ID プロバイダーを受け入れる

ID またはアクセストークン、ユーザープールトークン、SAML アサーション、またはソーシャルプロバイダーの OAuth トークンを AWS 認証情報と交換します。

自分のアイデンティティを検証する

独自のユーザー検証を実行し、デベロッパーの AWS 認証情報を使用してユーザーの認証情報を発行します。

アプリに認証および承認サービスを提供する Amazon Cognito ユーザープールがすでにある場合があります。ユーザープールは ID プロバイダー (IdP) としてユーザープールに設定できます。これを行うと、ユーザーはユーザープール を介して認証し IdPs、クレームを共通の OIDC ID トークンに統合し、そのトークンを AWS 認証情報と交換できます。その後、ユーザーは署名付きのリクエストで認証情報を AWS のサービス に提示できます。

また、任意の ID プロバイダーからの認証済みクレームをアイデンティティプールに直接提示することもできます。Amazon Cognito は、SAML、OAuth、および OIDC プロバイダーからのユーザークレームを短期認証情報の AssumeRoleWithWebIdentity API リクエストにカスタマイズします。

Amazon Cognito ユーザープールは、SSO 対応アプリケーションの OIDC ID プロバイダーのようなものです。アイデンティティプールは、IAM 認証に最も適したリソース依存関係を持つすべてのアプリの AWS ID プロバイダーとして機能します。

Amazon Cognito ID プールは、以下の ID プロバイダーをサポートします。

- パブリックプロバイダー:Login with Amazon を ID プール IdP としてセットアップする、Facebook を ID プール IdP としてセットアップする、Google を ID プール IdP としてセットアップする、アイデンティティプール IdP として「Apple でサインイン」を設定する、Twitter。
- Amazon Cognito ユーザープール
- OIDC プロバイダーを ID プール IdP としてセットアップする
- SAML プロバイダーを ID プール IdP としてセットアップする
- デベロッパーが認証したアイデンティティ (アイデンティティプール)

Amazon Cognito ID プールを利用できるリージョンの詳細については、「AWS リージョン別のサービス」を参照してください。

Amazon Cognito ID プールの詳細については、以下のトピックを参照してください。

トピック

- ID プール (フェデレーティッド ID) の使用
- ID プールの概念

- Amazon Cognito ID プールのセキュリティのベストプラクティス
- アクセスコントロールへの属性の使用
- ロールベースアクセスコントロールの使用
- 認証情報の取得
- AWS サービスへのアクセス
- ID プール外部 ID プロバイダー
- デベロッパーが認証したアイデンティティ (アイデンティティプール)
- 認証されていないユーザーから認証されたユーザー (ID プール) への切り替え

ID プール (フェデレーティッド ID) の使用

Amazon Cognito ID プールは、ゲストであるユーザー (認証されていないユーザー) と、認証されてトークンを受け取ったユーザーに一時的な AWS 認証情報を提供します。ID プールはアカウントに固有のユーザー ID データのストアです。

コンソールで新しい ID プールを作成する

1. Amazon Cognito コンソールにサインインし、アイデンティティプールを選択します。
2. [ID プールを作成] を選択します。
3. [ID プールの信頼を設定] で、アイデンティティプールを認証アクセス、ゲストアクセス、またはその両方に設定することを選択します。

    - [認証アクセス] を選択した場合、アイデンティティプールの認証済み ID のソースとして設定する ID タイプを 1 つ以上選択します。[カスタムデベロッパープロバイダー] を設定した場合、アイデンティティプールの作成後にそのプロバイダーを変更したり削除したりすることはできません。
4. [許可を設定] で、アイデンティティプール内の認証済みユーザーまたはゲストユーザーのデフォルトの IAM ロールを選択します。

    a. Amazon Cognito に、基本的な権限とアイデンティティプールとの信頼関係を持つ新しいロールを作成する場合は、[新しい IAM ロールを作成] を選択します。新しいロールを識別するための IAM ロール名を入力します (たとえば `myidentitypool_authenticatedrole`)。[ポリシードキュメントを表示] を選択して、Amazon Cognito が新しい IAM ロールに割り当てるアクセス権限を確認します。

b. 使用する にロールが既にある場合は、既存の IAM ロール AWS アカウント を使用することを選択できます。`cognito-identity.amazonaws.com` を含めるように IAM ロールの信頼ポリシーを設定する必要があります。リクエストが特定のアイデンティティプール内の認証されたユーザーから送信されたという証拠を提示した場合にのみ、Amazon Cognito がロールを引き継ぐことを許可するようにロールの信頼ポリシーを設定します。詳細については、「ロールの信頼とアクセス権限」を参照してください。

5. ID プロバイダーの接続 で、ID プールの信頼を設定する で選択した ID プロバイダー (IdPs) の詳細を入力します。OAuth アプリケーションクライアント情報の提供、Amazon Cognito ユーザープールの選択、IAM IdP の選択、またはディベロッパープロバイダーのカスタム識別子の入力を求められる場合があります。

   a. 各 IdP の [ロール設定] を選択します。その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。Amazon Cognito ユーザープール IdP では、トークンに preferred_role を含むロールを選択することもできます。`cognito:preferred_role` クレームの詳細については、「グループへの優先順位の値の割り当て」を参照してください。

      i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

      ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

   b. アクセスコントロールの属性は、IdP ごとに個別に設定できます。アクセスコントロールの属性は、Amazon Cognito がユーザーの一時セッションに適用するプリンシパルタグにユーザーのクレームをマッピングします。セッションに適用するタグに基づいてユーザーアクセスをフィルタリングするIAMポリシーを作成できます。

      i. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

      ii. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

      iii. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

6. [プロパティの設定] の [ID プール名] に [名前] を入力します。

7. [基本 (クラシック) 認証] で、ベーシックフローを有効にするかどうかを選択します。基本的なフローをアクティブにすると、 に対して行ったロールの選択をバイパス IdPs して、AssumeRoleWithWebIdentity を直接呼び出すことができます。詳細については、「ID プール (フェデレーティッドアイデンティティ) の認証フロー」を参照してください。
8. アイデンティティプールにタグを適用する場合は、[タグ] で [タグの追加] を選択します。
9. [確認および作成] で、新しいアイデンティティプールに対して行った選択を確認します。[編集] を選択してウィザードに戻り、設定を変更します。終了したら、[ID プールの作成] を選択します。

ユーザー IAM ロール

IAM ロールは、ユーザーが などの AWS リソースにアクセスするためのアクセス許可を定義しますAmazon Cognito Sync。アプリケーションのユーザーは、作成されたロールを引き受けます。認証されたユーザーと認証されていないユーザー用に、異なるロールを指定できます。IAM ロールの詳細については、「IAM ロール」を参照してください。

認証された ID と認証されていない ID

Amazon Cognito ID プールは、認証された ID と認証されていない ID の両方をサポートします。認証された ID は任意のサポートされている認証プロバイダーで認証されたユーザーに属します。通常、認証されていない ID はゲストユーザーに属します。

- パブリックログインプロバイダーに対して、認証された ID を設定する方法については、「ID プール外部 ID プロバイダー」を参照してください。
- 独自のバックエンド認証プロセスを設定するには、「デベロッパーが認証したアイデンティティ (アイデンティティプール)」を参照してください。

ゲストアクセスをアクティブ化/非アクティブ化する

Amazon Cognito ID プールのゲストアクセス (認証されていない ID) は、ID プロバイダーで認証しないユーザーに一意の識別子と AWS 認証情報を提供します。アプリケーションで、ログインしないユーザーを許可している場合、認証されていない ID 用にアクセスを許可できます。詳細については、「Amazon Cognito ID プールの開始方法」を参照してください。

アイデンティティプールのゲストアクセスを更新するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. ゲストアクセスを検索します。現在ゲストアクセスをサポートしていないアイデンティティプールでは、ステータスが非アクティブです。

   a. ゲストアクセスがアクティブで、非アクティブ化したい場合は、[非アクティブ化] を選択します。

   b. ゲストアクセスが非アクティブで、アクティブにしたい場合は、[編集] を選択します。

      - アイデンティティプールのゲストユーザーのデフォルトの IAM ロールを選択します。

        A. Amazon Cognito に、基本的な権限とアイデンティティプールとの信頼関係を持つ新しいロールを作成する場合は、[新しい IAM ロールを作成] を選択します。新しいロールを識別するための IAM ロール名を入力します (たとえば `myidentitypool_authenticatedrole`)。[ポリシードキュメントを表示] を選択して、Amazon Cognito が新しい IAM ロールに割り当てるアクセス権限を確認します。

        B. 使用する にロールがすでにある場合は、既存の IAM ロール AWS アカウント を使用するように選択できます。`cognito-identity.amazonaws.com` を含めるように IAM ロールの信頼ポリシーを設定する必要があります。リクエストが特定のアイデンティティプール内の認証されたユーザーから送信されたという証拠を提示した場合にのみ、Amazon Cognito がロールを引き継ぐことを許可するようにロールの信頼ポリシーを設定します。詳細については、「ロールの信頼とアクセス権限」を参照してください。

        C. [変更を保存] を選択します。

        D. ゲストアクセスを有効にするには、[ユーザーアクセス] タブで [アクティブ化] を選択します。

ID の種類に関連付けられたロールを変更する

ID プールのすべての ID は、認証されているか、認証されていないかのいずれかです。認証された ID は、パブリックログインプロバイダー (Amazon Cognito ユーザープール、Login with Amazon、「Apple でサインイン」、Facebook、Google、SAML、または任意の OpenID Connect プロバイ

ダー)、またはデベロッパープロバイダー (独自のバックエンド認証プロセス) によって認証されたユーザーに属します。通常、認証されていない ID はゲストユーザーに属します。

各 ID の種類について、割り当てられたロールがあります。このロールには、 AWS のサービス そのロールがアクセスできるロールを指定するポリシーがアタッチされています。リクエストを受け取った Amazon Cognito は、ID タイプとその ID タイプに割り当てられたロールを判別し、そのロールにアタッチされたポリシーを使用して応答を返します。ポリシーを変更するか、アイデンティティタイプに別のロールを割り当てることで、どのアイデンティティタイプにアクセスできる AWS のサービス かを制御できます。ID プールのロールに関連付けられたポリシーの表示または変更については、「AWS IAM コンソール」を参照してください。

アイデンティティプールのデフォルトの認証済みロールまたは非認証ロールを変更するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ゲストアクセス] または [認証されたアクセス] を見つけます。現在そのアクセスタイプに設定されていないアイデンティティプールでは、ステータスが非アクティブです。[Edit] (編集) を選択します。
4. アイデンティティプールのゲストまたは認証されたユーザーのデフォルトの IAM ロールを選択します。

   a. Amazon Cognito に、基本的な権限とアイデンティティプールとの信頼関係を持つ新しいロールを作成する場合は、[新しい IAM ロールを作成] を選択します。新しいロールを識別するための IAM ロール名を入力します (たとえば `myidentitypool_authenticatedrole`)。[ポリシードキュメントを表示] を選択して、Amazon Cognito が新しい IAM ロールに割り当てるアクセス権限を確認します。

   b. 使用する にロールが既にある場合は、既存の IAM ロール AWS アカウント を使用することを選択できます。`cognito-identity.amazonaws.com` を含めるように IAM ロールの信頼ポリシーを設定する必要があります。リクエストが特定のアイデンティティプール内の認証されたユーザーから送信されたという証拠を提示した場合にのみ、Amazon Cognito がロールを引き継ぐことを許可するようにロールの信頼ポリシーを設定します。詳細については、「ロールの信頼とアクセス権限」を参照してください。

5. [変更を保存] を選択します。

ID プロバイダーを編集する

コンシューマー ID プロバイダー (Amazon Cognito ユーザープール、Login with Amazon、Apple でサインイン、Facebook、Google など) を使用した認証をユーザーに許可している場合は、Amazon Cognito ID プール (フェデレーテッドアイデンティティ) のコンソールでアプリケーション識別子を指定できます。これにより、アプリケーション ID (パブリックログインプロバイダーによって提供) が ID プールに関連付けられます。

このページからプロバイダーごとの認証ルールを設定することもできます。プロバイダーごとに、最大 25 個のルールを割り当てることができます。ルールは、プロバイダーごとに保存した順序で適用されます。詳細については、「ロールベースアクセスコントロールの使用」を参照してください。

> **Warning**
>
> アイデンティティプールでリンクされた IdP アプリケーション ID を変更すると、そのアイデンティティプールによる既存ユーザーの認証ができなくなります。詳細については、「ID プール外部 ID プロバイダー」を参照してください。

アイデンティティプール ID プロバイダー (IdP) を更新するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. ID プロバイダーを見つけます。編集する ID プロバイダーを選択します。新しい IdP を追加する場合は、[ID プロバイダーを追加] を選択します。
   - [ID プロバイダーを追加] を選択した場合は、追加する ID タイプのいずれかを選択します。
4. アプリケーション ID を変更するには、[アイデンティティプロバイダーに関する情報] で [編集] を選択します。
5. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを変更するには、[ロール設定] で [編集] を選択します。
   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。Amazon Cognito ユーザープール IdP では、トークンに preferred_role を含むロールを選択することもできます。`cognito:preferred_role` クレームの詳細については、「グループへの優先順位の値の割り当て」を参照してください。

i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] で [編集] を選択します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

7. [変更を保存] を選択します。

アイデンティティプールを削除する

アイデンティティプールの削除は元に戻すことができません。アイデンティティプールを削除すると、そのプールに依存するすべてのアプリとユーザーは機能しなくなります。

ID プールを削除するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。削除するアイデンティティプールの横にあるラジオボタンをオンにします。
2. [削除] を選択します。
3. アイデンティティプールの名前を入力または貼り付けて、[削除] を選択します。

> **Warning**
>
> [Delete] (削除) ボタンをクリックすると、ID プールと、それに含まれているすべてのユーザーデータが完全に削除されます。アイデンティティプールを削除すると、そのアイデンティティプールを使用しているアプリケーションや他のサービスは動作を停止します。

ID プールから ID を削除する

アイデンティティプールから ID を削除すると、Amazon Cognito がそのフェデレーティッドユーザーについて保存した識別情報が削除されます。ユーザーが再度認証情報をリクエストすると、アイデンティティプールが依然として ID プロバイダーを信頼していれば、そのユーザーは新しい ID を受け取ります。このオペレーションは元に戻すことができません。

ID を削除するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ID ブラウザー] タブを選択します。
3. 削除する ID プロバイダーの横にあるチェックボックスを選択してから、[削除] を選択します。ID を削除することを確認し、[削除] を選択します。

Amazon Cognito Sync を ID プールで使用する

Amazon Cognito Sync は、アプリケーション関連のユーザーデータをデバイス間で同期できるようにする AWS のサービス および クライアントライブラリです。Amazon Cognito Sync は、独自のバックエンドを使用せずに、モバイルデバイスとウェブ間でユーザープロファイルデータを同期できます。クライアントライブラリはローカルにデータをキャッシュするため、アプリはデバイスの接続状態にかかわらず、データを読み書きすることができます。デバイスがオンラインになると、データを同期できます。プッシュ同期を設定すると、更新が利用できることが他のデバイスにすぐに通知されます。

データセットの管理

アプリケーションに Amazon Cognito Sync 機能を実装している場合、Amazon Cognito ID プールコンソールで、個々のアイデンティティ用のデータセットとレコードを手動で作成したり、削除したりできます。Amazon Cognito ID プールコンソールで、アイデンティティ用のデータセットまたはレコードに対して行った変更は、コンソールの [Synchronize] (同期) をクリックするまで保存されません。この変更は、Synchronize が呼び出され、ID が同期されるでは、エンドユーザーに表示されません。特定のアイデンティティのデータセット一覧ページを更新すると、個別のアイデンティティについて他のデバイスから同期されたデータが表示されます。

アイデンティティのデータセットを作成する

Amazon Cognito Sync は、データセットを 1 つの ID に関連付けます。ID が表すユーザーに関する識別情報をデータセットに入力し、その情報をユーザーのすべてのデバイスに同期できます。

データセットとデータセットレコードを ID に追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ID ブラウザー] タブを選択します。
3. 編集する ID を選択します。
4. [データセット] で、[データセットの作成] を選択します。
5. データセット名を入力し、[データセットの作成] を選択します。
6. データセットにレコードを追加する場合は、ID の詳細からデータセットを選択します。[レコード] で、[レコードの作成] を選択します。
7. レコードの [キー] と [値] を入力します。[確認] を選択します。同じ手順を繰り返して、さらにレコードを追加します。

ID に関連付けられたデータセットを削除する

ID からデータセットとそのレコードを削除するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ID ブラウザー] タブを選択します。
3. 削除するデータセットを含む ID を選択します。
4. [データセット] で、削除するデータセットの横にあるラジオボタンを選択します。
5. [削除] を選択します。選択内容を確認して、もう一度 [削除] を選択します。

データの一括発行

一括発行を使用すると、既に Amazon Cognito Sync ストアに保存されているデータを、Amazon Kinesis ストリームにエクスポートできます。すべてのストリームを一括発行する手順については、「Amazon Cognito ストリーム」を参照してください。

プッシュ同期を有効にする

Amazon Cognito は自動的にアイデンティティとデバイス間の関係を追跡します。プッシュ同期機能を使用すると、ID データが変更されたときに、特定の ID のすべてのインスタンスに確実に通知されます。プッシュ同期は、ID のデータセットが変更されるたびに、その ID に関連付けられているすべてのデバイスが、この変更を伝えるサイレントプッシュ通知を受け取ることを確実にします。

プッシュ同期は Amazon Cognito コンソールでアクティブ化できます。

プッシュ同期をアクティブ化するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザープールのプロパティ] タブを選択します。
3. [プッシュ同期] で [編集] を選択します
4. [ID プールとのプッシュ同期を有効にする] を選択します。
5. 現在 AWS リージョン で作成した Amazon Simple Notification Service (Amazon SNS) プラットフォームアプリケーションから 1 つを選択します。Amazon Cognito はプッシュ通知をプラットフォームアプリケーションに公開します。[プラットフォームアプリケーションの作成] を選択して Amazon SNS コンソールに移動し、新しいコンソールを作成します。
6. プラットフォームアプリケーションにパブリッシュするには、Amazon Cognito が AWS アカウント で IAM ロールを引き受けます。Amazon Cognito に、基本的な権限とアイデンティティプールとの信頼関係を持つ新しいロールを作成する場合は、[新しい IAM ロールを作成] を選択します。新しいロールを識別するための IAM ロール名を入力します (たとえば `myidentitypool_authenticatedrole`)。[ポリシードキュメントを表示] を選択して、Amazon Cognito が新しい IAM ロールに割り当てるアクセス権限を確認します。
7. 使用する にロールが既にある場合は、既存の IAM ロール AWS アカウント の使用を選択できます。`cognito-identity.amazonaws.com` を含めるように IAM ロールの信頼ポリシーを設定する必要があります。リクエストが特定のアイデンティティプール内の認証されたユーザーから送信されたという証拠を提示した場合にのみ、Amazon Cognito がロールを引き継ぐことを許可するようにロールの信頼ポリシーを設定します。詳細については、「ロールの信頼とアクセス権限」を参照してください。
8. [変更を保存] を選択します。

Amazon Cognito Streams をセットアップする

Amazon Cognito Streams は、Amazon Cognito Sync に保存されているデータに対する制御と洞察をデベロッパーに提供します。これで、デベロッパーはイベントをデータとして受け取るために Kinesis ストリームを設定できるようになります。Amazon Cognito は、所有する Kinesis ストリームに各データセットの変更をリアルタイムでプッシュできます。Amazon Cognito コンソールで Amazon Cognito Streams をセットアップする方法の手順については、「Amazon Cognito ストリーム」を参照してください。

Amazon Cognito Events をセットアップする

Amazon Cognito Events では、Amazon Cognito Sync の重要なイベントに応じて AWS Lambda 関数を実行できます。Amazon Cognito Sync は、データセットが同期されるときに Sync Trigger イベントを生成します。Sync Trigger イベントを使用して、ユーザーがデータを更新するときにアクションを実行できます。コンソールから Amazon Cognito Events をセットアップする手順については、「Amazon Cognito イベント」を参照してください。

の詳細については、 AWS Lambda「」を参照してくださいAWS Lambda。

ID プールの概念

Amazon Cognito ID プールを使用して、ユーザーのために一意のアイデンティティを作成し、ID プロバイダーでそれらを認証することができます。ID を使用すると、権限が制限された一時的な AWS 認証情報を取得して、他の にアクセスできます AWS のサービス。Amazon Cognito ID プールは、Amazon、Apple、Facebook、および Google などのパブリック ID プロバイダーだけでなく、認証されていない ID もサポートします。また、独自のバックエンド認証プロセスを通じてユーザーを登録し、認証することができる、開発者が認証した ID をサポートします。

Amazon Cognito ID プールを利用できるリージョンの詳細については、「AWS リージョン別のサービス」を参照してください。Amazon Cognito ID プールの概念に関する詳細については、以下のトピックを参照してください。

トピック

- ID プール (フェデレーティッドアイデンティティ) の認証フロー
- IAM ロール
- ロールの信頼とアクセス権限

ID プール (フェデレーティッドアイデンティティ) の認証フロー

Amazon Cognito は、エンドユーザーのために、デバイスおよびプラットフォーム全体で整合性が維持される一意の識別子を作成するために役立ちます。また、Amazon Cognito は、 AWS リソースにアクセスするために、権限が制限された一時的な認証情報をアプリケーションに配信します。このページでは、Amazon Cognito での認証の基礎と、ID プール内のアイデンティティのライフサイクルについて説明します。

外部プロバイダーの認証フロー

Amazon Cognito で認証されるユーザーは、その認証情報をブートストラップするために複数ステップのプロセスを経由します。Amazon Cognito には、パブリックプロバイダーでの認証に、拡張認証と基本認証の 2 つの異なるフローがあります。

これらのフローのいずれかを完了すると、ロールのアクセスポリシーで定義されている AWS のサービス 他の にアクセスできます。デフォルトでは、Amazon Cognito コンソールは、Amazon Cognito Sync ストアと Amazon Mobile Analytics へのアクセス権を持つロールを作成します。その他のアクセス権を付与する方法の詳細については、「IAM ロール」を参照してください。

ID プールは、プロバイダーからの次のアーティファクトを受け入れます。

プロバイダー	認証アーティファクト
Amazon Cognito ユーザープール	ID トークン
OpenID Connect (OIDC)	ID トークン
SAML 2.0	SAML アサーション
ソーシャルプロバイダー	アクセストークン

拡張 (簡略化) 認証フロー

拡張認証フローを使用する場合、アプリはまず、承認された Amazon Cognito ユーザープールまたはサードパーティー ID プロバイダーからの認証の証明をGetIdリクエストで提示します。

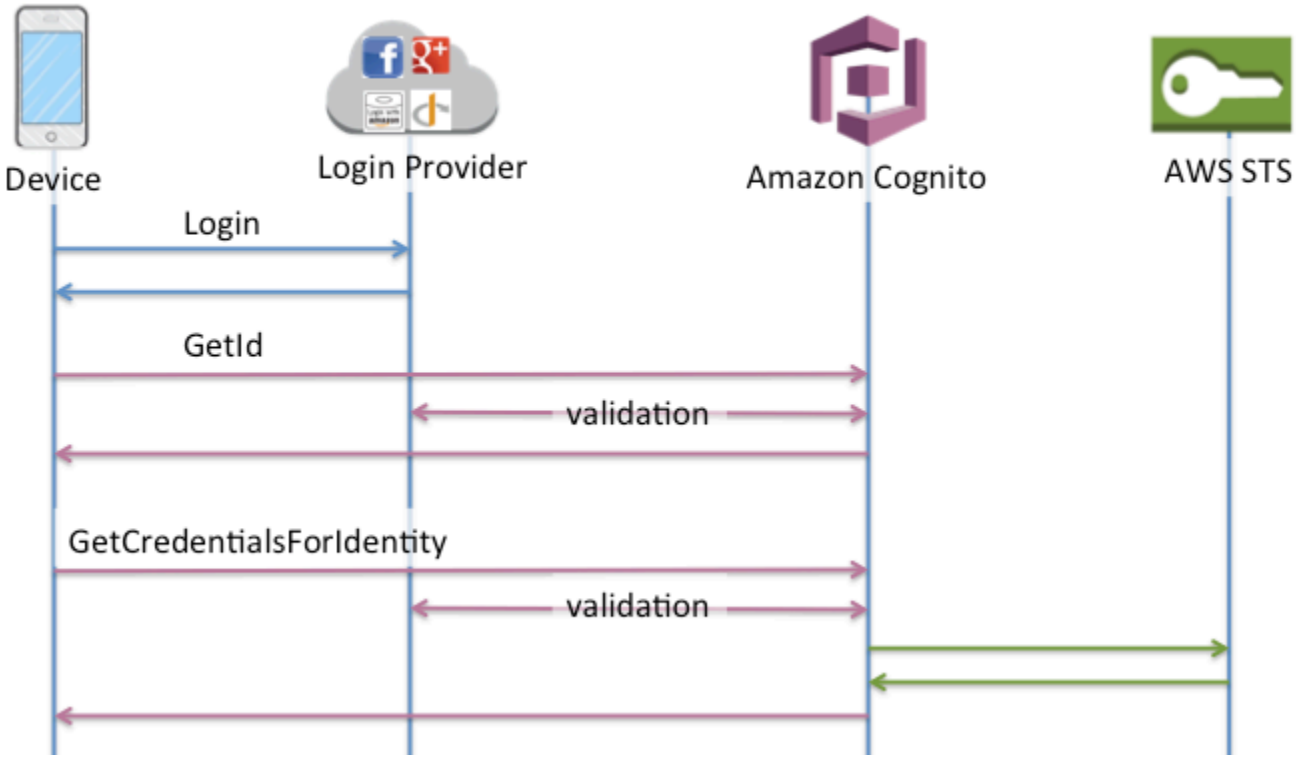
1. アプリケーションは、GetID リクエストで、認可された Amazon Cognito ユーザープールまたはサードパーティー ID プロバイダーからの JSON ウェブトークンまたは SAML アサーションなどの認証の証明を提示します。 GetID

2. ID プールは ID を返します。
3. アプリケーションは、ID を GetCredentialsForIdentity リクエスト内の同じ認証証明と組み合わせます。
4. ID プールは AWS 認証情報を返します。
5. アプリケーションは、一時的な認証情報を使用して AWS API リクエストに署名します。

拡張認証は、アイデンティティプール設定での IAM ロールの選択と認証情報の取得のロジックを管理します。デフォルトのロールを選択するように ID プールを設定して、属性ベースのアクセスコントロール (ABAC) またはロールベースのアクセスコントロール (RBAC) の原則をロール選択に適用できます。拡張認証の AWS 認証情報は 1 時間有効です。

拡張認証でのオペレーションの順序

1. `GetId`
2. `GetCredentialsForIdentity`



基本 (Classic) 認証フロー

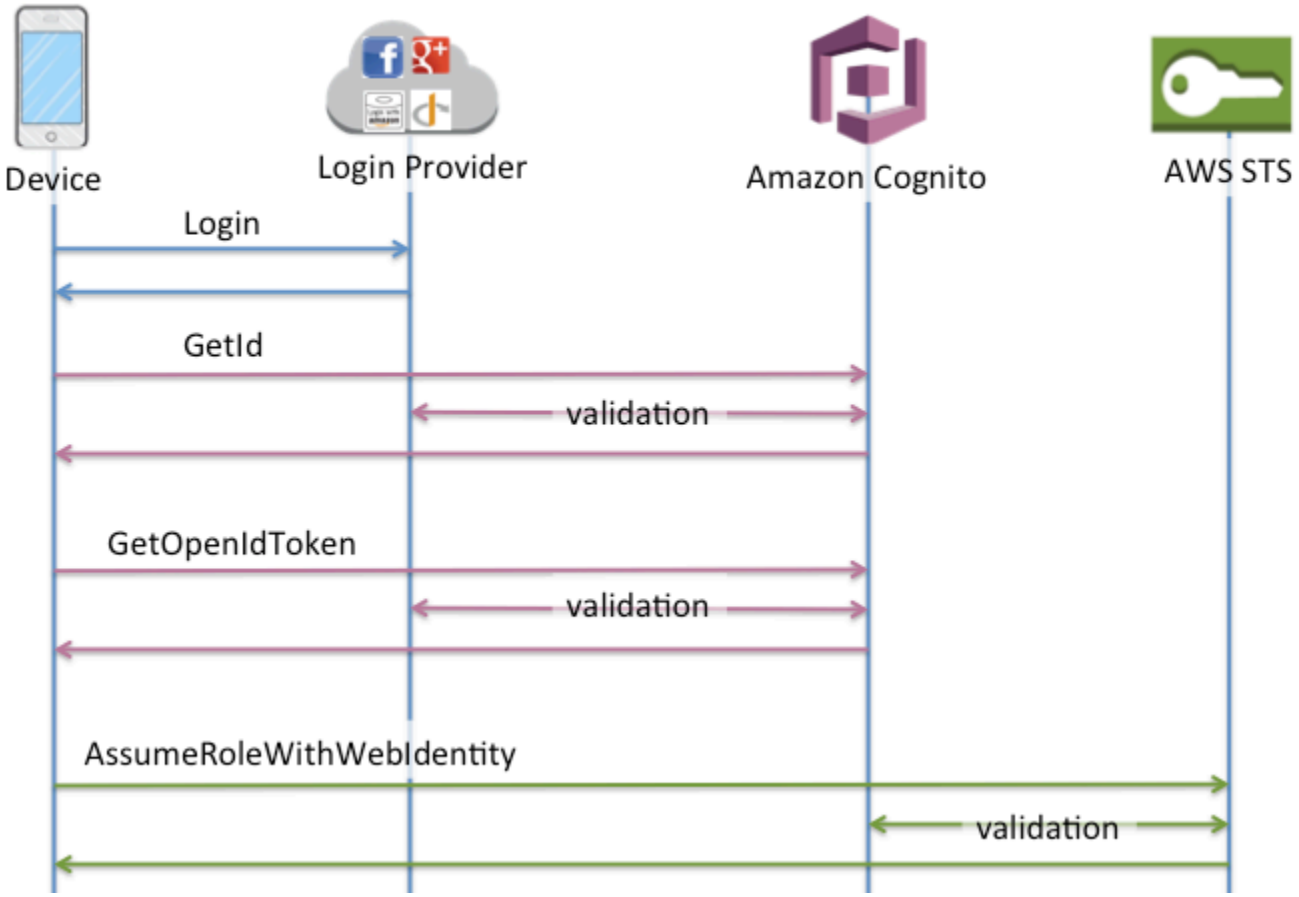
基本的な認証フローを使用すると、

1. アプリケーションは、GetID リクエストで、認可された Amazon Cognito ユーザープールまたはサードパーティー ID プロバイダーからの JSON ウェブトークンまたは SAML アサーションなどの認証の証明を提示します。 GetID
2. ID プールは ID を返します。
3. アプリケーションは、ID をGetOpenIdTokenリクエスト内の同じ認証証明と組み合わせます。
4. GetOpenIdToken は、ID プールによって発行された新しい OAuth 2.0 トークンを返します。
5. アプリケーションは新しいトークンを AssumeRoleWithWebIdentityリクエストに提示します。
6. AWS Security Token Service AWS STS) は AWS 認証情報を返します。
7. アプリケーションは、一時的な認証情報を使用して AWS API リクエストに署名します。

基本ワークフローでは、ユーザーに配布する認証情報をより細かく制御できます。拡張認証フローの GetCredentialsForIdentity リクエストは、アクセストークンの内容に基づいてロールをリクエストします。クラシックワークフローのAssumeRoleWithWebIdentityリクエストにより、十分な信頼ポリシーで設定した AWS Identity and Access Management ロールの認証情報をリクエストする機能がアプリに付与されます。カスタムロールセッション期間をリクエストすることもできます。

基本認証でのオペレーションの順序

1. GetId
2. GetOpenIdToken
3. AssumeRoleWithWebIdentity



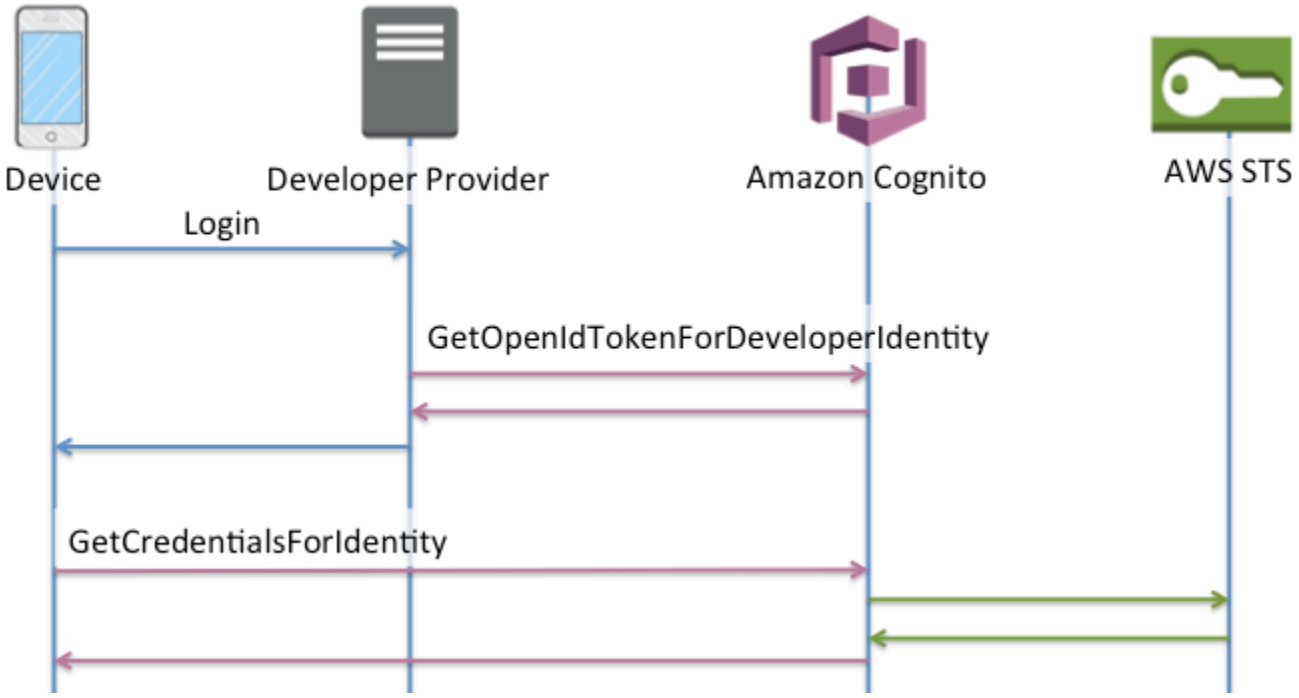
デベロッパーが認証した ID の認証フロー

デベロッパーが認証したアイデンティティ (アイデンティティプール) を使用する場合、クライアントは独自の認証システムでユーザーを検証するために、Amazon Cognito 外部のコードが含まれる異なる認証フローを使用します。Amazon Cognito 外部のコードは、外部のものであることが示されます。

拡張認証フロー

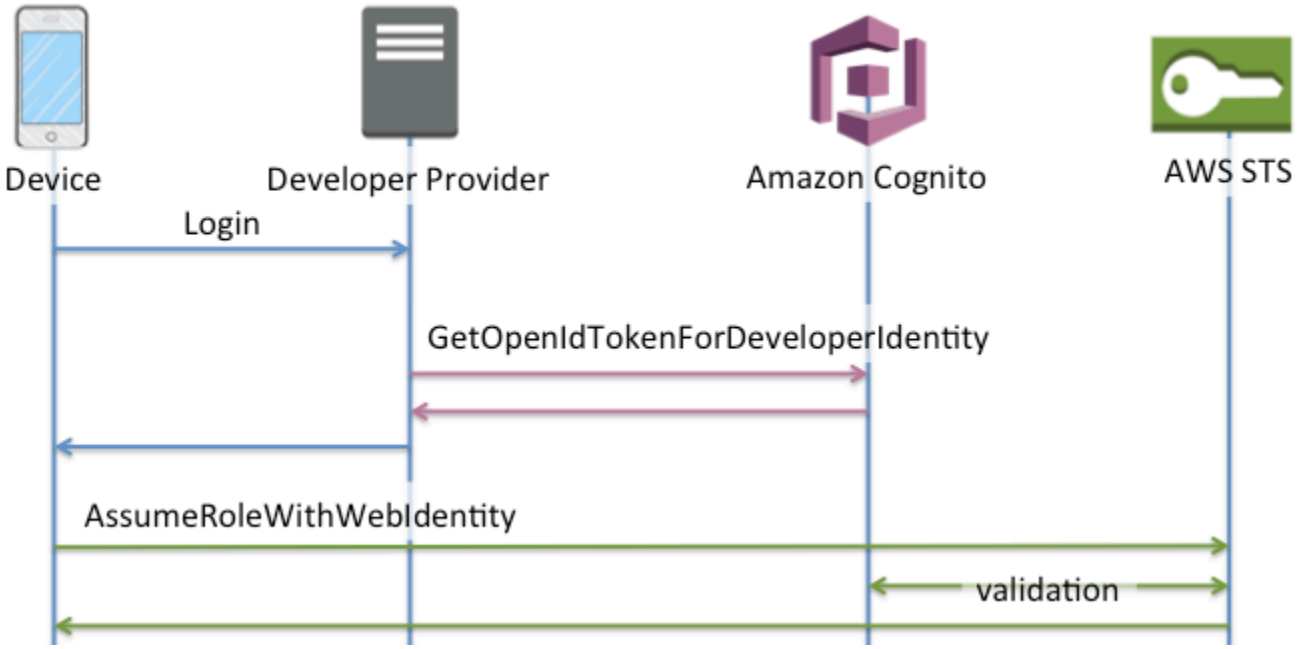
デベロッパープロバイダーによる拡張認証でのオペレーションの順序

1. デベロッパープロバイダー経由でのログイン (Amazon Cognito 外部のコード)
2. ユーザーログインの検証 (Amazon Cognito 外部のコード)
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



デベロッパープロバイダーによる基本認証でのオペレーションの順序

1. ID プールの外部にロジックを実装してサインインし、デベロッパーとプロバイダーの識別子を生成します。
2. 保存されたサーバー側の AWS 認証情報を取得します。
3. 承認された AWS 認証情報で署名された GetOpenIdTokenForDeveloperIdentity API リクエストでデベロッパープロバイダー識別子を送信します。
4. でアプリケーション認証情報をリクエストしますAssumeRoleWithWebIdentity。



使用するべき認証フロー

拡張フローは、デベロッパーの労力が最も低い、最も安全な選択肢です。

- 拡張フローにより、API リクエストの複雑さ、サイズ、レートが軽減されます。
- アプリケーションは、 に追加の API リクエストを行う必要はありません AWS STS。
- ID プールは、ユーザーに対して、受信する必要がある IAM ロール認証情報を評価します。クライアントでロールを選択するためにロジックを埋め込む必要はありません。

> **Important**
>
> 新しい ID プールを作成するときは、ベストプラクティスとして、基本 (クラシック) 認証をデフォルトでアクティブ化しないでください。基本認証を実装するには、まず IAM ロールのウェブ ID の信頼関係を評価します。次に、クライアントにロール選択のロジックを構築し、ユーザーによる変更からクライアントを保護します。

基本的な認証フローは、IAM ロール選択のロジックをアプリケーションに委任します。このフローでは、Amazon Cognito はユーザーの認証されたセッションまたは認証されていないセッションを検証し、 と認証情報と交換できるトークンを発行します AWS STS。ユーザーは、基本認証からトークンを、ID プールと を信頼する任意の IAM ロールamr、または認証済み/未認証の状態と交換できます。

同様に、デベロッパー認証は ID プロバイダー認証の検証に関するショートカットであることを理解してください。Amazon Cognito は、GetOpenIdTokenForDeveloperIdentityリクエストの内容をさらに検証することなく、リクエストを承認する AWS 認証情報を信頼します。ユーザーによるアクセスからデベロッパー認証を許可するシークレットを保護します。

API の要約

GetId

GetId API コールは、Amazon Cognito で新しい ID を確立するために必要な最初のコールです。

非認証アクセス

Amazon Cognito には、アプリケーションで認証されていないゲストアクセスを許可できます。この機能が ID プールで有効になっている場合、ユーザーはいつでも `GetId` API 経由で新しいアイデンティティ ID をリクエストできます。Amazon Cognito に後続のコールを実行するため、アプリケーションにはこのアイデンティティ ID をキャッシュすることが期待されます。ブラウザの Mobile SDKs AWS と AWS SDK for には JavaScript 、このキャッシュを処理する認証情報プロバイダーがあります。

認証されたアクセス

パブリックログインプロバイダー (Facebook、Google+、Login with Amazon、または「Apple でサインイン」など) のサポートでアプリケーションを設定すると、ユーザーは、これらのプロバイダーでユーザーを識別するトークン (OAuth または OpenID Connect) を提供できるようになります。`GetId` の呼び出しで使用されると、Amazon Cognito は新しい認証されたアイデンティティを作成するか、その特定のログインに既に関連付けられているアイデンティティを返します。Amazon Cognito は、プロバイダーでトークンを検証し、以下を確実にすることでこれを行います。

- トークンは有効で、設定されたプロバイダーからのものである。
- トークンの有効期限が切れていない。
- トークンがそのプロバイダーで作成されたアプリケーション識別子 (例えば、Facebook アプリ ID) と一致する。
- トークンがユーザー識別子と一致する。

GetCredentialsForIdentity

GetCredentialsForIdentity API は、ID を確立した後に呼び出すことができます。このオペレーションは、機能的には を呼び出しGetOpenIdToken、次に を呼び出すのと同等ですAssumeRoleWithWebIdentity。

代わりに Amazon Cognito で `AssumeRoleWithWebIdentity` を呼び出す場合は、Amazon Cognito に関連付けられた IAM ロールが ID プールにある必要があります。これは、Amazon Cognito コンソールまたは SetIdentityPoolRolesオペレーションを使用して手動で実行できます。

GetOpenIdToken

ID を確立した後に GetOpenIdToken API リクエストを行います。最初のリクエスト後にアイデンティティ ID をキャッシュし、`GetOpenIdToken` を使用してその ID 以降の基本 (クラシック) セッションを開始します。

`GetOpenIdToken` API リクエストに対するレスポンスは、Amazon Cognito が生成するトークンです。このトークンは、 AssumeRoleWithWebIdentityリクエストの `WebIdentityToken`パラメータとして送信できます。

OpenID トークンを送信する前に、アプリで検証してください。SDK の OIDC ライブラリや aws-jwt-verify のようなライブラリを使用して、Amazon Cognito がトークンを発行したことを確認できます。OpenID トークンの署名キー ID または `kid` は、Amazon Cognito ID jwks_uriドキュメント† にリストされているもののいずれかです。これらのキーは変更される可能性があります。Amazon Cognito ID トークンを検証する関数は、jwks_uri ドキュメントからキーのリストを定期的に更新する必要があります。Amazon Cognito は、jwks_uri キャッシュコントロールレスポンスヘッダーで更新期間を設定しており、現在 `max-age` の 30 日間に設定されています。

認証されていないアクセス

認証されていない ID のトークンを取得するには、アイデンティティ ID そのもののみが必要です。認証された ID または無効にした ID の認証されていないトークンを取得することはできません。

認証されたアクセス

認証済みの ID がある場合、その ID に既に関連付けられたログイン用に、少なくとも 1 つのトークンを渡す必要があります。`GetOpenIdToken` の呼び出し中に渡されるすべてのトークンは、前に説明したのと同じ検証を渡す必要があります。いずれかのトークンが失敗すると、呼び出し全体が失敗します。`GetOpenIdToken` 呼び出しからの応答に、アイデンティティ ID が含まれることもあります。これは、渡すアイデンティティ ID が、返される ID とは限らないためです。

ログインのリンク

既に任意の ID と関連付けられていないログインのトークンを送信すると、そのログインは関連付けられた ID に「リンクしている」と見なされます。パブリックプロバイダーごとに、1

つのログインのみをリンクできます。複数のログインをパブリックプロバイダーにリンクしようとすると、応答として ResourceConflictException エラーが発生します。ログインが単純に既存の アイデンティティ にリンクされている場合、GetOpenIdToken から返されるアイデンティティ ID は、渡された ID と同じになります。

ID の結合

現在、特定の ID にリンクされていないが、別の ID にリンクされているログインに対してトークンを渡すと、2 つの ID が結合されます。いったん結合されると、1 つの ID は、すべての関連ログインの親/所有者になり、もう 1 つは無視されます。この場合、親/所有者のアイデンティティ ID が返されます。この値が異なる場合は、ローカルキャッシュを更新する必要があります。ブラウザの AWS Mobile SDKs または AWS SDK for JavaScript のプロバイダーが、このオペレーションを実行します。

GetOpenIdTokenForDeveloperIdentity

GetOpenIdTokenForDeveloperIdentity オペレーションは、デベロッパーGetOpenIdTokenが認証した ID を使用する場合の、デバイスから GetIdおよび の使用を置き換えます。アプリケーションは AWS 認証情報を使用してこの API オペレーションへのリクエストに署名するため、Amazon Cognito はリクエストで指定されたユーザー識別子が有効であると信頼します。デベロッパー認証は、Amazon Cognito が実行するトークン検証を外部プロバイダーに置き換えます。

この API のペイロードにはloginsマップが含まれています。このマップには、デベロッパープロバイダーのキーと、システム内のユーザーの識別子としての 値が含まれている必要があります。ユーザー識別子がまだ既存のアイデンティティに既にリンクされていない場合は、Amazon Cognito が新しいアイデンティティを作成して、新しいアイデンティティ ID と、そのアイデンティティの OpenID Connect を返します。ユーザー識別子が既にリンクされている場合は、Amazon Cognito が既存のアイデンティティ ID と OpenID Connect トークンを返します。最初のリクエスト後にデベロッパーアイデンティティ ID をキャッシュし、GetOpenIdTokenForDeveloperIdentity を使用してその ID 以降の基本 (クラシック) セッションを開始します。

GetOpenIdTokenForDeveloperIdentity API リクエストに対するレスポンスは、Amazon Cognito が生成するトークンです。このトークンは、AssumeRoleWithWebIdentity リクエストの WebIdentityToken パラメータとして送信できます。

OpenID Connect トークンを送信する前に、アプリで検証してください。SDK の OIDC ライブラリや aws-jwt-verify のようなライブラリを使用して、Amazon Cognito がトークンを発行したことを確認できます。OpenID Connect トークンの署名キー ID または kid は、Amazon Cognito

ID jwks_uri ドキュメント† にリストされているもののいずれかです。これらのキーは変更される可能性があります。Amazon Cognito ID トークンを検証する関数は、jwks_uri ドキュメントからキーのリストを定期的に更新する必要があります。Amazon Cognito は、jwks_uri `cache-control` レスポンスヘッダーで更新期間を設定しており、現在 `max-age` の 30 日間に設定されています。

ログインのリンク

外部プロバイダーと同様に、既に ID に関連付けられていない追加のログインを指定すると、それらのログインがその ID に暗黙的にリンクされます。外部プロバイダーのログインを ID にリンクする場合、ユーザーはそのプロバイダーで外部プロバイダーの認証フローを使用できます。ただし、`GetId` または `GetOpenIdToken` を呼び出すときに、ログインマップで開発者プロバイダ名を使用することはできません。

ID の結合

デベロッパーが認証した ID では、Amazon Cognito は MergeDeveloperIdentities API を介した暗黙的なマージと明示的なマージの両方をサポートします。明示的なマージにより、システムでユーザー識別子を持つ 2 つのアイデンティティを、1 つのアイデンティティとしてマークできます。ソースと宛先のユーザー識別子を指定すると、Amazon Cognito がそれらをマージします。次回にいずれかのユーザー識別子に対して OpenID Connect トークンをリクエストすると、同じアイデンティティ ID が返されます。

AssumeRoleWithWebIdentity

OpenID Connect トークンを取得したら、これを () への AssumeRoleWithWebIdentity AWS Security Token Service API リクエストを通じて一時的な AWS 認証情報と交換できますAWS STS。

作成できる ID の数には制限がないため、ユーザーに付与するアクセス権限を理解することが重要です。アプリケーションに異なる IAM ロールを設定します。1 つは認証されていないユーザー用、もう 1 つは認証されたユーザー用です。Amazon Cognito コンソールでは、アイデンティティプールを初めてセットアップするときにデフォルトのロールを作成できます。これらのロールには、実質的にアクセス許可が付与されていません。ニーズに合わせて変更します。

ロールの信頼とアクセス権限 の詳細を確認してください。

† デフォルトの Amazon Cognito ID の jwks_uri ドキュメントには、ほとんどの AWS リージョンで ID プールのトークンに署名するキーに関する情報が含まれています。以下のリージョンには異なる jwks_uri ドキュメントがあります。

Amazon Cognito Identity JSON web key URIs in other AWS リージョン

AWS リージョン	jwks_uri ドキュメントへのパス
AWS GovCloud （米国西部）	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
中国 (北京)	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
欧州 (ミラノ) やアフリカ (ケープタウン) などのオプトインリージョン	`https://cognito-identity.`*`Region`*`.amazonaws.com/.well-known/jwks_uri`

また、発行者から jwks_uri を推定することも、Amazon Cognito から OpenID トークンで受け取った `iss` を推定することもできます。OIDC 標準の検出エンドポイント `<issuer>/.well-known/openid-configuration` には、トークンの jwks_uri へのパスがリストされます。

IAM ロール

ID プールの作成中に、ユーザーが引き受ける IAM ロールを更新するよう求められます。IAM ロールは次のように動作します。ユーザーがアプリにログインすると、Amazon Cognito はユーザーの一時的な AWS 認証情報を生成します。これらの一時的な認証情報は、特定の IAM ロールに関連付けられます。IAM ロールを使用すると、 AWS リソースにアクセスするための一連のアクセス許可を定義できます。

認証されたユーザーと認証されていないユーザー用に、デフォルトの IAM ロールを指定できます。また、ユーザーの ID トークンのクレームに基づいて、各ユーザーのロールを選択するルールを定義できます。詳細については、「ロールベースアクセスコントロールの使用」を参照してください。

Amazon Cognito コンソールは、Amazon Mobile Analytics および Amazon Cognito Sync へのアクセス権を提供する IAM ロールをデフォルトで作成します。または、既存の IAM ロールの使用を選択できます。

IAM ロールを変更して、他のサービスへのアクセスを許可または制限します。これを行うためには、IAM コンソールにログインします。次に、[Roles] (ロール) を選択し、ロールを選択します。選

択されたロールにアタッチされたポリシーが、[Permissions] (アクセス許可) タブに表示されます。対応する [Manage Policy] (ポリシーの管理) リンクを選択してアクセスポリシーをカスタマイズできます。ポリシーの使用と定義の詳細については、「IAM ポリシーの概要」を参照してください。

> **Note**
>
> ベストプラクティスとして、最小権限の付与の原則に準拠したポリシーを定義します。つまり、ポリシーには、ユーザーがタスクを実行するために必要なアクセス権限のみを含めます。詳細については、「IAM ユーザーガイド」の「Grant Least Privilege」(最小権限を付与する) を参照してください。
> 認証されていない ID は、アプリにログインしていないユーザーが引き受けることに注意してください。通常、認証されていない ID に割り当てるアクセス権限は、認証された ID に割り当てるアクセス権限よりも制限が厳しいものである必要があります。

トピック

- 信頼ポリシーのセットアップ
- アクセスポリシー

信頼ポリシーのセットアップ

Amazon Cognito は IAM ロールを使用して、アプリケーションのユーザー用の一時的な認証情報を生成します。アクセス権限へのアクセスは、ロールの信頼関係によって制御されます。詳細情報 ロールの信頼とアクセス権限。

に提示されるトークン AWS STS は、ユーザープール、ソーシャルプロバイダートークン、OIDC プロバイダートークン、または SAML アサーションを独自のトークンに変換する ID プールによって生成されます。アイデンティティプールトークンには、アイデンティティプール ID として aud クレームが含まれています。

次のロール信頼ポリシーの例では、フェデレーティッドサービスプリンシパルが AWS STS API を呼び出すcognito-identity.amazonaws.comことを許可しますAssumeRoleWithWebIdentity。リクエストが成功するのは、API リクエストのアイデンティティプールトークンに以下のクレームが含まれている場合のみです。

1. アイデンティティプール ID aud の us-west-2:abcdefg-1234-5678-910a-0e8443553f95 クレーム。

2. ユーザーがログインしていて、ゲストユーザーではない場合に追加される authenticated の amr クレーム。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "authenticated"
                }
            }
        }
    ]
}
```

基本 (クラシック) 認証における IAM ロールの信頼ポリシー

ID プールで使用するロールの信頼ポリシーを制限する条件を少なくとも 1 つ適用する必要があります。ID プールのロール信頼ポリシーを作成または更新するときに、ソース ID を制限する条件キーを少なくとも 1 つ使用せずに変更を保存しようとすると、IAM AWS STS はエラーを返します。ID プールからこのタイプの条件を持たない IAM ロールへのクロスアカウントAssumeRoleWithWebIdentityオペレーションは許可されません。

このトピックには、ID プールのソース ID を制限するいくつかの条件が含まれています。詳細なリストについては、AWS 「ウェブ ID フェデレーションで使用できるキー」を参照してください。

ID プールによる基本またはクラシックの認証では、適切な信頼ポリシー AWS STS がある場合、 で任意の IAM ロールを引き受けることができます。Amazon Cognito アイデンティティプールの IAM ロールは、サービスプリンシパル cognito-identity.amazonaws.com を信頼してロールを引き受けます。この設定では、リソースへの意図しないアクセスから IAM ロールを保護するには不十分

です。このタイプのロールは、ロール信頼ポリシーに追加の条件を適用する必要があります。次の条件の少なくとも 1 つがないと、ID プールのロールを作成または変更することはできません。

cognito-identity.amazonaws.com:aud

ロールを 1 つ以上の ID プールからのオペレーションに制限します。Amazon Cognito は、ID プールトークンの audクレーム内のソース ID プールを示します。

cognito-identity.amazonaws.com:amr

ロールを authenticatedまたは unauthenticated (ゲスト) ユーザーに制限します。Amazon Cognito は、ID プールトークンの amrクレームの認証状態を示します。

cognito-identity.amazonaws.com:sub

ロールを UUID で 1 人以上のユーザーに制限します。この UUID は、ID プール内のユーザーの ID です。この値は、ユーザーの元の ID プロバイダーsubの値ではありません。Amazon Cognito は、アイデンティティプールトークンの subクレームでこの UUID を示します。

拡張フロー認証では、IAM ロールが ID プール AWS アカウント と同じ にある必要がありますが、基本認証ではそうではありません。

クロスアカウントの IAM ロールを引き受ける Amazon Cognito アイデンティティプール には、その他の考慮事項が適用されます。これらのロールの信頼ポリシーは、cognito-identity.amazonaws.comサービスプリンシパルを受け入れ、特定のcognito-identity.amazonaws.com:aud条件を含める必要があります。 AWS リソースへの意図しないアクセスを防ぐため、 aud条件キーは、 条件値の ID プールからユーザーにロールを制限します。

ID プールが ID に対して発行するトークンには、ID プール AWS アカウント の送信元に関する情報が含まれます。AssumeRoleWithWebIdentity API リクエストで ID プールトークンを提示すると、は発信元の ID プールが IAM ロール AWS アカウント と同じ にある AWS STS かどうかを確認します。は、リクエストがクロスアカウントである AWS STS と判断すると、ロール信頼ポリシーに aud条件があるかどうかを確認します。ロール信頼ポリシーにそのような条件が存在しない場合、継承ロール呼び出しは失敗します。リクエストがクロスアカウントでない場合は、この制限を適用 AWS STS しないでください。ベストプラクティスとして、常にこのタイプの条件を ID プールロールの信頼ポリシーに適用します。

追加の信頼ポリシー条件

ID プール間でのロールの再利用

複数の ID プール間でロールを再利用するには、ロールが共通のアクセス権限のセットを共有するため、次のように複数の ID プールを含めることができます。

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": [
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"
    ]
}
```

特定の ID へのアクセスの制限

特定のアプリユーザーに限定されたポリシーを作成するには、cognito-identity.amazonaws.com:sub の値を確認します。

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
    "cognito-identity.amazonaws.com:sub": [
        "us-east-1:12345678-1234-1234-1234-123456790ab",
        "us-east-1:98765432-1234-1234-1243-123456790ab"
    ]
}
```

特定のプロバイダーへのアクセスの制限

特定のプロバイダー (おそらくはユーザー独自のログインプロバイダー) でログインしたユーザーに制限されたポリシーを作成するには、cognito-identity.amazonaws.com:amr の値を確認します。

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

たとえば、Facebook のみを信頼するアプリに、次の amr 句があるとします。

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

アクセスポリシー

ロールにアタッチするアクセス許可は、そのロールを引き受けるすべてのユーザーに適用されます。ユーザーのアクセスをパーティション化するには、ポリシー条件と変数を使用します。詳細については、「IAM ポリシーの要素: 変数とタグ」を参照してください。sub 条件を使用して、アクションをアクセスポリシーの Amazon Cognito アイデンティティ ID に制限できます。このオプションは注意して使用してください。特に、一貫したユーザー ID を持たない認証されていない ID には注意が必要です。Amazon Cognito とのウェブフェデレーションの IAM ポリシー変数の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS 条件コンテキストキー」を参照してください。

セキュリティ保護を向上させるために、Amazon Cognito は `GetCredentialsForIdentity` を使用して、拡張フローで認証されていないユーザーを割り当てる認証情報にスコープダウンポリシーを適用します。スコープダウンポリシーは、認証されていないロールに適用する IAM ポリシーに インラインセッションポリシー と AWS マネージドセッションポリシー を追加します。ロールの IAM ポリシーとセッションポリシーの両方でアクセスを許可する必要があるため、スコープダウンポリシーでは、次のリストに示すサービス以外のサービスへのユーザーアクセスが制限されます。

> **Note**
>
> 基本 (クラシック) フローでは、独自の AssumeRoleWithWebIdentity API リクエストを作成し、そのリクエストにこれらの制限を適用できます。セキュリティのベストプラクティスとして、このスコープダウンポリシーを超えるアクセス許可を認証されていないユーザーに割り当てないでください。

また、Amazon Cognito は、認証されたユーザーと認証されていないユーザーが Amazon Cognito ID プールと Amazon Cognito Sync に API リクエストを実行できないようにします。ウェブ ID からのサービスアクセスに制限が課せられる AWS のサービス 場合もあります。

拡張フローでリクエストが成功すると、Amazon Cognito はバックグラウンドで `AssumeRoleWithWebIdentity` API リクエストを行います。このリクエストのパラメータとしては、Amazon Cognito に次のものが含まれます。

1. ユーザーのアイデンティティ ID。
2. ユーザーが引き受ける IAM ロールの ARN。
3. インラインセッションポリシーを追加する `policy` パラメータ。

4. 値が Amazon で追加のアクセス許可を付与する AWS マネージドポリシーであるPolicyArns.member.Nパラメータ CloudWatch。

認証されていないユーザーがアクセスできるサービス

拡張フローを使用すると、Amazon Cognito がユーザーセッションに適用するスコープダウンポリシーにより、次の表に記載されている以外のサービスを使用できなくなります。一部のサービスでは、特定のアクションのみが許可されます。

カテゴリ	サービス
分析	Amazon Data Firehose Amazon Managed Service for Apache Flink
アプリケーション統合	Amazon Simple Queue Service
AR とバーチャルリアリティ	Amazon Sumerian[1]
ビジネスアプリケーション	Amazon Mobile Analytics Amazon Simple Email Service
コンピューティング	AWS Lambda
暗号化と PKI	AWS Key Management Service 1
データベース	Amazon DynamoDB Amazon SimpleDB
フロントエンドウェブおよびモバイル	AWS AppSync Amazon Location Service Amazon Simple Notification Service Amazon Pinpoint
ゲーム開発	Amazon GameLift

カテゴリ	サービス
IoT	AWS IoT
機械学習	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition Amazon SageMaker1 Amazon Textract[1] Amazon Transcribe Amazon Translate
管理とガバナンス	Amazon CloudWatch Amazon CloudWatch Logs
ネットワーキングとコンテンツ配信	Amazon API Gateway
セキュリティ、アイデンティティ、コンプライアンス	Amazon Cognito ユーザープール
[Storage (ストレージ)]	Amazon Simple Storage Service

1 次の表の AWS のサービス の場合、インラインポリシーはアクションのサブセットを付与します。表には、それぞれで実行可能なアクションが表示されています。

AWS のサービス	認証されていない拡張フローユーザーの最大アクセス許可

AWS Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncrypt`
	`GenerateDataKey`

Amazon SageMaker	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`

このリスト AWS のサービス 以外の へのアクセスを許可するには、ID プールで基本 (クラシック) 認証フローを有効にします。認証されていないユーザーの IAM ロールに割り当てられたポリシーで許可されている AWS のサービス からの `NotAuthorizedException` エラーがユーザーに表示された場合は、そのサービスをユースケースから削除できるかどうかを評価してください。できない場合は、基本フローに切り替えてください。

インラインセッションポリシー

インラインセッションポリシーは、ユーザーの有効なアクセス許可に、次のリストの AWS のサービス 外部へのアクセスを含めることを制限します。また、ユーザーの IAM ロールに適用するポリシー AWS のサービス で、これらにアクセス許可を付与する必要があります。ロールを引き受けたセッションにおけるユーザーの有効なアクセス許可は、そのユーザーのロールに割り当てられたポリシーとセッションポリシーが交差したものです。詳細については、AWS Identity and Access Management ユーザーガイドの「セッションポリシー」を参照してください。

Amazon Cognito は、デフォルトで有効になっている AWS リージョン ユーザーのセッションに次のインラインポリシーを追加します。

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

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "logs:*",
                "dynamodb:*",
                "kinesis:*",
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "scs:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

他のすべてのリージョンの場合、インラインスコープダウンポリシーには、次の Action ステートメントを除いて、デフォルトのリージョンにリストされているものがすべて含まれます。

```
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

AWS マネージドセッションポリシー

また、Amazon Cognito は、 AWS マネージドポリシー AmazonCognitoUnAuthedIdentitiesSessionPolicy による未認証ユーザーのアクセス許可の範囲を、拡張フローの未認証ユーザーに制限します。また、認証されていない IAM ロールに適用するポリシーでこのアクセス許可を付与する必要があります。

AmazonCognitoUnAuthedIdentitiesSessionPolicy 管理ポリシーには以下のアクセス許可が記載されています。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint"
        ],
        "Resource": "*"
    }]
```

```
}
```

アクセスポリシーの例

このセクションでは、特定のオペレーションの実行に必要な最小限のアクセス許可を付与する Amazon Cognito アクセスポリシーの例を示します。可能な場合は、ポリシー変数を使用して特定のアイデンティティ ID のアクセス権限をさらに制限できます。たとえば、${cognito-identity.amazonaws.com:sub} を使用します。詳細については、AWS モバイルブログで「Understanding Amazon Cognito Authentication Part 3: Roles and Policies」を参照してください。

> **Note**
>
> セキュリティのベストプラクティスとして、ポリシーには、ユーザーがタスクを実行するために必要なアクセス権限のみを含めてください。つまり、可能であれば必ずオブジェクトの個々の ID ごとにアクセスを設定するようにしてください。

ID に Amazon S3 内の単一のオブジェクトへの読み取りアクセスを許可する

次のアクセスポリシーは、特定の S3 バケットから 1 つのオブジェクトを取得するための読み取りアクセス権限を ID に付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

ID に Amazon S3 の ID 固有のパスへの読み取りおよび書き込みアクセスを許可する

次のアクセスポリシーは、プレフィックスを `${cognito-identity.amazonaws.com:sub}` 変数にマッピングすることで、S3 バケット内の特定のプレフィックス「folder」にアクセスする読み取りおよび書き込みアクセス権限を付与します。

このポリシーでは、${cognito-identity.amazonaws.com:sub} を使用して挿入された us-east-1:12345678-1234-1234-1234-123456790ab などの ID は、オブジェクトを取得し、arn:aws:s3:::mybucket/us-east-1:12345678-1234-1234-1234-123456790ab に配置およびリストできます。ただし、この ID には arn:aws:s3:::mybucket の他のオブジェクトへのアクセスは許可されません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-
identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

ID に Amazon DynamoDB への詳細に設定されたアクセスを割り当てる

次のアクセスポリシーは、Amazon Cognito 環境変数を使用して、Amazon DynamoDB リソースへの詳細に設定されたアクセスコントロールを行います。これらの変数は、アイデンティティ ID によって DynamoDB のアイテムへのアクセス権限を付与します。詳細については、Amazon DynamoDB デベロッパーガイドの「詳細に設定されたアクセスコントロールのための IAM ポリシー条件の使用」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
```

```
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys":  ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

Lambda 関数を呼び出すためのアクセス権限を ID に与えます。

次のアクセスポリシーは、Lambda 関数を呼び出すアクセス権限を ID に付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
             "Effect": "Allow",
             "Action": "lambda:InvokeFunction",
             "Resource": [
                 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
             ]
         }
     ]
 }
```

Amazon Kinesis Data Stream へのレコードを発行するアクセス権限を ID に付与する

次のアクセスポリシーでは、ID が任意の Kinesis Data Streams で PutRecord オペレーションを使用することを許可します。これは、アカウント内のすべてのストリームにデータレコードを追加する必要があるユーザーに適用できます。詳細については、Amazon Kinesis Data Streams デベロッパーガイドの「IAM を使用して Amazon Kinesis Streams リソースへのアクセスを制御する」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

ID に Amazon Cognito Sync ストアのデータへのアクセス権を付与する

以下のアクセスポリシーは、アイデンティティに Amazon Cognito Sync ストア内のデータのみに対するアクセス権限を付与します。

```
{
  "Version": "2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":"cognito-sync:*",
      "Resource":["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/${cognito-identity.amazonaws.com:aud}/identity/${cognito-identity.amazonaws.com:sub}/*"]
      }]
  }
```

ロールの信頼とアクセス権限

これらのロールが異なる点は、それらの信頼関係です。未認証ロールの信頼ポリシーの例を次に示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

このポリシーは、cognito-identity.amazonaws.com (OpenID Connect トークンの発行者) のフェデレーティッドユーザーに、このロールを引き受ける許可を与えます。さらに、ポリシーでは、ID プールに一致させるためにトークンの aud が制限されます (この場合は ID プール ID)。最後に、このポリシーは、Amazon Cognito GetOpenIdToken API オペレーションによって発行されたトークンの複数の値からなる amr クレームの配列メンバーの 1 つが unauthenticated 値を含みます

Amazon Cognito がトークンを作成すると、トークンの amr を unauthenticated または authenticated のいずれかに設定します。amr が authenticated の場合、トークンには、認証中に使用されるすべてのプロバイダーが含まれます。これにより、amr 条件を次のように変更するだけで、Facebook 経由でログインしたユーザーのみを信頼するロールを作成できます。

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

ロールで信頼関係を変更するときや、ID プール間でロールの使用を試みるときは注意が必要です。ID プールを正しく信頼するようにロールが設定されていない場合は、STS の結果から次のような例外が表示されます。

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

このメッセージが表示された場合は、ID プールと認証タイプに適切なロールがあることを確認します。

Amazon Cognito ID プールのセキュリティのベストプラクティス

Amazon Cognito ID プールは、アプリケーションの一時的な AWS 認証情報を提供します。 には、アプリケーションユーザーが必要とするリソースとプライベートバックエンドリソースの両方が含まれる AWS アカウント ことがよくあります。 AWS 認証情報を構成する IAM ロールとポリシーは、これらのリソースへのアクセスを許可できます。

ID プール設定の主なベストプラクティスは、アプリケーションが過剰な権限や意図しない権限なしでジョブを完了できるようにすることです。セキュリティの設定ミスを防ぐには、本番環境にリリースする各アプリケーションを起動する前に、これらの推奨事項を確認してください。

トピック

IAM 設定のベストプラクティス

ゲストまたは認証されたユーザーが、アイデンティティプールの認証情報を必要とするセッションをアプリケーションで開始すると、アプリケーションは IAM ロールの一時的な AWS 認証情報を取得します。認証情報は、デフォルトのロール、ID プール設定のルールによって選択されたロール、またはアプリによって選択されたカスタムロールです。各ロールに割り当てられているアクセス許可により、ユーザーは リソースにアクセスできます AWS 。

一般的な IAM のベストプラクティスの詳細については、「 ユーザーガイド」の「IAM のベストプラクティス AWS Identity and Access Management 」を参照してください。

IAM ロールで信頼ポリシー条件を使用する

IAM では、ID プールのロールに少なくとも 1 つの信頼ポリシー条件が必要です。この条件は、例えば、ロールの範囲を認証されたユーザーのみに設定することができます。 AWS STS では、クロスアカウントの基本認証リクエストに `cognito-identity.amazonaws.com:aud`と の 2 つの特定の条件が必要です`cognito-identity.amazonaws.com:amr`。ベストプラクティスとして、ID プールサービスプリンシパル を信頼するすべての IAM ロールにこれらの条件の両方を適用します`cognito-identity.amazonaws.com`。

- `cognito-identity.amazonaws.com:aud`: ID プールトークンの aud クレームは、信頼できる ID プール ID と一致する必要があります。

- cognito-identity.amazonaws.com:amr: ID プールトークンの amr クレームは、認証済みまたは未認証の である必要があります。この条件では、認証されていないゲストのみ、または認証されたユーザーのみにロールへのアクセスを予約できます。この条件の値をさらに絞り込んで、ロールを特定のプロバイダーのユーザーに制限できますgraph.facebook.com。例えば、。

次のロール信頼ポリシーの例では、以下の条件でロールへのアクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

ID プールに関連する要素

- "Federated": "cognito-identity.amazonaws.com": ユーザーは ID プールから取得する必要があります。
- "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111": ユーザーは特定の ID プール から取得する必要がありますus-east-1:a1b2c3d4-5678-90ab-cdef-example11111。
- "cognito-identity.amazonaws.com:amr": "authenticated": ユーザーは認証されている必要があります。ゲストユーザーはロールを引き受けることができません。

最小特権のアクセス許可を適用する

認証されたアクセスまたはゲストアクセスの IAM ポリシーでアクセス許可を設定する場合は、特定のタスクの実行に必要な特定のアクセス許可、または最小権限のアクセス許可のみを付与します。次の IAM ポリシーの例では、ロールに適用すると、Amazon S3 バケット内の 1 つのイメージファイルへの読み取り専用アクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

ID プール設定のベストプラクティス

ID プールには、 AWS 認証情報を生成するための柔軟なオプションがあります。アプリケーションが最も安全な方法で動作できる場合は、設計ショートカットを使用しないでください。

ゲストアクセスの影響を理解する

認証されていないゲストアクセスにより、ユーザーはサインイン AWS アカウント する前に からデータを取得できます。ID プール ID を知っているユーザーは誰でも、認証されていない認証情報をリクエストできます。ID プール ID は機密情報ではありません。ゲストアクセスをアクティブ化すると、認証されていないセッションに付与した AWS アクセス許可はすべてのユーザーが利用できます。

ベストプラクティスとして、ゲストアクセスを非アクティブ化したままにして、ユーザーが認証された後にのみ必要なリソースを取得します。アプリケーションがサインイン前に リソースにアクセスする必要がある場合は、次の予防措置を講じてください。

- 認証されていないロールに適用される自動制限を理解します。
- アプリケーションの特定のニーズに合わせて、認証されていない IAM ロールのアクセス許可をモニタリングおよび調整します。

- 特定のリソースへのアクセスを許可します。
- デフォルトの認証されていない IAM ロールの信頼ポリシーを保護します。
- インターネット上のすべてのユーザーに IAM ロールのアクセス許可を付与すると確信できる場合にのみ、ゲストアクセスを有効にします。

拡張認証をデフォルトで使用する

基本 (クラシック) 認証では、Amazon Cognito は IAM ロールの選択をアプリケーションに委任します。対照的に、拡張フローは ID プール内の集中ロジックを使用して IAM ロールを決定します。また、IAM アクセス許可の上限を設定するスコープダウンポリシーにより、認証されていない ID のセキュリティも強化されます。拡張フローは、デベロッパーの労力が最も低い、最も安全な選択肢です。これらのオプションの詳細については、ID プール (フェデレーティッドアイデンティティ) の認証フロー「」を参照してください。

基本的なフローでは、認証情報の AWS STS API リクエストのロール選択とアセンブリに入るクライアント側のロジックを公開できます。拡張フローでは、ID プールの自動化の背後にあるロジックと assume-role リクエストの両方が非表示になります。

基本認証を設定するときは、IAM ロールとそのアクセス許可に IAM ベストプラクティスを適用します。

デベロッパープロバイダーを安全に使用する

デベロッパーが認証した ID は、サーバー側のアプリケーションの ID プールの機能です。ID プールがデベロッパー認証に必要とする認証の唯一の証拠は、ID プールデベロッパーの AWS 認証情報です。ID プールは、この認証フローで提示したデベロッパーとプロバイダーの識別子の有効性に制限を課しません。

ベストプラクティスとして、デベロッパープロバイダーは、以下の条件でのみ実装してください。

- デベロッパーが認証した認証情報の使用に関する説明責任を作成するには、デベロッパープロバイダーの名前と識別子を設計して認証ソースを示します。例: `"Logins" : {"MyCorp provider" : "[provider application ID]"}`。
- 存続期間の長いユーザー認証情報は避けてください。EC2 インスタンスプロファイルや Lambda 実行ロール などのサービスにリンクされたロールを使用して ID をリクエストするようにサーバー側のクライアントを設定します。
- 内部と外部の信頼ソースを同じ ID プールに混在させないようにしてください。デベロッパープロバイダーとシングルサインオン (SSO) プロバイダーを別々の ID プールに追加します。

アクセスコントロールへの属性の使用

アクセスコントロールの属性は、アクセスコントロール (ABAC) の Amazon Cognito アイデンティティプール実装です。ユーザー属性に基づく Amazon Cognito の ID プール経由で AWS リソースへのアクセスを制御するには、IAM ポリシーを使用することができます。これらの属性は、ソーシャル ID プロバイダーおよび企業 ID プロバイダーから収集できます。プロバイダーのアクセストークンと ID トークン、または SAML アサーション内の属性を、IAM 許可ポリシーで参照できるタグにマップできます。

Amazon Cognito ID プールでは、デフォルトのマッピングを選択する、または独自のカスタムマッピングを作成することが可能です。デフォルトのマッピングでは、ユーザー属性の固定セットに基づいた IAM ポリシーを記述できます。カスタムマッピングでは、IAM 許可ポリシーで参照されるユーザー属性のカスタムセットを選択できます。Amazon Cognito コンソールの [Attribute names] (属性名) は、IAM 許可ポリシーで参照される [Tag key for principal] (プリンシパルのタグキー) にマップされます。

例えば、無料と有料のメンバーシップがあるメディアストリーミングサービスを所有しているとしましょう。メディアファイルを Amazon S3 に保存し、無料タグまたはプレミアムタグでそれらをタグ付けします。アクセスコントロールの属性を使用して、ユーザーのプロファイルの一部であるユーザーメンバーシップレベルに基づいて、無料コンテンツと有料コンテンツへのアクセスを許可できます。メンバーシップ属性は、IAM 許可ポリシーに渡されるプリンシパルのタグキーにマップできます。そうすることで、単一の許可ポリシーを作成し、メンバーシップレベルの値とコンテンツファイルのタグに基づいて、プレミアムコンテンツへのアクセスを条件的に許可することができます。

トピック

属性を使用したアクセスの制御には、いくつかのメリットがあります。

- アクセスコントロールに属性を使用すると、許可の管理がより効率的になります。異なるジョブ機能のために複数のポリシーを作成するのではなく、ユーザー属性を使用する基本的な許可ポリシーを作成できます。
- アプリケーションのリソースやユーザーを追加または削除するたびにポリシーを更新する必要がなくなる。許可ポリシーは、一致するユーザー属性を持つユーザーのみにアクセス権を付与します。

例えば、ユーザーの役職に基づいて、特定の S3 バケットへのアクセスを制御する必要がある場合があります。この場合、定義された役職のユーザーのみにこれらのファイルへのアクセスを許可する許可ポリシーを作成できます。詳細については、「IAM チュートリアル: ABAC で SAML セッションタグを使用する」を参照してください。

- 属性の値に基づいて許可を許可または拒否するポリシーに、プリンシパルタグとしてこれらの属性を渡すことができる。

Amazon Cognito の ID プールを用いたアクセスコントロールへの属性の使用

アクセスコントロールに属性を使用する前に、以下の前提条件を満たしていることを確認します。

- AWS アカウント
- ユーザープール
- ID プール
- SDKE の設定
- 統合された ID プロバイダー
- 認証情報

アクセスコントロールに属性を使用するには、データソースとして設定したクレームによって、選択したタグキーの値が設定されます。Amazon Cognito はタグキーと値をユーザーのセッションに適用します。IAM ポリシーでは、`${aws:PrincipalTag/tagkey}` 条件からユーザーのアクセスを評価できます。IAM は、ユーザーのタグの値をポリシーに照らして評価します。

ユーザーに認証情報を渡したい IAM ロールを準備する必要があります。これらのロールの信頼ポリシーで、Amazon Cognito がユーザーのロールを引き継ぐことを許可されている必要があります。アクセスコントロール用の属性については、Amazon Cognito がユーザーの一時セッションにプリンシパルタグを適用することも許可しなければなりません。アクション AssumeRoleWithWebIdentity を使って、ロールを引き受ける権限を付与します。権限のみのアクション `sts:TagSession` を使って、ユーザーのセッションにタグを付ける権限を付与します。詳細については、「AWS Identity and Access Management ユーザーガイド」の「AWS Security Token Service でのタグ付けの規則」を参照してください。たとえば、`sts:AssumeRoleWithWebIdentity` と `sts:TagSession` 権限を Amazon Cognito サービスプリンシパル `cognito-identity.amazonaws.com` に付与する信頼ポリシーの例については、「アクセスコントロールポリシーへの属性の使用例」を参照してください。

コンソールでアクセスコントロールの属性を無効にするには

1. Amazon Cognito コンソールにサインインし、アイデンティティプールを選択します。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. ID プロバイダーを見つけます。編集する ID プロバイダーを選択します。新しい IdP を追加する場合は、[ID プロバイダーを追加] を選択します。
4. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] で [編集] を選択します。
   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。
   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。
5. [Save Changes] (変更を保存) を選択します。

アクセスコントロールポリシーへの属性の使用例

ある企業の法務部門の従業員が、その部門に属しており、セキュリティレベルで分類されているバケット内のすべてのファイルをリストする必要があるというシナリオを考えてみましょう。この従業員が ID プロバイダーから取得するトークンには、以下のクレームが含まれているとします。

クレーム

```
{ .
  .
"sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
"department" : "legal",
"clearance" : "confidential",
 .
 .
}
```

これらの属性は、タグにマップして、IAM 許可ポリシーでプリンシパルタグとして参照できます。これで、ID プロバイダー側でユーザープロファイルを変更することによって、アクセスを管理できるようになります。または、ポリシー自体を変更せずに、名前またはタグを使用することによって、リソース側の属性を変更することもできます。

以下の許可ポリシーは、以下の 2 つを行います。

- ユーザーの部門名に一致するプレフィックスで終わるすべての S3 バケットへのリストアクセスを許可します。
- ファイルのクリアランスタグがユーザーのクリアランス属性と一致している限り、これらのバケット内のファイルに対する読み取りアクセスを許可する。

アクセス許可ポリシー

```
   {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:List*",
            "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}"
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObject*",
            "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}/*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/clearance}"
                }
            }
        }
    ]
}
```

信頼ポリシーは、誰がこのロールを引き受けられるのかを決定します。信頼関係ポリシーは、アクセスを許可するための `sts:AssumeRoleWithWebIdentity` と `sts:TagSession` の使用を可能にし

ます。これは、ポリシーをユーザーが作成したアイデンティティプールに制限する条件を追加し、それが認証されたロール用であることを確実にします。

信頼ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "IDENTITY-POOL-ID"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

アクセスコントロールの属性をオフにする (コンソール)

アクセスコントロールの属性を非アクティブにするには、以下の手順に従います。

コンソールでアクセスコントロールの属性を非アクティブにするには

1. Amazon Cognito コンソールにサインインし、アイデンティティプールを選択します。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. ID プロバイダーを見つけます。編集する ID プロバイダーを選択します。

4. [アクセスコントロールの属性] で、[編集] を選択します。
5. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
6. [Save Changes] (変更を保存) を選択します。

デフォルトのプロバイダーマッピング

以下の表には、Amazon Cognito がサポートする認証プロバイダーのデフォルトマッピング情報が記載されています。

プロバイダー	トークンタイプ	プリンシパルタグの値	例
Amazon Cognito ユーザープール	ID トークン	aud (クライアント ID) および sub (ユーザー ID)	"6jk8ltokc7ac9es6jrtg9q572f"、"57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	アクセストークン	aud(app_id)、sub(user_id)	「492844718097981」、「112177216992379」
Google	ID トークン	aud (クライアント ID) および sub (ユーザー ID)	「620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com」、「109220063452404746097」
SAML	アサーション	「http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier」、「http://schemas.xmlsoap.org/ws/2005/0	"auth0|5e28d196f8f55a0eaaa95de3"、"user123@gmail.com"

プロバイダー	トークンタイプ	プリンシパルタグの値	例
		5/identity/claims/name」	
Apple	ID トークン	aud (クライアント ID) および sub (ユーザー ID)	「com.amazonaws.ec2-54-80-172-243.compute-1.client」、「001968.a6ca34e9c1e742458a26cf8005854be9.0733」
Amazon	アクセストークン	aud (Amzn Dev Ac のクライアント ID)、user_id (ユーザー ID)	「amzn1.application-oa2-client.9d70d938 2d3446108aee3dd763a0fa6」、「amzn1.account.Aghnifjqmfsbg3g6xcPVB35orqaa」
標準 OIDC プロバイダー	ID トークンとアクセストークン	aud (client_id として)、sub (ユーザー ID として)	「620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com」、「109220063452404746097」
Twitter	アクセストークン	aud (アプリID; アプリシークレット)、サブ (ユーザー ID)	"DfwifTtKEX1FiIBRnOTIR0CFK; Xgj5xb8xIrIVCPjXgLIdkW7fXmwcJJrFvnoK9gwZkLexo1y5z1"、"1269003884292222976"

プロバイダー	トークンタイプ	プリンシパルタグの値	例
DevAuth	マップ	該当なし	「tag1」、「tag2」

> **Note**
>
> デフォルト属性マッピングオプションは、[Tag Key for Principal] (プリンシパルのタグキー) と [Attribute] (属性) の名前に自動で入力されます。デフォルトマッピングを変更することはできません。

ロールベースアクセスコントロールの使用

Amazon Cognito ID プールは、認証されたユーザーに、 AWS リソースにアクセスするための権限が制限された一時的な認証情報のセットを割り当てます。各ユーザーの許可は、作成された IAM ロールを介して制御されます。ユーザーの ID トークンのクレームに基づいて、各ユーザーのロールを選択するルールを定義できます。認証されたユーザー用のデフォルトのロールを定義できます。また、認証されていないゲストユーザーに対して、権限が制限された個別の IAM ロールを定義することもできます。

ロールマッピング用のロールの作成

各ロールに適切な信頼ポリシーを追加し、Amazon Cognito が ID プール内の認証されたユーザー限定でそのポリシーを引き受けるようにすることが重要です。このような信頼ポリシーの例を示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
```

```
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

このポリシーでは、cognito-identity.amazonaws.com のフェデレーティッドユーザー (OpenID Connect トークンの発行者) が、このロールを引き受けることができます。さらに、ポリシーでは、ID プールに一致させるためにトークンの aud が制限されます (この場合は ID プール ID)。最後に、このポリシーは、Amazon Cognito によって発行されたトークンの複数の値からなる amr クレームの配列メンバーの 1 つを指定します。GetOpenIdToken API アクションは authenticated 値を含みます。

pass-role 許可の付与

ユーザーが、ID プールでユーザーの既存のアクセス許可を超えるアクセス許可でロールを設定できるようにするには、set-identity-pool-roles API にロールを渡すための iam:PassRole アクセス許可をそのユーザーに付与します。例えば、ユーザーが Amazon S3 に書き込むことはできなくても、ID プールでユーザーが設定する IAM ロールが Amazon S3 への書き込み許可を付与するという場合、ユーザーは iam:PassRole 許可がそのロールに付与されている場合にのみ、このロールを設定できます。次のサンプルポリシー例は、iam:PassRole 権限を付与する方法を示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
```

```
}
```

このサンプルポリシーでは、iam:PassRole ロールに myS3WriteAccessRole アクセス権限を付与します。このロールは、それ自体の Amazon リソースネーム (ARN) を使用して指定します。また、このポリシーをユーザーにアタッチする必要があります。詳細については、「管理ポリシーの使用」を参照してください。

> **Note**
>
> Lambda 関数はリソースベースのポリシーを使用し、このポリシーは Lambda 関数自体に直接アタッチされます。Lambda 関数を呼び出すルールを作成するときはロールを渡さないため、ルールを作成するユーザーに iam:PassRole 許可は必要ありません。Lambda 関数の認証に関する詳細については、「アクセス許可の管理: Lambda 関数ポリシーを使用する」を参照してください。

ユーザーにロールを割り当てるためのトークンの使用

Amazon Cognito ユーザープール経由でログインするユーザーの場合、ユーザープールによって割り当てられた ID トークンにロールを渡すことができます。これらのロールは、ID トークンの以下のクレームに表示されます。

- cognito:preferred_role クレームはロール ARN です。
- cognito:roles クレームは、許可されたロール ARN のセットを含むカンマ区切りの文字列です。

クレームは以下のように設定されます。

- cognito:preferred_role クレームは、最高 (最も低い) Precedence 値のグループのロールに設定されます。許可されているロールが 1 つだけである場合、cognito:preferred_role はそのロールに設定されます。複数のロールがあり、いずれの単一のロールにも最良の優先順位がない場合、このクレームは設定されません。
- cognito:roles クレームは、少なくとも 1 つのロールがある場合に設定されます。

ロールの割り当てにトークンを使用するときに、ユーザーに割り当てることができる複数のロールがある場合は、Amazon Cognito ID プール (フェデレーティッド ID) が以下のようにロールを選択します。

- パラメータが設定されていて、cognito:rolesクレーム内のロールと一致する場合は、GetCredentialsForIdentityCustomRoleArnパラメータを使用します。このパラメータが cognito:roles のロールと一致しない場合は、アクセスを拒否します。
- cognito:preferred_role クレームが設定されている場合は、それを使用します。
- cognito:preferred_role クレームが設定されていない場合、cognito:rolesクレームが設定され、 の呼び出しで指定CustomRoleArnされていない場合GetCredentialsForIdentity、コンソールのロール解決設定または AmbiguousRoleResolutionフィールド (SetIdentityPoolRolesAPI の RoleMappingsパラメータ) を使用して、割り当てるロールを決定します。

ルールベースのマッピングを使用してユーザーにロールを割り当てる

ルールは、IAM ロールに ID プロバイダートークンからのクレームをマップすることを許可します。

各ルールは、トークンクレーム (Amazon Cognito ユーザープールからの ID トークンのユーザー属性など)、一致タイプ、値、および IAM ロールを指定します。一致タイプは、Equals、NotEqual、StartsWith、または Contains となります。クレームに一致する値をユーザーが持っている場合、そのユーザーは、認証情報を取得するときにそのロールを引き受けることができます。例えば、custom:dept カスタム属性値が Sales のユーザーに特定の IAM ロールを割り当てるルールを作成できます。

> **Note**
>
> ルールの設定で、カスタム属性は標準属性と区別するために、custom: プレフィックスを必要とします。

CustomRoleArn が順序を上書きするように指定されている場合を除き、ルールは順序に従って評価され、一致する最初のルールの IAM ロールが使用されます。Amazon Cognito ユーザープールでのユーザー属性の詳細については、「ユーザープール属性」を参照してください。

ID プール (フェデレーティッド ID) コンソールでは、認証プロバイダー用に複数のルールを設定できます。ルールは順序に従って適用されます。ルールをドラッグして、それらの順序を変更できます。一致する最初のルールが優先されます。一致タイプが NotEqual で、クレームが存在しない場合、ルールは評価されません。一致するルールがない場合は、[ロールの解決] 設定が [デフォルトの認証されたロールを使用する] または [リクエストの拒否] に適用されます。

API および CLI では、SetIdentityPoolRolesAPI の RoleMappingsパラメータで指定されている RoleMappingタイプの AmbiguousRoleResolutionフィールドでルールが一致しない場合に割り当てるロールを指定できます。

OpenID Connect (OIDC) および SAML ID プロバイダーのルールベースのマッピングは、 AWS CLI または API で RoleMapping タイプの RulesConfigurationフィールドを使用して設定できます。このフィールドは、 SetIdentityPoolRoles API の RoleMappingsパラメータで指定できます。 AWS Management Console 現在、 では、OIDC または SAML プロバイダーのルールを追加することはできません。

例えば、次の AWS CLI コマンドは、OIDC IdP によって認証された Sacramento ロケーションのarn:aws:iam::123456789012:role/Sacramento_team_S3_adminユーザーにロールを割り当てるルールを追加しますarn:aws:iam::123456789012:oidc-provider/myOIDCIdP。

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

role-mapping.json の内容:

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
        }
```

```
    }
}
```

ID プールに対して設定された各ユーザープールまたはその他の認証プロバイダーの場合、最大 25 のルールを作成できます。この制限は調整できません。詳細については、「Amazon Cognito のクォータ」を参照してください。

ルールベースのマッピングで使用するトークンクレーム

Amazon Cognito

Amazon Cognito ID トークンは JSON Web トークン (JWT) として表されます。トークンには、認証ユーザーの ID に関するクレームが含まれます。たとえば、`name`、`family_name`、`phone_number` などです。標準クレームに関する詳細については、「OpenID Connect 仕様」を参照してください。標準クレームとは別に、以下のような Amazon Cognito に固有の追加のクレームがあります。

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

以下のクレームとそのクレームに使用できる値を、Login with Amazon で使用できます。

- `iss`: www.amazon.com
- `aud`: App Id
- `sub`: Login with Amazon トークンからの `sub`

Facebook

以下のクレームとそのクレームに使用できる値を、Facebook で使用できます。

- `iss`: graph.facebook.com
- `aud`: App Id
- `sub`: Facebook トークンからの `sub`

Google

Google トークンには、OpenID Connect 仕様からの標準クレームが含まれます。OpenID トークンのクレームはすべて、ルールベースのマッピングに使用できます。Google トークンで使用できるクレームについては、Google の OpenID Connect サイトを参照してください。

Apple

Apple トークンには、OpenID Connect specification からの標準クレームが含まれます。Apple トークンから入手できるクレームの詳細については、Apple のドキュメントの「Apple でサインインを使用してユーザーを認証する」を参照してください。`email` は Apple のトークンには必ずしも含まれているわけではありません。

OpenID

Open ID トークンのクレームはすべて、ルールベースのマッピングに使用できます。標準クレームに関する詳細については、「OpenID Connect 仕様」を参照してください。使用できる追加クレームについては、ご使用の OpenID プロバイダーのドキュメントを参照してください。

SAML

クレームは受信した SAML アサーションから分析されます。SAML アサーションで利用可能なすべてのクレームは、ルールベースのマッピングで使用できます。

ロールベースのアクセスコントロールのベストプラクティス

> **Important**
>
> ロールにマッピングするクレームがエンドユーザーによって変更できる場合、いずれのエンドユーザーもロールを引き受け、それに応じてポリシーを設定できます。エンドユーザーによって直接設定できないクレームのみを、昇格されたアクセス権限のあるロールにマッピングします。Amazon Cognito ユーザープールで、各ユーザー属性にアプリごとの読み取りおよび書き込み許可を設定できます。

> **Important**
>
> Amazon Cognito ユーザープールのグループにロールを設定する場合、これらのロールはユーザーの ID トークンを通じて渡されます。これらのロールを使用するには、ID プールの認証ロールの選択に対して [トークンからロールを選択する] を設定する必要があります。

コンソールのロール解決設定と SetIdentityPoolRoles API の RoleMappingsパラメータを使用して、トークンから正しいロールを特定できない場合のデフォルトの動作を指定できます。

認証情報の取得

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。このセクションでは、認証情報を取得する方法と、ID プールから Amazon Cognito ID を取得する方法について説明します。

Amazon Cognito は、認証されている ID と認証されていない ID の両方をサポートします。認証されていないユーザーは ID が検証されないため、このロールはアプリケーションのゲストユーザーに適切です。または、ユーザーの ID が検証されているかどうかが重要ではない場合に適切です。認証されているユーザーは、アプリケーションにログイン際にサードパーティーの ID プロバイダーまたはユーザープールを通じて ID が検証されます。リソースの許可の範囲を適切に設定し、認証されていないユーザーからのアクセスを許可しないようにします。

Amazon Cognito ID は認証情報ではありません。これらは、 AWS Security Token Service () のウェブ ID フェデレーションサポートを使用して認証情報と交換されますAWS STS。アプリケーションユーザーによる AWS 認証情報の取得に推奨される方法は、AWS.CognitoIdentityCredentials の使用です。認証情報オブジェクトの ID は、 を使用して認証情報と交換されます AWS STS。

> **Note**
>
> 2015 年 2 月より前に作成したアイデンティティプールの場合で、ロールをパラメータとせずに AWS.CognitoIdentityCredentials コンストラクターを使用するには、再度そのロールと ID プールの関連付けを行う必要があります。これを実行するには、Amazon Cognito コンソールを開き、[Manage identity pools] (ID プールの管理) をクリックしてから ID プールを選択し、[Edit identity Pool] (ID プールの編集) をクリックします。さらに、認証済みと認証済みでないロールを指定した上で変更を保存します。

ウェブ ID 認証情報プロバイダーは、 AWS SDK のデフォルトの認証情報プロバイダーチェーンの一部です。SDK または の AWS ローカルconfigファイルに ID プールトークンを設定するには AWS CLI、web_identity_token_fileプロファイルエントリを追加します。「 SDK とツールのリファレンスガイド」の「ロール認証情報プロバイダーを引き受ける AWS SDKs 」を参照してください。

SDK にウェブ ID 認証情報を入力する方法の詳細については、SDK ディベロッパーガイドを参照してください。最良の結果を得るには、 に組み込まれている ID プール統合からプロジェクトを開始します AWS Amplify。

AWS ID プールで認証情報を取得および設定するための SDK リソース

- Amplify Dev Center のアイデンティティプールフェデレーション (Android)
- Amplify Dev Center のアイデンティティプールフェデレーション (iOS)
- デ AWS SDK for JavaScript ベロッパーガイドのAmazon Cognito Identity を使用したユーザーの認証」
- AWS SDK for .NET デベロッパーガイドのAmazon Cognito 認証情報プロバイダー」
- デ AWS SDK for Go ベロッパーガイドの「認証情報をプログラムで指定する」
- 「 デベロッパーガイド」の「コードに一時的な認証情報を指定する AWS SDK for Java 2.x 」
- assumeRoleWithWebIdentityCredentialProvider AWS SDK for PHP デベロッパーガイドの プロバイダー
- AWS SDK for Python (Boto3) ドキュメントの「ウェブ ID プロバイダーの役割を引き受ける」
- 「 デベロッパーガイド」の「認証情報とデフォルトのリージョンの指定 AWS SDK for Rust 」

以下のセクションでは、いくつかのレガシー AWS SDKs。

Android

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。Amazon Cognito は、認証されている ID と認証されていない ID の両方をサポートします。アプリに AWS 認証情報を提供するには、以下の手順に従います。

Android アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「認証」を参照してください。

Amazon Cognito アイデンティティの取得

認証されていないユーザーを許可している場合は、エンドユーザーの固有 Amazon Cognito 識別子 (アイデンティティ ID) をただちに取得できます。ユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で、アイデンティティ ID を取得できます。

```
String identityId = credentialsProvider.getIdentityId();
```

```
Log.d("LogTag", "my ID is " + identityId);
```

> **Note**
>
> アプリケーションのメインスレッドで、getIdentityId()、refresh()、または getCredentials() を呼び出さないでください。Android 3.0 (API レベル 11) NetworkOnMainThreadException以降、メインアプリケーションスレッドでネットワーク I/O を実行すると、アプリケーションは自動的に失敗し、 をスローします。AsyncTask を使用して、バックグラウンドスレッドにコードを移動する必要があります。詳細については、「Android のドキュメント」を参照してください。getCachedIdentityId() を呼び出して ID を取得することもできますが、この操作が可能なのは、既に ID がローカルにキャッシュされている場合のみです。それ以外の場合、メソッドは null を返します。

iOS - Objective-C

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。Amazon Cognito ID プールは、認証された ID と認証されていない ID の両方をサポートします。アプリに AWS 認証情報を提供するには、次のステップを実行します。

iOS アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「Swift 認証」と「Flutter 認証」を参照してください。

Amazon Cognito アイデンティティの取得

エンドユーザー用の固有 Amazon Cognito 識別子 (アイデンティティ ID) は、認証されていないユーザーを許可している場合はただちに取得、またはユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で取得することができます。

```
// Retrieve your Amazon Cognito ID
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
```

```
    }
    return nil;
}];
```

> **Note**
>
> `getIdentityId` は非同期呼び出しです。アイデンティティ ID が既にプロバイダーで設定されている場合は、`credentialsProvider.identityId` を呼び出して (ローカルにキャッシュされた) その ID を取得できます。ただし、プロバイダーでアイデンティティ ID が設定されていない場合、`credentialsProvider.identityId` を呼び出すと `nil` が返されます。詳細については、Amplify iOS SDK リファレンスを参照してください。

iOS - Swift

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。Amazon Cognito は、認証されている ID と認証されていない ID の両方をサポートします。アプリに AWS 認証情報を提供するには、以下の手順に従います。

iOS アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「Swift 認証」を参照してください。

Amazon Cognito アイデンティティの取得

エンドユーザー用の固有 Amazon Cognito 識別子 (アイデンティティ ID) は、認証されていないユーザーを許可している場合はただちに取得、またはユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で取得することができます。

```
// Retrieve your Amazon Cognito ID
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
```

```
})
```

> **Note**
>
> getIdentityId は非同期呼び出しです。アイデンティティ ID が既にプロバイダーで設定されている場合は、credentialsProvider.identityId を呼び出して (ローカルにキャッシュされた) その ID を取得できます。ただし、プロバイダーでアイデンティティ ID が設定されていない場合、credentialsProvider.identityId を呼び出すと nil が返されます。詳細については、Amplify iOS SDK リファレンスを参照してください。

JavaScript

ID プールをまだ作成して AWS.CognitoIdentityCredentials を使用する前に Amazon Cognito コンソールで ID プールを作成してください。

ID プロバイダーで ID プールを設定すると、AWS.CognitoIdentityCredentials を使用してユーザーを認証できます。AWS.CognitoIdentityCredentials を使用するようにアプリケーションを設定するには、credentials またはサービス別の設定の AWS.Config プロパティを設定します。次の例では AWS.Config を使用しています。

```
// Set the region where your identity pool exists (us-east-1, eu-west-1)
AWS.config.region = 'us-east-1';

// Configure the credentials provider to use your identity pool
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: { // optional tokens, used for authenticated login
        'graph.facebook.com': 'FBTOKEN',
        'www.amazon.com': 'AMAZONTOKEN',
        'accounts.google.com': 'GOOGLETOKEN',
        'appleid.apple.com': 'APPLETOKEN'
    }
});

// Make the call to obtain credentials
AWS.config.credentials.get(function(){

    // Credentials will be available when this function is called.
    var accessKeyId = AWS.config.credentials.accessKeyId;
    var secretAccessKey = AWS.config.credentials.secretAccessKey;
```

```
    var sessionToken = AWS.config.credentials.sessionToken;

});
```

オプションの Logins プロパティは、ID プロバイダー名の ID トークンへのマッピングです。ID プロバイダーからのトークンの取得方法は、使用するプロバイダーによって異なります。たとえば、Facebook を ID プロバイダーとして使用する場合は、FB.loginFacebook SDK の 関数を使用して ID プロバイダートークンを取得します。

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
          IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030',
          Logins: {
            'graph.facebook.com': response.authResponse.accessToken
          }
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

Amazon Cognito アイデンティティの取得

エンドユーザー用の固有 Amazon Cognito 識別子 (アイデンティティ ID) は、認証されていないユーザーを許可している場合はただちに取得、またはユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で取得することができます。

```
var identityId = AWS.config.credentials.identityId;
```

Unity

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。Amazon Cognito は、認証されている ID と認証されていない ID の両方をサポートします。アプリに AWS 認証情報を提供するには、以下の手順に従います。

AWS SDK for Unity も、AWS SDK for .NET の一部になりました。で Amazon Cognito の使用を開始するには AWS SDK for .NET、「 AWS SDK for .NET デベロッパーガイド」のAmazon Cognito 認証

情報プロバイダー」を参照してください。または、 でアプリケーションを構築するためのオプションについては、「Amplify Dev Center」を参照してください AWS Amplify。

Amazon Cognito アイデンティティの取得

エンドユーザー用の固有 Amazon Cognito 識別子 (アイデンティティ ID) は、認証されていないユーザーを許可している場合はただちに取得、またはユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で取得することができます。

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
    string identityId = result.Response;
});
```

Xamarin

Amazon Cognito を使用して、権限が制限された一時的な認証情報をアプリケーションに配信し、ユーザーが AWS リソースにアクセスできるようにします。Amazon Cognito は、認証されている ID と認証されていない ID の両方をサポートします。アプリに AWS 認証情報を提供するには、以下の手順に従います。

AWS SDK for Xamarin も、AWS SDK for .NET の一部になりました。で Amazon Cognito の使用を開始するには AWS SDK for .NET、「 AWS SDK for .NET デベロッパーガイド」のAmazon Cognito 認証情報プロバイダー」を参照してください。または、 でアプリケーションを構築するためのオプションについては、「Amplify Dev Center」を参照してください AWS Amplify。

> **Note**
>
> 注: 2015 年 2 月より前に作成した ID プールで、ロールをパラメータとせずに、このコンストラクターを使用するには、再度そのロールと ID プールの関連付けを行う必要があります。これを実行するには、Amazon Cognito コンソールを開き、[Manage identity pools] (ID プールの管理) をクリックしてから ID プールを選択し、[Edit identity Pool] (ID プールの編集) をクリックします。さらに、認証済みと認証済みでないロールを指定した上で変更を保存します。

Amazon Cognito アイデンティティの取得

エンドユーザー用の固有 Amazon Cognito 識別子 (アイデンティティ ID) は、認証されていないユーザーを許可している場合はただちに取得、またはユーザーを認証している場合は、認証情報プロバイダーでログイントークンを設定した後で取得することができます。

```
var identityId = await credentials.GetIdentityIdAsync();
```

AWS サービスへのアクセス

Amazon Cognito 認証情報プロバイダーを設定し、 AWS 認証情報を取得したら、 AWS のサービスクライアントを作成できます。

AWS クライアントを作成するための SDK リソース

- AWS SDK for C++ デベロッパーガイドのAWS クライアント設定
- デ AWS SDK for Go ベロッパーガイドの「 での AWS SDK for Go V2 の使用 AWS のサービス」
- デ AWS SDK for Java 2.x ベロッパーガイドの「HTTP クライアントの設定」
- デ AWS SDK for JavaScript ベロッパーガイドの「サービスオブジェクトの作成と呼び出し」
- AWS SDK for Python (Boto3) ドキュメント内のクライアントの作成
- 「 デベロッパーガイド」の「サービスクライアントの作成 AWS SDK for Rust 」
- AWS SDK for Swift デベロッパーガイドのクライアントの使用

以下のスニペットは Amazon DynamoDB クライアントを初期化します。

Android

Android アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「認証」を参照してください。

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と Mobile AWS SDK の一時的な制限付き特権 AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

iOS - Objective-C

iOS アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「Swift 認証」と「Flutter 認証」を参照してください。

```
// create a configuration that uses the provider
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
 configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];
// get a client with the default service configuration
AWSDynamoDB *dynamoDB = [AWSDynamoDB defaultDynamoDB];
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と Mobile AWS SDK の一時的な制限付き特権 AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

iOS - Swift

iOS アプリケーションで Amazon Cognito ID プールを使用するには、 をセットアップします AWS Amplify。詳細については、Amplify Dev Center の「Swift 認証」を参照してください。

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()

// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と Mobile AWS SDK の一時的な制限付き特権 AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と、 AWS Mobile SDK の権限が制限された一時的な AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

Unity

AWS SDK for Unity も、AWS SDK for .NET の一部になりました。で Amazon Cognito の使用を開始するには AWS SDK for .NET、「 AWS SDK for .NET デベロッパーガイド」のAmazon Cognito 認証情報プロバイダー」を参照してください。または、 でアプリケーションを構築するためのオプションについては、「Amplify Dev Center」を参照してください AWS Amplify。

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と、 AWS Mobile SDK の権限が制限された一時的な AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

Xamarin

AWS SDK for Xamarin も、AWS SDK for .NET の一部になりました。で Amazon Cognito の使用を開始するには AWS SDK for .NET、「 AWS SDK for .NET デベロッパーガイド」のAmazon Cognito 認証情報プロバイダー」を参照してください。または、 でアプリケーションを構築するためのオプションについては、「Amplify Dev Center」を参照してください AWS Amplify。

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

認証情報プロバイダーは Amazon Cognito と通信し、認証されたユーザーと認証されていないユーザーの一意の識別子と、 AWS Mobile SDK の権限が制限された一時的な AWS 認証情報の両方を取得します。取得された認証情報は 1 時間有効で、有効期限が切れるとプロバイダーによって更新されます。

ID プール外部 ID プロバイダー

`logins` プロパティを使用すると、ID プロバイダー (IdP) から受信した認証情報を設定できます。さらに、ID プールを複数の に関連付けることができます IdPs。例えば、Facebook と Google の両方

のトークンを `logins` プロパティに設定して、一意の Amazon Cognito アイデンティティが両方の IdP のログインに関連付けられるようにすることができます。ユーザーはどちらのアカウントでも認証できますが、Amazon Cognito は同じユーザー ID を返します。

次の手順では、Amazon Cognito ID プールがサポート IdPs する による認証について説明します。

トピック

- Facebook を ID プール IdP としてセットアップする
- Login with Amazon を ID プール IdP としてセットアップする
- Google を ID プール IdP としてセットアップする
- アイデンティティプール IdP として「Apple でサインイン」を設定する
- OIDC プロバイダーを ID プール IdP としてセットアップする
- SAML プロバイダーを ID プール IdP としてセットアップする

Facebook を ID プール IdP としてセットアップする

Amazon Cognito ID プールは、モバイルアプリケーションユーザーにフェデレートされた認証を提供するために Facebook と統合します。このセクションでは、IdP として Facebook に対してアプリケーションを登録し、セットアップする方法について説明します。

Facebook のセットアップ

Facebook ユーザーを認証し、Facebook API と統合するには、事前にアプリケーションを Facebook に登録します。

Facebook デベロッパーポータルは、アプリケーションのセットアップに役立ちます。Amazon Cognito アイデンティティプールに Facebook を統合する前に、次の手順を実行してください。

Facebook のセットアップ

1. Facebook デベロッパーポータルで、Facebook 認証情報を使用してログインします。
2. アプリ メニューの Add a New App (新しいアプリを追加する)を選択します。
3. プラットフォームを選択し、クイックスタートプロセスを完了します。

Android

Android アプリと Facebook ログインとの統合に関する詳細情報については、Facebook 入門ガイドを参照してください。

iOS - Objective-C

iOS Objective-C アプリと Facebook ログインとの統合に関する詳細情報については、Facebook 入門ガイドを参照してください。

iOS - Swift

iOS Swift アプリと Facebook ログインとの統合に関する詳細情報については、Facebook 入門ガイドを参照してください。

JavaScript

JavaScript ウェブアプリを Facebook ログインと統合する方法の詳細については、「Facebook 入門ガイド」を参照してください。

Unity

Unity アプリと Facebook ログインとの統合に関する詳細情報については、Facebook 入門ガイドを参照してください。

Xamarin

Facebook 認証を追加するには、まず以下の適切なフローに従って、アプリケーションに Facebook SDK を統合します。Amazon Cognito ID プールは、Amazon Cognito アイデンティティに関連付けられた一意のユーザー識別子を生成するために Facebook アクセストークンを使用します。

- Facebook iOS SDK by Xamarin
- Facebook Android SDK by Xamarin

Amazon Cognito アイデンティティプールコンソールでアイデンティティプロバイダーを設定する

以下の手順を使用して、ID プロバイダーを設定します。

Facebook ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Facebook] を選択します。
5. Meta for Developers で作成した OAuth プロジェクトのアプリ ID を入力します。詳細については、Meta for Developers ドキュメントの「Facebook ログイン」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。

     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。
8. [変更を保存] を選択します。

Facebook の使用

Android

Facebook 認証を追加するには、まず Facebook ガイドに従って、アプリケーションに Facebook SDK を統合します。次に、[Login with Facebook] ボタンを Android ユーザーインターフェイスに追加します。Facebook SDK は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトのアクセストークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じてユーザーに他の AWS リソースへのアクセスを許可します。

Facebook SDK でユーザーを認証したら、Amazon Cognito 認証情報プロバイダーにセッショントークンを追加します。

Facebook SDK 4.0 以降:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK 4.0 以前:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook ログインプロセスにより、その SDK でシングルトンセッションが初期化されます。Facebook セッションオブジェクトには、認証されたエンドユーザーの AWS 認証情報を生成するために Amazon Cognito が使用する OAuth トークンが含まれています。Amazon Cognito は、この特定の Facebook アイデンティティに一致するユーザーの存在についてユーザーデータベースをチェックするためにもこのトークンを使用します。既にユーザーに存在する場合、API は既存の ID を返します。それ以外の場合、API は新しい ID を返します。クライアント SDK は、ローカルデバイスで ID を自動的にキャッシュします。

> **Note**
>
> ログインマップを設定したら、 `refresh`または を呼び出しgetて AWS 認証情報を取得します。

iOS - Objective-C

Facebook 認証を追加するには、まず Facebook ガイドに従って、アプリケーションに Facebook SDK を統合します。次に、[Login with Facebook (Facebook でログイン)] ボタンを ユーザーインターフェイスに追加します。Facebook SDK は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトからのアクセストークンを使用してユーザーを認証し、一意の Amazon Cognito ID プール (フェデレーティッド ID) にバインドします。

Amazon Cognito に Facebook アクセストークンを提供するには、AWSIdentityProviderManager プロトコルを実装します。

`logins` メソッドを実装するときは、`AWSIdentityProviderFacebook` を含むディクショナリを返します。以下のコード例にあるように、このディクショナリはキーとして機能し、認証された Facebook ユーザーからの現在のアクセストークンが値として機能します。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook : token }];
    }else{
            return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                                              code:-1
                                                          userInfo:@{@"error":@"No current
 Facebook access token"}]];
    }
}
```

`AWSCognitoCredentialsProvider` をインスタンス化するときに、コンストラクターで `AWSIdentityProviderManager` の値として `identityProviderManager` を実装するクラスを渡します。詳細については、AWSCognitoCredentialsProviderリファレンスページに移動し、initWithRegionタイプ：identityPoolId：identityProviderManager を選択します。

iOS - Swift

Facebook 認証を追加するには、まず Facebook ガイドに従って、アプリケーションに Facebook SDK を統合します。次に、[Login with Facebook (Facebook でログイン)] ボタンを ユーザーインターフェイスに追加します。Facebook SDK は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトからのアクセストークンを使用してユーザーを認証し、一意の Amazon Cognito ID プール (フェデレーティッド ID) にバインドします。

Amazon Cognito に Facebook アクセストークンを提供するには、AWSIdentityProviderManager プロトコルを実装します。

logins メソッドを実装するときは、AWSIdentityProviderFacebook を含むディクショナリを返します。以下のコード例にあるように、このディクショナリはキーとして機能し、認証された Facebook ユーザーからの現在のアクセストークンが値として機能します。

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }
}
```

AWSCognitoCredentialsProvider をインスタンス化するときに、コンストラクターで AWSIdentityProviderManager の値として identityProviderManager を実装するクラスを渡します。詳細については、AWSCognitoCredentialsProviderリファレンスページに移動し、initWithRegionタイプ：identityPoolId：identityProviderManager を選択します。

JavaScript

Facebook 認証を追加するには、「ウェブ用の Facebook ログイン」に従って、ウェブサイトに「Facebook でログイン」ボタンを追加します。Facebook SDK は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトのアクセストークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じてユーザーに他の AWS リソースへのアクセスを許可します。

Facebook SDK でユーザーを認証したら、Amazon Cognito 認証情報プロバイダーにセッショントークンを追加します。

```
FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {

    console.log('You are now logged in.');

    // Add the Facebook access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
```

```
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'graph.facebook.com': response.authResponse.accessToken
      }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook SDK は、認証されたエンドユーザーの AWS 認証情報を生成するために Amazon Cognito が使用する OAuth トークンを取得します。Amazon Cognito は、この特定の Facebook アイデンティティに一致するユーザーの存在についてユーザーデータベースをチェックするためにもこのトークンを使用します。既にユーザーに存在する場合、API は既存の ID を返します。それ以外の場合、新しい ID が返されます。ID はローカルデバイスでクライアント SDK によって自動的にキャッシュされます。

> **Note**
>
> ログインマップを設定した後、`refresh` または `get` を呼び出して、認証情報を取得します。コード例については、JavaScript README ファイル の「ユースケース 17、ユーザープールと Cognito ID の統合」を参照してください。

Unity

Facebook 認証を追加するには、まず Facebook ガイドに従って、アプリケーションに Facebook SDK を統合します。Amazon Cognito は FB オブジェクトからの Facebook アクセストークンを使用して、Amazon Cognito アイデンティティに関連付けられた一意のユーザー 識別子を生成します。

Facebook SDK でユーザーを認証したら、Amazon Cognito 認証情報プロバイダーにセッショントークンを追加します。

```
void Start()
```

```
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
 AccessToken.CurrentAccessToken.TokenString);
}
```

FB.AccessToken を使用する前に、FB.Login() を呼び出し、FB.IsLoggedIn が true であることを確認してください。

Xamarin

Xamarin for Android:

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
 LoginResult &gt; () {
      HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
```

```
        //open new activity
      },
      HandleCancel = () = &gt; {
        //throw error message
      },
      HandleError = loginError = &gt; {
        //throw error message
      }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
      "public_profile"
    });
  }
```

Xamarin for iOS:

```
public void InitializeFacebook() {
  LoginManager login = new LoginManager();
  login.LogInWithReadPermissions(readPermissions.ToArray(),
 delegate(LoginManagerLoginResult result, NSError error) {
    if (error != null) {
      //throw error message
    } else if (result.IsCancelled) {
      //throw error message
    } else {
      var accessToken = loginResult.AccessToken;
      credentials.AddLogin("graph.facebook.com", accessToken.Token);
      //open new view controller
    }
  });
}
```

Login with Amazon を ID プール IdP としてセットアップする

Amazon Cognito は Login with Amazon と統合して、モバイルアプリケーションとウェブアプリケーションのユーザーにフェデレートされた認証を提供します。このセクションでは、ID プロバイダー (IdP) として Login with Amazon を使用してアプリケーションを登録し、セットアップする方法について説明します。

デベロッパーポータルで Amazon Cognito を使用するための Login with Amazon を設定します。詳細については、Login with Amazon のよくある質問の「Login with Amazon のセットアップ」を参照してください。

> **Note**
>
> Login with Amazon を Xamarin アプリケーションに統合するには、Xamarin 入門ガイドに従ってください。

> **Note**
>
> Login with Amazon を Unity プラットフォームでネイティブに統合することはできません。代わりに、ウェブビューを使用して、ブラウザのサインインフローに従います。

Login with Amazon のセットアップ

Login with Amazon の実装

Amazon デベロッパーポータルでは、OAuth アプリケーションをセットアップして ID プールと統合し、Login with Amazon のドキュメントを検索し、SDK をダウンロードできます。[Developer console] (デベロッパーコンソール) を選択し、デベロッパーポータルで Login with Amazon を選択します。アプリケーションのセキュリティプロファイルを作成し、Login with Amazon 認証メカニズムをアプリにビルドできます。Login with Amazon 認証をアプリに統合する方法の詳細については、「認証情報の取得」を参照してください。

Amazon は、新しいセキュリティプロファイルに対して OAuth 2.0 クライアント ID を発行します。クライアント ID は、セキュリティプロファイルの [Web Settings] (ウェブ設定) タブにあります。アイデンティティプールの Login with Amazon IdP の [アプリケーション ID] フィールドにセキュリティプロファイル ID を入力します。

> **Note**
>
> アイデンティティプールの Login with Amazon IdP の [アプリケーション ID] フィールドにセキュリティプロファイル ID を入力します。これは、クライアント ID を使用するユーザープールとは異なります。

Amazon Cognito コンソールで外部プロバイダーを設定する

Amazon アイデンティティプロバイダー (IdP) を使ってログインを追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Login with Amazon] を選択します。
5. Login with Amazon で作成した OAuth プロジェクトのアプリ ID を入力します。詳細については、「Login with Amazon のドキュメント」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。
   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。
     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。
     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。
   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。
   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。
8. [変更を保存] を選択します。

Login with Amazon の使用: Android

Amazon ログインを認証したら、 TokenListener インターフェイスの onSuccess メソッドでトークンを Amazon Cognito 認証情報プロバイダーに渡すことができます。コードは次のようになります。

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString(AuthzConstants.BUNDLE_KEY.TOKEN.val);
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("www.amazon.com", token);
    credentialsProvider.setLogins(logins);
}
```

Login with Amazon の使用: iOS - Objective-C

Amazon ログインを認証したら、AMZN の requestDidSucceed メソッドでトークンを Amazon Cognito 認証情報プロバイダーに渡すことができますAccessTokenDelegate。

```
- (void)requestDidSucceed:(APIResult \*)apiResult {
    if (apiResult.api == kAPIAuthorizeUser) {
        [AIMobileLib getAccessTokenForScopes:[NSArray arrayWithObject:@"profile"]
 withOverrideParams:nil delegate:self];
    }
    else if (apiResult.api == kAPIGetAccessToken) {
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyLoginWithAmazon):
 apiResult.result };
    }
}}
```

Login with Amazon の使用: iOS - Swift

Amazon のログインを認証したら、`AMZNAccessTokenDelegate` の `requestDidSucceed` メソッドで Amazon Cognito 認証情報プロバイダーにトークンを渡すことができます。

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil,
 delegate: self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins =
 [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue: apiResult.result]
    }
```

```
}
```

Login with Amazon の使用： JavaScript

ユーザーが Login with Amazon と認証し、ウェブサイトにリダイレクトされると、Login with Amazon access_token がクエリ文字列で提供されます。このトークンを認証情報ログインマップに渡します。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   Logins: {
       'www.amazon.com': 'Amazon Access Token'
   }
});
```

Login with Amazon の使用: Xamarin

Xamarin for Android

```
AmazonAuthorizationManager manager = new AmazonAuthorizationManager(this,
 Bundle.Empty);

var tokenListener = new APIListener {
  Success = response => {
    // Get the auth token
    var token = response.GetString(AuthzConstants.BUNDLE_KEY.Token.Val);
    credentials.AddLogin("www.amazon.com", token);
  }
};

// Try and get existing login
manager.GetToken(new[] {
  "profile"
}, tokenListener);
```

Xamarin for iOS

`AppDelegate.cs` で、次のコードを挿入します。

```
public override bool OpenUrl (UIApplication application, NSUrl url, string
 sourceApplication, NSObject annotation)
{
```

```
    // Pass on the url to the SDK to parse authorization code from the url
    bool isValidRedirectSignInURL = AIMobileLib.HandleOpenUrl (url, sourceApplication);
    if(!isValidRedirectSignInURL)
        return false;

    // App may also want to handle url
    return true;
}
```

次に、ViewController.cs で、以下の作業を行います。

```
public override void ViewDidLoad ()
{
    base.LoadView ();

    // Here we create the Amazon Login Button
    btnLogin = UIButton.FromType (UIButtonType.RoundedRect);
    btnLogin.Frame = new RectangleF (55, 206, 209, 48);
    btnLogin.SetTitle ("Login using Amazon", UIControlState.Normal);
    btnLogin.TouchUpInside += (sender, e) => {
        AIMobileLib.AuthorizeUser (new [] { "profile"}, new AMZNAuthorizationDelegate
 ());
    };
    View.AddSubview (btnLogin);
}

// Class that handles Authentication Success/Failure
public class AMZNAuthorizationDelegate : AIAuthenticationDelegate
{
  public override void RequestDidSucceed(ApiResult apiResult)
    {
      // Your code after the user authorizes application for requested scopes
      var token = apiResult["access_token"];
      credentials.AddLogin("www.amazon.com",token);
    }

    public override void RequestDidFail(ApiError errorResponse)
    {
      // Your code when the authorization fails
      InvokeOnMainThread(() => new UIAlertView("User Authorization Failed",
 errorResponse.Error.Message, null, "Ok", null).Show());
    }
}
```

Google を ID プール IdP としてセットアップする

Amazon Cognito は Google と統合して、モバイルアプリケーションユーザーにフェデレーションされた認証を提供します。このセクションでは、ID プロバイダーとして Google に対してアプリケーションを登録し、セットアップする方法について説明します。

Android

> **Note**
>
> アプリが Google を使用していて、複数のモバイルプラットフォームで利用可能になる場合は、OpenID Connect プロバイダーとして設定する必要があります。作成されたすべてのクライアント ID を追加オーディエンス値として追加することで、統合が向上します。Google のクライアント間の ID モデルの詳細については、「クライアント間の ID」を参照してください。

Google のセットアップ

Android 用 Google サインインを有効化するには、アプリケーション用の Google デベロッパーコンソールプロジェクトを作成する必要があります。

1. Google 開発者コンソールに移動して、新しいプロジェクトを作成します。
2. [APIs & Services] (API とサービス) を選択し、次に [OAuth consent screen] (OAuth 同意画面) を選択します。Google がプロフィールデータをアプリと共有するための同意を求められたときに、Google がユーザーに表示する情報をカスタマイズします。
3. [Credentials] (認証情報) を選択し、次に[Create credentials] (認証情報の作成) を選択します。[OAuth client ID] (OAuth クライアント ID) を選択します。アプリケーションタイプとして [Android] を選択します。アプリを開発するプラットフォームごとに個別のクライアント ID を作成します。
4. [Credentials] (認証情報) で、[Manage service accounts] (サービスアカウントの管理) を選択します。[Create service account] (サービスアカウントの作成) を選択します。サービスアカウントの詳細を入力し、[Create and continue] (作成して続ける) を選択します。
5. サービスアカウントにプロジェクトへのアクセス権を付与します。アプリが必要とするサービスアカウントへのアクセス権をユーザーに付与します。
6. 新しいサービスアカウントを選択し、[Keys] (キー)、および [Add key] (キーを追加する) を選択します。新しい JSON キーを作成してダウンロードします。

Google デベロッパーコンソールの使用方法の詳細については、Google Cloud ドキュメントの「プロジェクトの作成と管理」を参照してください。

Google を Android アプリに統合する方法の詳細については、Google Identity ドキュメントの「Google でサインインしてユーザーを認証する」を参照してください。

Google ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Google] を選択します。
5. Google Cloud プラットフォームで作成した OAuth プロジェクトのクライアント ID を入力します。詳細については、Google Cloud Platform コンソールヘルプの「OAuth 2.0 の設定」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。

     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

Google の使用

アプリケーションで Google を使用したログインを有効にするには、Android に関する Google のドキュメントに従います。ユーザーがサインインすると、Google に OpenID Connect 認証トークンを要求します。Amazon Cognito はこのトークンを使用してユーザーを認証し、一意の ID を生成します。

次のサンプルコードは、Google Play サービスから認証トークンを取得する方法を示しています。

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
        "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

> **Note**
>
> アプリが Google を使用していて、複数のモバイルプラットフォームで利用可能になる場合は、Google を OpenID Connect プロバイダーとします。作成されたすべてのクライアント ID を追加オーディエンス値として追加することで、統合が向上します。Google のクライアント間の ID モデルの詳細については、「クライアント間の ID」を参照してください。

Google のセットアップ

iOS 用 Google サインインを有効にするには、アプリケーション用の Google デベロッパーコンソールプロジェクトを作成します。

1. Google 開発者コンソールに移動して、新しいプロジェクトを作成します。
2. [APIs & Services] (API とサービス) を選択し、次に [OAuth consent screen] (OAuth 同意画面) を選択します。Google がプロフィールデータをアプリと共有するための同意を求められたときに、Google がユーザーに表示する情報をカスタマイズします。

3. [Credentials] (認証情報) を選択し、次に[Create credentials] (認証情報の作成) を選択します。[OAuth client ID] (OAuth クライアント ID) を選択します。アプリケーションタイプとして [iOS] を選択します。アプリを開発するプラットフォームごとに個別のクライアント ID を作成します。
4. [Credentials] (認証情報) で、[Manage service accounts] (サービスアカウントの管理) を選択します。[Create service account] (サービスアカウントの作成) を選択します。サービスアカウントの詳細を入力し、[Create and continue] (作成して続ける) を選択します。
5. サービスアカウントにプロジェクトへのアクセス権を付与します。アプリが必要とするサービスアカウントへのアクセス権をユーザーに付与します。
6. 新しいサービスアカウントを選択します。[Keys] (キー) タブを選択し、[Add key] (キーを追加する) を選択します。新しい JSON キーを作成してダウンロードします。

Google デベロッパーコンソールの使用方法の詳細については、Google Cloud ドキュメントの「プロジェクトの作成と管理」を参照してください。

iOS アプリへの Google の統合に関する詳細については、Google ID に関するドキュメントの「iOS 用 Google サインイン」を参照してください。

Google ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Google] を選択します。
5. Google Cloud プラットフォームで作成した OAuth プロジェクトのクライアント ID を入力します。詳細については、Google Cloud Platform コンソールヘルプの「OAuth 2.0 の設定」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。

     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール

割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

Google の使用

アプリケーションで Google を使用したログインを有効にするには、iOS に関する Google のドキュメントに従います。認証に成功すると、OpenID Connect 認証トークンが取得されます。Amazon Cognito はこのトークンを使用してユーザーを認証し、固有識別子を生成します。

認証に成功すると、`id_token` が含まれる `GTMOAuth2Authentication` オブジェクトが取得されます。Amazon Cognito はこのオブジェクトを使用してユーザーを認証し、固有識別子を生成します。

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> アプリが Google を使用していて、複数のモバイルプラットフォームで利用可能になる場合は、Google を OpenID Connect プロバイダーとします。作成されたすべてのクライアント

ID を追加オーディエンス値として追加することで、統合が向上します。Google のクライアント間の ID モデルの詳細については、「クライアント間の ID」を参照してください。

Google のセットアップ

iOS 用 Google サインインを有効にするには、アプリケーション用の Google デベロッパーコンソールプロジェクトを作成します。

1. Google 開発者コンソールに移動して、新しいプロジェクトを作成します。
2. [APIs & Services] (API とサービス) を選択し、次に [OAuth consent screen] (OAuth 同意画面) を選択します。Google がプロフィールデータをアプリと共有するための同意を求められたときに、Google がユーザーに表示する情報をカスタマイズします。
3. [Credentials] (認証情報) を選択し、次に[Create credentials] (認証情報の作成) を選択します。[OAuth client ID] (OAuth クライアント ID) を選択します。アプリケーションタイプとして [iOS] を選択します。アプリを開発するプラットフォームごとに個別のクライアント ID を作成します。
4. [Credentials] (認証情報) で、[Manage service accounts] (サービスアカウントの管理) を選択します。[Create service account] (サービスアカウントの作成) を選択します。サービスアカウントの詳細を入力し、[Create and continue] (作成して続ける) を選択します。
5. サービスアカウントにプロジェクトへのアクセス権を付与します。アプリが必要とするサービスアカウントへのアクセス権をユーザーに付与します。
6. 新しいサービスアカウントを選択し、[Keys] (キー)、および [Add key] (キーを追加する) を選択します。新しい JSON キーを作成してダウンロードします。

Google デベロッパーコンソールの使用方法の詳細については、Google Cloud ドキュメントの「プロジェクトの作成と管理」を参照してください。

iOS アプリへの Google の統合に関する詳細については、Google ID に関するドキュメントの「iOS 用 Google サインイン」を参照してください。

Amazon Cognito コンソールのホームページで [Manage Identity Pools] (ID プールの管理) をクリックします。

Amazon Cognito コンソールで外部プロバイダーを設定する

1. 外部プロバイダーとして Google を有効にする ID プールの名前を選択します。アイデンティティプールの [Dashboard] (ダッシュボード) ページが表示されます。

2. [ダッシュボード] ページの右上にある、[ID プールの編集] を選択します。[Edit identity pool] (ID プールの編集) ページが表示されます。
3. 下にスクロールし、[Authentication providers] (認証プロバイダー) を選択してセクションを展開します。
4. [Google] タブを選択します。
5. [ロック解除] を選択します。
6. Google から入手した Google Client ID を入力し、[Save Changes] (変更の保存) を選択します。

Google の使用

アプリケーションで Google を使用したログインを有効にするには、iOS に関する Google のドキュメントに従います。認証に成功すると、OpenID Connect 認証トークンが取得されます。Amazon Cognito はこのトークンを使用してユーザーを認証し、一意の識別子を生成します。

認証が成功すると、`id_token` を含む `GTMOAuth2Authentication` オブジェクトが生成されます。Amazon Cognito はこのトークンを使用してユーザーを認証し、一意の ID を生成します。

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue:
 idToken!]
    }
}
```

JavaScript

> **Note**
>
> アプリが Google を使用していて、複数のモバイルプラットフォームで利用可能になる場合は、Google を OpenID Connect プロバイダーとして設定する必要があります。作成されたすべてのクライアント ID を追加オーディエンス値として追加することで、統合が向上します。Google のクライアント間の ID モデルの詳細については、「クライアント間の ID」を参照してください。

Google のセットアップ

JavaScript ウェブアプリケーションの Google サインインを有効にするには、アプリケーション用の Google Developers コンソールプロジェクトを作成します。

1. Google 開発者コンソールに移動して、新しいプロジェクトを作成します。
2. [APIs & Services] (API とサービス) を選択し、次に [OAuth consent screen] (OAuth 同意画面) を選択します。Google がプロフィールデータをアプリと共有するための同意を求められたときに、Google がユーザーに表示する情報をカスタマイズします。
3. [Credentials] (認証情報) を選択し、次に[Create credentials] (認証情報の作成) を選択します。[OAuth client ID] (OAuth クライアント ID) を選択します。アプリケーションタイプとして [Web application] (ウェブアプリケーション) を選択します。アプリを開発するプラットフォームごとに個別のクライアント ID を作成します。
4. [Credentials] (認証情報) で、[Manage service accounts] (サービスアカウントの管理) を選択します。[Create service account] (サービスアカウントの作成) を選択します。サービスアカウントの詳細を入力し、[Create and continue] (作成して続ける) を選択します。
5. サービスアカウントにプロジェクトへのアクセス権を付与します。アプリが必要とするサービスアカウントへのアクセス権をユーザーに付与します。
6. 新しいサービスアカウントを選択し、[Keys] (キー)、および [Add key] (キーを追加する) を選択します。新しい JSON キーを作成してダウンロードします。

Google デベロッパーコンソールの使用方法の詳細については、Google Cloud ドキュメントの「プロジェクトの作成と管理」を参照してください。

Google をウェブアプリに統合する方法の詳細については、Google ID に関するドキュメントの「Google でサインインする」を参照してください。

Amazon Cognito コンソールで外部プロバイダーを設定する

Google ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Google] を選択します。

5. Google Cloud プラットフォームで作成した OAuth プロジェクトのクライアント ID を入力します。詳細については、Google Cloud Platform コンソールヘルプの「OAuth 2.0 の設定」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。
   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。
     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。
     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。
   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。
   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。
8. [変更を保存] を選択します。

Google の使用

アプリケーションで Google を使用したログインを有効にするには、「ウェブに関する Google のドキュメント」に従います。

認証に成功すると、`id_token` が含まれるレスポンスオブジェクトが取得されます。Amazon Cognito はこのオブジェクトを使用してユーザーを認証し、固有識別子を生成します。

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Add the Google access token to the Amazon Cognito credentials login map.
```

```
      AWS.config.credentials = new AWS.CognitoIdentityCredentials({
         IdentityPoolId: 'IDENTITY_POOL_ID',
         Logins: {
            'accounts.google.com': authResult['id_token']
         }
      });

      // Obtain AWS credentials
      AWS.config.credentials.get(function(){
         // Access AWS resources here.
      });
   }
}
```

Unity

Google のセットアップ

Unity アプリ用 Google サインインを有効にするには、アプリケーション用の Google デベロッパーコンソールプロジェクトを作成します。

1. Google 開発者コンソールに移動して、新しいプロジェクトを作成します。
2. [APIs & Services] (API とサービス) を選択し、次に [OAuth consent screen] (OAuth 同意画面) を選択します。Google がプロフィールデータをアプリと共有するための同意を求められたときに、Google がユーザーに表示する情報をカスタマイズします。
3. [Credentials] (認証情報) を選択し、次に[Create credentials] (認証情報の作成) を選択します。[OAuth client ID] (OAuth クライアント ID) を選択します。アプリケーションタイプとして [Web application] (ウェブアプリケーション) を選択します。アプリを開発するプラットフォームごとに個別のクライアント ID を作成します。
4. Unity の場合は、Android 用に追加の OAuth クライアント ID を作成し、iOS 用には別の OAuth クライアント ID を作成します。
5. [Credentials] (認証情報) で、[Manage service accounts] (サービスアカウントの管理) を選択します。[Create service account] (サービスアカウントの作成) を選択します。サービスアカウントの詳細を入力し、[Create and continue] (作成して続ける) を選択します。
6. サービスアカウントにプロジェクトへのアクセス権を付与します。アプリが必要とするサービスアカウントへのアクセス権をユーザーに付与します。
7. 新しいサービスアカウントを選択し、[Keys] (キー)、および [Add key] (キーを追加する) を選択します。新しい JSON キーを作成してダウンロードします。

Google デベロッパーコンソールの使用方法の詳細については、Google Cloud ドキュメントの「プロジェクトの作成と管理」を参照してください。

IAM コンソールでの OpenID プロバイダーの作成

1. IAM コンソールでの OpenID プロバイダーの作成 OpenID プロバイダーをセットアップする方法の詳細については、「OpenID Connect ID プロバイダーの使用」を参照してください。
2. プロバイダー URL の入力を求められたら、「`"https://accounts.google.com"`」と入力します。
3. [Audience (対象者)] フィールドに値を入力するよう求められたら、前のステップで作成した 3 つのクライアント ID のいずれかを入力します。
4. プロバイダー名を選択し、2 名の対象ユーザーを追加して、他の 2 つのクライアント ID を追加します。

Amazon Cognito コンソールで外部プロバイダーを設定する

Amazon Cognito コンソールのホームページで [Manage Identity Pools] (ID プールの管理) をクリックします。

Google ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Google] を選択します。
5. Google Cloud プラットフォームで作成した OAuth プロジェクトのクライアント ID を入力します。詳細については、Google Cloud Platform コンソールヘルプの「OAuth 2.0 の設定」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。

     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール

割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

Unity Google プラグインのインストール

1. Unity プロジェクトに、Google Play Games plugin for Unity を追加します。
2. Unity で、[Windows] メニューから Android と iOS プラットフォーム用の 3 つの ID を使用してプラグインを設定します。

Google の使用

次のサンプルコードは、Google Play サービスから認証トークンを取得する方法を示しています。

```
void Start()
{
  PlayGamesClientConfiguration config = new
 PlayGamesClientConfiguration.Builder().Build();
  PlayGamesPlatform.InitializeInstance(config);
  PlayGamesPlatform.DebugLogEnabled = true;
  PlayGamesPlatform.Activate();
  Social.localUser.Authenticate(GoogleLoginCallback);
}

void GoogleLoginCallback(bool success)
{
  if (success)
```

```
  {
    string token = PlayGamesPlatform.Instance.GetIdToken();
    credentials.AddLogin("accounts.google.com", token);
  }
  else
  {
    Debug.LogError("Google login failed. If you are not running in an actual Android/
iOS device, this is expected.");
  }
}
```

Xamarin

> **Note**
>
> Amazon Cognito は、Xamarin プラットフォームで Google をネイティブにサポートしていません。現在、この統合では、ウェブビューを使用してブラウザのサインインの流れに従う必要があります。他の SDK での Google の統合の詳細については、他のプラットフォームを選択してください。

アプリケーションで Google を使用したログインを有効にするには、ユーザーを認証し、ユーザーから OpenID Connect トークンを取得します。Amazon Cognito は、Amazon Cognito アイデンティティに関連付けられた一意のユーザー識別子を生成するためにこのトークンを使用します。残念ながら、Xamarin 用の Google SDK では OpenID Connect トークンを取得することができないため、代替クライアントまたはウェブビューのウェブフローを使用します。

トークンを取得したら、`CognitoAWSCredentials` で設定できます。

```
credentials.AddLogin("accounts.google.com", token);
```

> **Note**
>
> アプリが Google を使用していて、複数のモバイルプラットフォームで利用可能になる場合は、Google を OpenID Connect プロバイダーとして設定する必要があります。作成されたすべてのクライアント ID を追加オーディエンス値として追加することで、統合が向上します。Google のクライアント間の ID モデルの詳細については、「クライアント間の ID」を参照してください。

アイデンティティプール IdP として「Apple でサインイン」を設定する

Amazon Cognito は「Apple でサインイン」と統合して、モバイルアプリケーションとウェブアプリケーションのユーザーにフェデレートされた認証を提供します。このセクションでは、ID プロバイダー (IdP) として「Apple でサインイン」を使用してアプリケーションを登録し、セットアップする方法について説明します。

認証プロバイダーとして「Apple でサインイン」を ID プールに追加するには、2 つのステップが必要です。まず、アプリケーションで「Apple でサインイン」を統合し、次に ID プールで「Apple でサインイン」を設定します。「Apple でサインイン」の設定 up-to-date の詳細については、Apple デベロッパードキュメントの「Apple でサインインするための環境の設定」を参照してください。 https://developer.apple.com/documentation/sign_in_with_apple/configuring_your_environment_for_sign_in_with_apple

「Apple でサインイン」を設定する

IdP として「Apple でサインイン」を設定するには、クライアント ID を受け取るためにアプリケーションを Apple に登録する必要があります。

1. Apple の開発者アカウントを作成します。
2. Apple 認証情報を使用してサインインします。
3. 左のナビゲーションペインで、[Certificates, IDs & Profiles (証明書、ID & プロファイル)] を選択します。
4. 左のナビゲーションペインで、[Identifiers (識別子)] を選択します。
5. [Identifiers (識別子)] ページで、[+] アイコンを選択します。
6. [Register a New Identifier] (新しい識別子の登録) ページで、[App IDs] (アプリ ID)、[Continue] (続行) の順に選択します。
7. [Register an App ID] ページで、次の操作を行います。

    a. [説明] に、説明を入力します。

    b. [Bundle ID (バンドル ID)] に、識別子を入力します。このバンドル ID を書き留めておきます。Apple を ID プールのプロバイダーとして設定するには、この値が必要になります。

    c. [Capabilities] (機能) で、[Sign In with Apple] (Apple でサインイン) を選択してから [Edit] (編集) を選択します。

    d. [Sign in with Apple: App ID Configuration] (Apple でサインイン: アプリ ID の設定) ページで、アプリの適切な設定を選択します。次に、[Save] (保存) を選択します。

    e. [Continue] を選択します。

8. [Confirm your App ID] ページで、[登録] を選択します。
9. 「Apple でサインイン」をネイティブ iOS アプリケーションと統合する場合は、ステップ 10 に進みます。ステップ 11 は、「Apple JS でサインイン」と統合するアプリケーション用です。
10. [ID] ページで、[App IDs] (アプリケーション ID) メニューを選択し、次に [Services IDs] (サービス ID) を選択します。[+] アイコンを選択します。
11. [Register a New Identifier] (新しい識別子の登録) ページで、[Services IDs] (サービス ID)、[Continue] (続行) の順に選択します。
12. [Register a Services ID] ページで、次の操作を行います。

    a. [説明] に、説明を入力します。
    b. [Identifier] に、識別子を入力します。このサービス ID を書き留めておきます。Apple を ID プールのプロバイダーとして設定するには、この値が必要になります。
    c. [Apple でサインイン] を選択後、[Configure (設定)] を選択します。
    d. [Web Authentication Configuration] ページで、[Primary App ID] を選択します。[Website URLs] (ウェブサイトの URL) で、[+] アイコンを選択します。[Domains and Subdomains (ドメインとサブドメイン)] で、アプリのドメイン名を入力します。[Return URLs] (リターン URL) に、ユーザーが Sign in with Apple を介して認証した後、認証によってユーザーがリダイレクトされるコールバック URL を入力します。
    e. [次へ] をクリックします。
    f. [Continue] (続行) を選択し、[Register] (登録) を選択します。
13. 左のナビゲーションペインで [Keys (キー)] を選択します。
14. [Keys (キー)] ページで、[+] アイコンを選択します。
15. [Register a New Key] ページで、次の操作を行います。

    a. [Key Name] に、キー名を入力します。
    b. [Sign In with Apple] (Apple でサインイン) を選択後、[Configure] (設定) を選択します。
    c. [Configure Key (キーの設定)] ページで、[Primary App ID (プライマリアプリ ID)]、[Save (保存)] の順に選択します。
    d. [Continue] (続行) を選択し、[Register] (登録) を選択します。

> **Note**
>
> Apple でサインインをネイティブ iOS アプリケーションに統合するには、「Apple でサインインしてユーザー認証を実装する」を参照してください。

ネイティブ iOS 以外のプラットフォームで「Apple でサインイン」を統合する方法については、「Apple JS でサインイン」を参照してください。

Amazon Cognito フェデレーティッド ID コンソールで外部プロバイダーを設定する

以下の手順を使用して、外部プロバイダーを設定します。

Apple ID プロバイダー (IdP) を使ってサインインを追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [Apple でサインイン] を選択します。
5. Apple Developer で作成した OAuth プロジェクトのサービス ID を入力します。詳細については、Apple でサインインのドキュメントの「Apple でサインインを使用してユーザーを認証する」を参照してください。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。
   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。
     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。
     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。
   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。
   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

Amazon Cognito フェデレーティッド ID で Apple をプロバイダーとして Apple でサインインするための CLI 例

この例では、IdP として Apple でサインインする MyIdentityPool という名前の ID プールを作成します。

```
aws cognito-identity create-identity-pool --identity-
pool-name MyIdentityPool --supported-login-providers
appleid.apple.com="sameple.apple.clientid"
```

詳細については、「ID プールの作成」を参照してください。

Amazon Cognito アイデンティティ ID を生成する

この例では、Amazon Cognito ID を生成 (または取得) します。これはパブリック API なので、この API を呼び出すために認証情報は必要ありません。

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --
logins appleid.apple.com="SignInWithAppleIdToken"
```

詳細については、「get-id」を参照してください。

Amazon Cognito アイデンティティ ID の認証情報を取得する

この例では、指定された ID および 「Apple でサインイン」のログインの認証情報を返します。これはパブリック API なので、この API を呼び出すために認証情報は必要ありません。

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

詳細については、「」を参照してください。 get-credentials-for-identity

Apple でサインインを使用する: Android

Apple では、Android 版の 「Apple でサインイン」をサポートする SDK を提供していません。代わりにウェブビューでウェブフローを使用できます。

- アプリケーションで「Apple でサインイン」を設定するには、Apple のドキュメントの「「Apple でサインイン」するためのウェブページの設定」を参照してください。
- Android ユーザーインターフェイスに [Apple でサインイン] ボタンを追加するには、Apple のドキュメントの「「Apple でサインイン」ボタンの表示と設定」に従ってください。
- 「Apple でサインイン」を使用してユーザーを安全に認証するには、Apple のドキュメントの「Apple でサインインするためのウェブページの設定」を参照してください。

「Apple でサインイン」は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトの ID トークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じてユーザーに他の AWS リソースへのアクセスを許可します。

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

「Apple でサインインを使用する」: iOS-Objective-C

Apple は、ネイティブ iOS アプリケーションでの「Apple でサインイン」の SDK サポートを提供しました。ネイティブ iOS デバイスで「Apple でサインイン」を使用してユーザー認証を実装するには、Apple のドキュメントで「「Apple でサインイン」を使用したユーザー認証の実装」を参照してください。

Amazon Cognito は ID トークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じて他の AWS リソースへのアクセス権をユーザーに付与します。

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *)
 error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

「Apple でサインイン」: iOS - Swift

Apple は、ネイティブ iOS アプリケーションでの「Apple でサインイン」の SDK サポートを提供しました。ネイティブ iOS デバイスで「Apple でサインイン」を使用してユーザー認証を実装するには、Apple のドキュメントで「「Apple でサインイン」を使用したユーザー認証の実装」を参照してください。

Amazon Cognito は ID トークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じて他の AWS リソースへのアクセス権をユーザーに付与します。

iOS で「Apple でサインイン」を設定する方法についての詳細は、「Apple でサインインを設定する」を参照してください。

```
func finishedWithAuth(auth: ASAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.identityToken,
      credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
}
```

「Apple でサインイン」を使用します。 JavaScript

Apple は、 の「Apple でサインイン」をサポートする SDK を提供していません JavaScript。代わりにウェブビューでウェブフローを使用できます。

- アプリケーションで「Apple でサインイン」を設定するには、Apple のドキュメントの「「Apple でサインイン」するためのウェブページの設定」を参照してください。
- JavaScript ユーザーインターフェイスに「Apple でサインイン」ボタンを追加するには、Apple ドキュメントの「Apple でサインインボタンの表示と設定」を参照してください。
- 「Apple でサインイン」を介してユーザーを安全に認証するには、Apple のドキュメントの「「Apple でサインイン」するためのウェブページの設定」を参照してください。

「Apple でサインイン」は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトの ID トークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じてユーザーに他の AWS リソースへのアクセスを許可します。

```
function signinCallback(authResult) {
     // Add the apple's id token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'appleid.apple.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
}
```

「Apple でサインイン」を使用する: Xamarin

Xamarin 用の「Apple でサインイン」をサポートする SDK はありません。代わりにウェブビューでウェブフローを使用できます。

- アプリケーションで「Apple でサインイン」を設定するには、Apple のドキュメントの「「Apple でサインイン」するためのウェブページの設定」を参照してください。
- Xamarin ユーザーインターフェイスに [Apple でサインイン] ボタンを追加するには、Apple のドキュメントの「「Apple でサインイン」ボタンの表示と設定」に従ってください。
- 「Apple でサインイン」を介してユーザーを安全に認証するには、Apple のドキュメントの「「Apple でサインイン」するためのウェブページの設定」を参照してください。

「Apple でサインイン」は、その状態の追跡にセッションオブジェクトを使用します。Amazon Cognito は、このセッションオブジェクトの ID トークンを使用してユーザーを認証し、一意の識別子を生成し、必要に応じてユーザーに他の AWS リソースへのアクセスを許可します。

トークンを取得したら、`CognitoAWSCredentials` で設定できます。

```
credentials.AddLogin("appleid.apple.com", token);
```

OIDC プロバイダーを ID プール IdP としてセットアップする

OpenID Connect は認証のためのオープン標準で、多数のログインプロバイダーでサポートされています。Amazon Cognito は、AWS Identity and Access Management を使用して設定した OpenID Connect プロバイダーとアイデンティティをリンクすることをサポートしています。

OpenID Connect プロバイダーの追加

OpenID Connect プロバイダーの作成方法については、AWS Identity and Access Management ユーザーガイドの「OpenID Connect (OIDC) ID プロバイダーの作成」を参照してください。

Amazon Cognito とのプロバイダーの関連付け

OIDC ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. [OpenID Connect (OIDC)] を選択します。
5. の IAM から OIDC ID プロバイダー IdPs を選択します AWS アカウント。新しい SAML プロバイダーを追加する場合は、[新しいプロバイダの作成] を選択して IAM コンソールに移動します。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。
   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。
     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。
     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。
7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

複数の OpenID Connect プロバイダーを、単一の ID プールと関連付けることができます。

OpenID Connect の使用

ログインして ID トークンを受け取る方法については、プロバイダーのドキュメントを参照してください。

トークンを取得したら、ログインマップにトークンを追加します。プロバイダーの URI をキーとして使用します。

OpenID Connect トークンの検証

初めて Amazon Cognito と統合するときは、`InvalidToken` 例外を受け取る場合があります。Amazon Cognito が OpenID Connect (OIDC) トークンを検証する方法を理解しておくことが重要です。

> **Note**
>
> https://tools.ietf.org/html/rfc7523 で規定されているように、Amazon Cognito では、システム間のクロックスキューを処理するために 5 分の猶予期間が提供されます。

1. `iss` パラメータは、ログインマップで使われるキー (login.provider.com など) と一致する必要があります。
2. 署名が有効である必要があります。署名は、RSA パブリックキーを通じて検証可能である必要があります。
3. 証明書の公開キーのフィンガープリントは、OIDC プロバイダーを作成したときに IAM で設定したフィンガープリントと一致します。

4. azp パラメータが存在する場合は、OIDC プロバイダーでリストされたクライアント ID に対してこの値を確認します。
5. azp パラメータが存在しない場合は、OIDC プロバイダーでリストされたクライアント ID に対して aud パラメータを確認します。

ウェブサイトの jwt.io は、これらの値を検証するためのトークンをデコードするための貴重なリソースです。

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

iOS - Swift

Amazon Cognito に OIDC ID トークンを提供するには、AWSIdentityProviderManager プロトコルを実装します。

logins メソッドを実装する際は、設定した OIDC プロバイダー名を含むディクショナリを返します。次のコード例に示すように、このディクショナリはキーとして機能し、認証されたユーザーからの現在の ID トークンが値として機能します。

```
class OIDCProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        let completion = AWSTaskCompletionSource<NSString>()
        getToken(tokenCompletion: completion)
        return completion.task.continueOnSuccessWith { (task) -> AWSTask<NSDictionary>?
 in
            //login.provider.name is the name of the OIDC provider as setup in the
 Amazon Cognito console
            return AWSTask(result:["login.provider.name":task.result!])
        } as! AWSTask<NSDictionary>

    }
```

```
    func getToken(tokenCompletion: AWSTaskCompletionSource<NSString>) -> Void {
        //get a valid oidc token from your server, or if you have one that hasn't
 expired cached, return it

        //TODO code to get token from your server
        //...

        //if error getting token, set error appropriately
        tokenCompletion.set(error:NSError(domain: "OIDC Login", code: -1 , userInfo:
 ["Unable to get OIDC token" : "Details about your error"]))
        //else
        tokenCompletion.set(result:"result from server id token")
    }
}
```

をインスタンス化するときはAWSCognitoCredentialsProvider、 が実装するクラスをコンストラクターの identityProviderManagerの値AWSIdentityProviderManagerとして渡します。詳細については、AWSCognitoCredentialsProviderリファレンスページに移動し、initWithRegionタイプ：identityPoolId：identityProviderManager を選択します。

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

SAML プロバイダーを ID プール IdP としてセットアップする

Amazon Cognito は、Security Assertion Markup Language 2.0 (SAML 2.0IdPs) による ID プロバイダー () による認証をサポートしています。SAML をサポートする IdP を Amazon Cognito で使用し

て、ユーザーにシンプルなオンボーディングフローを提供することができます。SAML サポート IdP は、ユーザーが引き受けることができる IAM ロールを指定します。このようにして、さまざまなユーザーがさまざまな権限のセットを受け取ることができます。

SAML プロバイダーの ID プールの設定

次のステップでは、SAML ベースの IdP が使用されるように ID プールを設定する方法について説明します。

> **Note**
>
> SAML プロバイダーをサポートするように ID プールを設定する前に、まず IAM コンソールで SAML ID プロバイダーを設定します。詳細については、IAM ユーザーガイドの「サードパーティーの SAML ソリューションプロバイダーと AWSの統合」を参照してください。

SAML ID プロバイダー (IdP) を追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。
4. SAML を選択します。
5. の IAM から SAML ID プロバイダー IdPs を選択します AWS アカウント。新しい SAML プロバイダーを追加する場合は、[新しいプロバイダの作成] を選択して IAM コンソールに移動します。
6. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときにリクエストするロールを設定するには、[ロール設定] を設定します。

   - その IdP のユーザーに、認証済みロールを設定したときに設定したデフォルトロールを割り当てることも、ルール付きのロールを選択することもできます。

     i. [ルールを使用してロールを選択する] を選択した場合、ユーザー認証からのソースクレーム、クレームを比較するオペレータ、このロール選択と一致する値、およびロール割り当てが一致したときに割り当てるロールを入力します。別の条件に基づいて追加のルールを作成するには、[別のものを追加] を選択します。

     ii. [ロールの解決] を選択します。ユーザーのクレームがルールに合わない場合は、認証情報を拒否するか、認証済みロールの認証情報を発行できます。

7. Amazon Cognito がこのプロバイダーで認証されたユーザーに認証情報を発行するときに割り当てるプリンシパルタグを変更するには、[アクセスコントロールの属性] を設定します。

   a. プリンシパルタグを適用しない場合は、[非アクティブ] を選択します。

   b. sub および aud クレームに基づいてプリンシパルタグを適用するには、[デフォルトマッピングを使用] を選択します。

   c. プリンシパルタグへの属性の独自のカスタムスキーマを作成するには、[カスタムマッピングを使用] を選択します。次に、タグに表示したい各クレームから取得するタグキーを入力します。

8. [変更を保存] を選択します。

SAML IdP の設定

SAML プロバイダーを作成したら、SAML ID プロバイダーを設定し、IdP と AWSの間に証明書利用者の信頼を追加します。多くの では IdPs、IdP が XML ドキュメントから証明書利用者の情報と証明書を読み取るために使用できる URL を指定できます。には AWS、https://signin.aws.amazon.com/static/saml-metadata.xml を使用できます。次のステップでは、IdP からの SAML アサーションレスポンスを設定して、 AWS に必要なクレームを入力します。クレーム設定の詳細については、「認証レスポンスの SAML アサーションを設定する」を参照してください。

SAML IdP の SAML メタデータに複数の署名証明書が含まれている場合、サインイン時に、SAML アサーションが SAML メタデータ内のいずれかの証明書と一致すると、ユーザープールは SAML アサーションが有効であると判断します。

SAML を使用したユーザーロールのカスタマイズ

Amazon Cognito ID で SAML を使用する場合、エンドユーザーのロールをカスタマイズできます。Amazon Cognito は、SAML ベースの IdP で拡張フローのみをサポートします。SAML ベースの IdP を使用するために、ID プールに認証されたロールや認証されていないロールを指定する必要はありません。`https://aws.amazon.com/SAML/Attributes/Role` クレーム属性は、カンマ区切りのロールおよびプロバイダー ARN のペアを 1 つ以上指定します。これらは、ユーザーが引き受けることができるロールです。IdP から入手できるユーザー属性情報に基づいてロール属性を設定するように SAML IdP を設定できます。SAML アサーションで複数のロールを受け取る場合は、`getCredentialsForIdentity` を呼び出すときにオプションの `customRoleArn` パラメータを設定します ロールが SAML アサーションのクレームのロールと一致する場合、ユーザーはこの `customRoleArn` を想定します。

SAML IdP を使用したユーザーの認証

SAML ベースの IdP とフェデレートするには、ユーザーが login. AWS federation を開始する URL を決定します。 は IdP によって開始されたログインを使用します。AD FS 2.0 では、URL の形式は `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices` です。

Amazon Cognito に SAML IdP のサポートを追加するには、まず iOS アプリケーションまたは Android アプリケーションから SAML ID プロバイダーを使用してユーザーを認証する必要があります。SAML IdP との統合と認証に使用するコードは、SAML プロバイダーに固有です。ユーザーが認証されたら、Amazon Cognito API を使用して、結果として得られた SAML アサーションを Amazon Cognito アイデンティティに提供できます。

ID プール API リクエストの `Logins` マップで SAML アサーションを繰り返したり、再生したりすることはできません。SAML アサーションを再生すると、そのアサーション ID は以前の API リクエストの ID と重複します。`Logins` マップで SAML アサーションを受け入れることができる API オペレーションにはGetId、、GetCredentialsForIdentity、GetOpenIdToken、および GetOpenID TokenForDeveloperIdentityが含まれます。SAML アサーション ID は、ID プール認証フローの API リクエストごとに 1 回再生できます。例えば、同じ SAML アサーションを `GetId` リクエストと後続の `GetCredentialsForIdentity` リクエストでは指定できますが、2 番目の `GetId` リクエストでは指定できません。

Android

Android SDK を使用する場合、以下のように、ログインマップに SAML アサーションを入力できます。

```
Map logins = new HashMap();
logins.put("arn:aws:iam::aws account id:saml-provider/name", "base64 encoded assertion
 response");
// Now this should be set to CognitoCachingCredentialsProvider object.
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(context, identity pool id, region);
credentialsProvider.setLogins(logins);
// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role
credentialsProvider.setCustomRoleArn("arn:aws:iam::aws account id:role/
customRoleName");
// This should trigger a call to the Amazon Cognito service to get the credentials.
credentialsProvider.getCredentials();
```

iOS

iOS SDK を使用している場合、以下のように、AWSIdentityProviderManager で SAML アサーションを指定できます。

```
- (AWSTask<NSDictionary<NSString*,NSString*> *> *) logins {
    //this is hardcoded for simplicity, normally you would asynchronously go to your
 SAML provider
    //get the assertion and return the logins map using a AWSTaskCompletionSource
    return [AWSTask taskWithResult:@{@"arn:aws:iam::aws account id:saml-provider/
name":@"base64 encoded assertion response"}];
}

// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role.
// Implementing this is optional if there is only one role.
- (NSString *)customRoleArn {
    return @"arn:aws:iam::accountId:role/customRoleName";
}
```

デベロッパーが認証したアイデンティティ (アイデンティティプール)

Amazon Cognito は、Facebook を ID プール IdP としてセットアップする、Google を ID プール IdP としてセットアップする、Login with Amazon を ID プール IdP としてセットアップする、および アイデンティティプール IdP として「Apple でサインイン」を設定する 経由でのウェブ ID フェデレーションに加えて、デベロッパーが認証したアイデンティティもサポートします。デベロッパーが認証した ID では、Amazon Cognito を使用してユーザーデータを同期し、 AWS リソースにアクセスしながら、独自の既存の認証プロセスを通じてユーザーを登録および認証できます。デベロッパーが認証したアイデンティティの使用には、エンドユーザーのデバイス、認証のバックエンド、および Amazon Cognito 間の対話が関連します。詳細については、 AWS ブログの「Understanding Amazon Cognito Authentication Part 2: Developer Authenticated ID」を参照してください。

認証のフローについて

GetOpenIdTokenForDeveloperIdentity API オペレーションは、拡張認証と基本認証の両方に対してデベロッパー認証を開始できます。この API は、管理者認証情報を使用してリクエストを認証します。Logins マップは、カスタム識別子とlogin.mydevproviderペアになった のような ID プールデベロッパープロバイダー名です。

例：

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

拡張認証

からのトークンの名前cognito-identity.amazonaws.comと値を含むLoginsマップを使用して GetCredentialsForIdentity API オペレーションを呼び出しますGetOpenIdTokenForDeveloperIdentity。

例：

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

GetCredentialsForIdentity デベロッパーが認証した ID を持つ は、ID プールのデフォルトの認証されたロールの一時的な認証情報を返します。

基本認証

AssumeRoleWithWebIdentity API オペレーションを呼び出し、適切な信頼関係が定義されている IAM ロールRoleArnの をリクエストします。 ???の値を から取得したトークンWebIdentityTokenに設定しますGetOpenIdTokenForDeveloperIdentity。

デベロッパーが認証した ID 認証フローと、外部プロバイダー ID との違いについては、「」を参照してくださいID プール (フェデレーティッドアイデンティティ) の認証フロー。

デベロッパー名を指定して ID プールに関連付ける

デベロッパーが認証したアイデンティティを使用するには、デベロッパープロバイダーに関連付けられたアイデンティティプールが必要です。そのためには、以下の手順を実行します。

カスタムデベロッパープロバイダーを追加するには

1. Amazon Cognito コンソールで [ID プールの管理] をクリックします。アイデンティティプールを選択します。
2. [ユーザーアクセス] タブを選択します。
3. [ID プロバイダーを追加] を選択します。

4. カスタムデベロッパープロバイダーを選択します。
5. デベロッパープロバイダー名を入力します。デベロッパープロバイダーを追加した後は、変更または削除することはできません。
6. [変更を保存] を選択します。

注意: プロバイダー名を設定した場合、それを変更することはできません。

Amazon Cognito コンソールでの作業に関する追加の手順については、「Amazon Cognito コンソールの使用」を参照してください。

ID プロバイダーの実装

Android

ディベロッパーが認証した ID を使用するには、AWSAbstractCognitoIdentityProvider を拡張する独自の ID プロバイダークラスを実装します。ID プロバイダークラスは、属性としてトークンを含むレスポンスオブジェクトを返す必要があります。

以下に示しているのは、ID プロバイダーの基本的な例です。

```
public class DeveloperAuthenticationProvider extends
 AWSAbstractCognitoDeveloperIdentityProvider {

  private static final String developerProvider = "<Developer_provider_name>";

  public DeveloperAuthenticationProvider(String accountId, String identityPoolId,
 Regions region) {
    super(accountId, identityPoolId, region);
    // Initialize any other objects needed here.
  }

  // Return the developer provider name which you choose while setting up the
  // identity pool in the &COG; Console

  @Override
  public String getProviderName() {
    return developerProvider;
  }

  // Use the refresh method to communicate with your backend to get an
  // identityId and token.
```

```
  @Override
  public String refresh() {

    // Override the existing token
    setToken(null);

    // Get the identityId and token by making a call to your backend
    // (Call to your backend)

    // Call the update method with updated identityId and token to make sure
    // these are ready to be used from Credentials Provider.

    update(identityId, token);
    return token;

  }

  // If the app has a valid identityId return it, otherwise get a valid
  // identityId from your backend.

  @Override
  public String getIdentityId() {

    // Load the identityId from the cache
    identityId = cachedIdentityId;

    if (identityId == null) {
       // Call to your backend
    } else {
       return identityId;
    }

  }
}
```

この ID プロバイダーを使用するには、これを CognitoCachingCredentialsProvider に渡す必要があります。例を示します。

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.USEAST1);
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

iOS - Objective-C

ディベロッパーが認証した ID を使用するには、AWSCognitoCredentialsProviderHelper を拡張する独自の ID プロバイダークラスを実装します。ID プロバイダークラスは、属性としてトークンを含むレスポンスオブジェクトを返す必要があります。

```
@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */

- (AWSTask <NSString*> *) token {
    //Write code to call your backend:
    //Pass username/password to backend or some sort of token to authenticate user
    //If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins
 map
    //containing "your.provider.name":"enduser.username"
    //Return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = response.identityId;
    return [AWSTask taskWithResult:response.token];
}

@end
```

この ID プロバイダーを使用するには、次の例に示すように、それを AWSCognitoCredentialsProvider に渡します。

```
DeveloperAuthenticatedIdentityProvider * devAuth =
 [[DeveloperAuthenticatedIdentityProvider alloc]
 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                          identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                                         useEnhancedFlow:YES
                                identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc]

 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                                        identityProvider:devAuth];
```

認証されていない ID と開発者が認証した ID の両方をサポートする場合は、AWSCognitoCredentialsProviderHelper の実装で logins メソッドをオーバーライドします。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}
```

開発者が認証した ID とソーシャルプロバイダーをサポートする場合は、AWSCognitoCredentialsProviderHelper の logins 実装で現在のプロバイダーが誰であるかを管理する必要があります。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

iOS - Swift

ディベロッパーが認証した ID を使用するには、AWSCognitoCredentialsProviderHelper を拡張する独自の ID プロバイダークラスを実装します。ID プロバイダークラスは、属性としてトークンを含むレスポンスオブジェクトを返す必要があります。

```
import AWSCore
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSCognitoCredentialsProviderHelper {
    override func token() -> AWSTask<NSString> {
    //Write code to call your backend:
```

```
    //pass username/password to backend or some sort of token to authenticate user, if
 successful,
    //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
 "your.provider.name":"enduser.username"
    //return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = resultFromAbove.identityId
    return AWSTask(result: resultFromAbove.token)
}
```

この ID プロバイダーを使用するには、次の例に示すように、それを AWSCognitoCredentialsProvider に渡します。

```
let devAuth =
 DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
 identityProviderManager:nil)
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
 credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

認証されていない ID と開発者が認証した ID の両方をサポートする場合は、AWSCognitoCredentialsProviderHelper の実装で logins メソッドをオーバーライドします。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
}
```

開発者が認証した ID とソーシャルプロバイダーをサポートする場合は、AWSCognitoCredentialsProviderHelper の logins 実装で現在のプロバイダーが誰であるかを管理する必要があります。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

JavaScript

バックエンドからアイデンティティ ID とセッショントークンを取得したら、それらを `AWS.CognitoIdentityCredentials` プロバイダーに渡します。以下に例を示します。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   IdentityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
   Logins: {
      'cognito-identity.amazonaws.com': 'TOKEN_RETURNED_FROM_YOUR_PROVIDER'
   }
});
```

Unity

開発者が認証した ID を使用するには、`CognitoAWSCredentials` を拡張し、`RefreshIdentity` メソッドを上書きして、ユーザー ID とトークンをバックエンドから取得して返す必要があります。「example.com」という仮定のバックグランドに接続する ID プロバイダーの簡単な例を以下に示します。

```
using UnityEngine;
using System.Collections;
using Amazon.CognitoIdentity;
using System.Collections.Generic;
using ThirdParty.Json.LitJson;
using System;
using System.Threading;
```

```
public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;

    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override IdentityState RefreshIdentity()
    {
        IdentityState state = null;
        ManualResetEvent waitLock = new ManualResetEvent(false);
        MainThreadDispatcher.ExecuteCoroutineOnMainThread(ContactProvider((s) =>
        {
            state = s;
            waitLock.Set();
        }));
        waitLock.WaitOne();
        return state;
    }

    IEnumerator ContactProvider(Action<IdentityState> callback)
    {
        WWW www = new WWW("http://example.com/?username="+login);
        yield return www;
        string response = www.text;

        JsonData json = JsonMapper.ToObject(response);

        //The backend has to send us back an Identity and a OpenID token
        string identityId = json["IdentityId"].ToString();
        string token = json["Token"].ToString();

        IdentityState state = new IdentityState(identityId, PROVIDER_NAME, token,
 false);
        callback(state);
    }
```

```
}
```

上記のコードでは、スレッドディスパッチャーオブジェクトを使用してコルーチンを呼び出します。プロジェクトでこれを行う方法がない場合は、シーンで以下のスクリプトを使用できます。

```
using System;
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class MainThreadDispatcher : MonoBehaviour
{
    static Queue<IEnumerator> _coroutineQueue = new Queue<IEnumerator>();
    static object _lock = new object();

    public void Update()
    {
        while (_coroutineQueue.Count > 0)
        {
            StartCoroutine(_coroutineQueue.Dequeue());
        }
    }

    public static void ExecuteCoroutineOnMainThread(IEnumerator coroutine)
    {
        lock (_lock) {
            _coroutineQueue.Enqueue(coroutine);
        }
    }
}
```

Xamarin

開発者が認証した ID を使用するには、CognitoAWSCredentials を拡張し、RefreshIdentity メソッドを上書きして、ユーザー ID とトークンをバックエンドから取得して返す必要があります。「example.com」という仮定のバックグランドに接続する ID プロバイダーの基本的な例を以下に示します。

```
public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
```

```
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override async Task<IdentityState> RefreshIdentityAsync()
    {
        IdentityState state = null;
        //get your identity and set the state
        return state;
    }
}
```

ログインマップの更新 (Android および iOS のみ)

Android

ご使用の認証システムでユーザーが正常に認証されたら、デベロッパープロバイダー名と開発ユーザー ID を使用してログインマップを更新します。これは、認証システム内のユーザーを一意に識別する英数字の文字列です。`refresh` が変更されている可能性があるため、ログインマップを更新した後は、必ず `identityId` メソッドを呼び出してください。

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
loginsMap.put(developerAuthenticationProvider.getProviderName(),
 developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS - Objective-C

認証情報がないか、または期限切れ場合、iOS SDK は `logins` メソッドを呼び出して最新のログインマップを取得するのみです。SDK で新しい認証情報の入手を強制する場合は (たとえば、エンドユーザーが認証されていない状態から認証され、認証されたユーザーに対して認証情報を必要とするなど)、`clearCredentials` で `credentialsProvider` を呼び出します。

```
[credentialsProvider clearCredentials];
```

iOS - Swift

認証情報がないか、または期限切れ場合、iOS SDK は `logins` メソッドを呼び出して最新のログインマップを取得するのみです。SDK で新しい認証情報の入手を強制する場合は (たとえば、エンドユーザーが認証されていない状態から認証され、認証されたユーザーに対して認証情報を必要とするなど)、`clearCredentials` で `credentialsProvider` を呼び出します。

```
credentialsProvider.clearCredentials()
```

トークンの取得 (サーバー)

トークンを取得するには、 を呼び出しますGetOpenIdTokenForDeveloperIdentity。この API は、AWS 開発者の認証情報を使用してバックエンドから呼び出す必要があります。クライアント SDK から呼び出すことはできません。API は Cognito アイデンティティプール ID (ID プロバイダー名をキーとして、識別子を値として含むログインマップ) と、オプションで Cognito アイデンティティ ID (認証されていないユーザーを認証する場合など) を受け取ります。識別子はユーザーのユーザー名、メールアドレス、または数値の場合があります。API は、ユーザーの一意の Cognito ID と、エンドユーザーの OpenID Connect トークンを使用して呼び出しに応答します。

`GetOpenIdTokenForDeveloperIdentity` によって返されるトークンについては、以下の点に注意してください。

- トークンのカスタムの有効期限を指定して、トークンをキャッシュすることができます。カスタムの有効期限を指定しない場合、トークンは 15 分間有効です。
- 設定できる最大のトークンの期間は 24 時間です。
- トークンの期間を増やすことによるセキュリティへの影響について注意してください。攻撃者がこのトークンを取得した場合、そのトークンをエンドユーザーの AWS 認証情報とトークンの有効期間の間交換できます。

以下の Java スニペットは、Amazon Cognito クライアントを初期化して、デベロッパーが認証したアイデンティティのトークンを取得する方法を示しています。

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
```

```
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client
 has an
                                                    //identity ID that you want to link
 to this
                                                    //developer account

// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

前述の手順に従って、アプリでデベロッパーが認証したアイデンティティを統合できます。問題またはご質問がある場合は、お気軽にフォーラムに投稿してください。

既存のソーシャル ID への接続

デベロッパーが認証したアイデンティティを使用する際のプロバイダーのすべてのリンク作成は、バックエンドから行う必要があります。カスタム ID をユーザーのソーシャル ID (Login with Amazon、Sign in with Apple、Facebook、または Google) に接続するには、 を呼び出すときに ID プロバイダートークンをログインマップに追加しますGetOpenIdTokenForDeveloperIdentity。これを可能にするため、クライアント SDK からバックエンドを呼び出してエンドユーザーを認証するときは、エンドユーザーのソーシャルプロバイダートークンを追加で渡します。

たとえば、カスタム ID の Facebook へのリンクを試みている場
合、GetOpenIdTokenForDeveloperIdentity を呼び出すときに、ID プロバイダー ID に加えて Facebook のトークンをログインマップに追加します。

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

プロバイダー間の移行のサポート

Android

アプリケーションでは、デベロッパーが認証したアイデンティティとともに、パブリックプロバイダー (Login with Amazon、「Apple でサインイン」、Facebook、または Google) を使用した、認証されていないアイデンティティまたは認証されたアイデンティティのサポートが必要になる場合があります。デベロッパーが認証したアイデンティティとその他のアイデンティティ (認証されていないアイデンティティと、パブリックプロバイダーを使用して認証されたアイデンティティ) の重要な違いは、identityId およびトークンの取得方法にあります。他のアイデンティティについては、モバイルアプリケーションが認証システムに接続する代わりに Amazon Cognito と直接やり取りします。そのため、モバイルアプリケーションは、アプリユーザーが行った選択に基づいて、2 つの明確の流れをサポートできる必要があります。そのためには、カスタム ID プロバイダーにいくつか変更を加える必要があります。

refresh メソッドはログインマップをチェックします。マップが空でなくデベロッパープロバイダー名を含むキーがある場合は、バックエンドを呼び出します。それ以外の場合は、 メソッドを getIdentityId呼び出して null を返します。

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {

      /**
       * This is where you would call your backend
       **/
```

```
      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

同様に、getIdentityId メソッドにも、ログインマップの内容に応じて 2 つのフローがあります。

```
public String getIdentityId() {

   // Load the identityId from the cache
   identityId = cachedIdentityId;

   if (identityId == null) {

      // If the logins map is not empty make a call to your backend
      // to get the token and identityId

      if (getProviderName() != null && !this.loginsMap.isEmpty()
         && this.loginsMap.containsKey(getProviderName())) {

         /**
           * This is where you would call your backend
          **/

         // now set the returned identity id and token in the provider
         update(identityId, token);
         return token;

      } else {
         // Otherwise call &COG; using getIdentityId of super class
         return super.getIdentityId();
      }

   } else {
      return identityId;
   }
```

```
}
```

iOS - Objective-C

アプリケーションでは、デベロッパーが認証したアイデンティティとともに、パブリックプロバイダー (Login with Amazon、「Apple でサインイン」、Facebook、または Google) を使用した、認証されていないアイデンティティまたは認証されたアイデンティティのサポートが必要になる場合があります。これを行うには、 AWSCognitoCredentialsProviderHelperloginsメソッドをオーバーライドして、現在の ID プロバイダーに基づいて正しいログインマップを返せるようにします。この例では、非認証、Facebook 認証、デベロッパー認証を切り替える方法を示しています。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

非認証から認証に移行するときは、[credentialsProvider clearCredentials]; を呼び出して、認証された新しい認証情報の取得を SDK に強制します。2 つの認証されたプロバイダーを切り替え、2 つのプロバイダーへのリンクを試みない (ログインディクショナリで複数のプロバイダー用のトークンを提供していない) 場合は、[credentialsProvider clearKeychain]; を呼び出します。これにより、認証情報とアイデンティティの両方がクリアされ、SDK は新しい認証を取得します。

iOS - Swift

アプリケーションでは、デベロッパーが認証したアイデンティティとともに、パブリックプロバイダー (Login with Amazon、「Apple でサインイン」、Facebook、または Google) を使用した、認証されていないアイデンティティまたは認証されたアイデンティティのサポートが必要になる場合があります。これを行うには、 AWSCognitoCredentialsProviderHelperloginsメソッドをオーバーライドして、現在の ID プロバイダーに基づいて正しいログインマップを返せるようにします。この例では、非認証、Facebook 認証、デベロッパー認証を切り替える方法を示しています。

```
override func logins () -> AWSTask<NSDictionary> {
```

```
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

非認証から認証に移行するときは、credentialsProvider.clearCredentials() を呼び出して、認証された新しい認証情報の取得を SDK に強制します。2 つの認証されたプロバイダーを切り替え、2 つのプロバイダーへのリンクを試みない (ログインディクショナリで複数のプロバイダー用のトークンを提供していない) 場合は、credentialsProvider.clearKeychain() を呼び出します。これにより、認証情報とアイデンティティの両方がクリアされ、SDK は新しい認証を取得します。

Unity

アプリケーションでは、デベロッパーが認証したアイデンティティとともに、パブリックプロバイダー (Login with Amazon、「Apple でサインイン」、Facebook、または Google) を使用した、認証されていないアイデンティティまたは認証されたアイデンティティのサポートが必要になる場合があります。デベロッパーが認証したアイデンティティとその他のアイデンティティ (認証されていないアイデンティティと、パブリックプロバイダーを使用して認証されたアイデンティティ) の重要な違いは、identityId およびトークンの取得方法にあります。他のアイデンティティについては、モバイルアプリケーションが認証システムに接続する代わりに Amazon Cognito と直接やり取りします。モバイルアプリケーションは、アプリユーザーが行った選択に基づいて、2 つの明確の流れをサポートできる必要があります。そのためには、カスタム ID プロバイダーにいくつか変更を加える必要があります。

Unity で実行する推奨方法は、ID プロバイダーを AmazonCognitoEnhancedIdentityProvide ではなくから拡張し AbstractCognitoIdentityProvider、ユーザーが独自のバックエンドで認証されない場合に備えて、親 RefreshAsync メソッドを呼び出します。ユーザーが認証されている場合、前に説明したのと同じ流れを使用できます。

Xamarin

アプリケーションでは、デベロッパーが認証したアイデンティティとともに、パブリックプロバイダー (Login with Amazon、「Apple でサインイン」、Facebook、または Google) を使用した、認証されていないアイデンティティまたは認証されたアイデンティティのサポートが必要になる場合があります。デベロッパーが認証したアイデンティティとその他のアイデンティティ (認証されていないアイデンティティと、パブリックプロバイダーを使用して認証されたアイデンティティ) の重要な違いは、identityId およびトークンの取得方法にあります。他のアイデンティティについては、モバイルアプリケーションが認証システムに接続する代わりに Amazon Cognito と直接やり取りします。モバイルアプリケーションは、アプリユーザーが行った選択に基づいて、2 つの明確の流れをサポートできる必要があります。そのためには、カスタム ID プロバイダーにいくつか変更を加える必要があります。

認証されていないユーザーから認証されたユーザー (ID プール) への切り替え

Amazon Cognito ID プールは、認証されたユーザーと認証されていないユーザーの両方をサポートします。認証されていないユーザーは、ID プロバイダー (IdP) を使用してログインしていない場合でも、AWS リソースへのアクセス権を受け取ります。このレベルのアクセスは、ユーザーがログインする前にユーザーにコンテンツを表示する場合に便利です。認証されていない各ユーザーは、個別にログインして認証していない場合でも、ID プールに一意の ID があります。

このセクションでは、ユーザーがログインに使用する ID を、認証されていない ID から認証された ID に切り替える方法について説明します。

Android

ユーザーは、認証されていないゲストとしてアプリケーションにログインできます。最終的に、ユーザーはサポートされている IdP のいずれかを使用してログインすることを決定する可能性があります。Amazon Cognito は、古いアイデンティティが新しいアイデンティティと同じ識別子を維持し、プロファイルデータが自動的にマージされることを確実にします。

アプリケーションに、プロファイルの結合が `IdentityChangedListener` インターフェイスを通じて通知されます。インターフェイスで `identityChanged` メソッドを実装して、これらのメッセージを受け取ります。

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
```

```
    // handle the change
}
```

iOS - Objective-C

ユーザーは、認証されていないゲストとしてアプリケーションにログインできます。最終的に、ユーザーはサポートされている IdP のいずれかを使用してログインすることを決定する可能性があります。Amazon Cognito は、古いアイデンティティが新しいアイデンティティと同じ識別子を維持し、プロファイルデータが自動的にマージされることを確実にします。

NSNotificationCenter はプロファイルの結合をアプリケーションに通知します。

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(identityIdDidChange:)
                                      name:AWSCognitoIdentityIdChangedNotification
                                      object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"identity changed from %@ to %@",
        [userInfo objectForKey:AWSCognitoNotificationPreviousId],
        [userInfo objectForKey:AWSCognitoNotificationNewId]);
}
```

iOS - Swift

ユーザーは、認証されていないゲストとしてアプリケーションにログインできます。最終的に、ユーザーはサポートされている IdP のいずれかを使用してログインすることを決定する可能性があります。Amazon Cognito は、古いアイデンティティが新しいアイデンティティと同じ識別子を維持し、プロファイルデータが自動的にマージされることを確実にします。

NSNotificationCenter はプロファイルの結合をアプリケーションに通知します。

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
   selector:"identityDidChange"
   name:AWSCognitoIdentityIdChangedNotification
   object:nil)

func identityDidChange(notification: NSNotification!) {
  if let userInfo = notification.userInfo as? [String: AnyObject] {
    print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])
```

```
    to: \(userInfo[AWSCognitoNotificationNewId])")
  }
}
```

JavaScript

認証されていないユーザーとしての開始

通常、ユーザーは認証されていないロールから開始します。このロールでは、Logins プロパティを使用しないで設定オブジェクトの認証情報プロパティを設定します。この場合、デフォルト設定は次のようになります。

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

認証されたユーザーへの切り替え

認証されていないユーザーが IdP にログインしたときに、トークンがあれば、カスタム関数を呼び出して認証情報オブジェクトを更新し Logins トークンを追加することで、認証されていないユーザーを認証されたユーザーに切り替えることができます。

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    creds.params.Logins = creds.params.Logins || {};
    creds.params.Logins[providerName] = token;

    // Expire credentials to refresh them on the next request
    creds.expired = true;
}
```

また、CognitoIdentityCredentials オブジェクトを作成することもできます。作成する場合は、既存のサービスオブジェクトの認証情報プロパティをリセットして、更新された認証情報の設定内容を反映する必要があります。「グローバル設定オブジェクトの使用」を参照してください。

CognitoIdentityCredentials オブジェクトの詳細については、AWS SDK for JavaScript API リファレンスの「AWS.CognitoIdentityCredentials」を参照してください。

Unity

ユーザーは、認証されていないゲストとしてアプリケーションにログインできます。最終的に、ユーザーはサポートされている IdP のいずれかを使用してログインすることを決定する可能性があります。Amazon Cognito は、古いアイデンティティが新しいアイデンティティと同じ識別子を維持し、プロファイルデータが自動的にマージされることを確実にします。

プロファイルの結合の通知を受け取るには、IdentityChangedEvent にサブスクライブできます。

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e)
{
    // handle the change
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

ユーザーは、認証されていないゲストとしてアプリケーションにログインできます。最終的に、ユーザーはサポートされている IdP のいずれかを使用してログインすることを決定する可能性があります。Amazon Cognito は、古い ID の一意の識別子を新しい ID に継承して、プロファイルデータが自動的に結合されるようにします。

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
 e.NewIdentityId);
};
```

Amazon Cognito Sync

Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

Amazon Cognito Sync は、アプリケーションに関連するユーザーデータをデバイス間で同期を可能にする AWS のサービス およびクライアントライブラリです。Amazon Cognito Sync は、独自のバックエンドを使用せずに、モバイルデバイスとウェブ間でユーザープロファイルデータを同期できます。クライアントライブラリはローカルにデータをキャッシュするため、アプリはデバイスの接続状態にかかわらず、データを読み書きすることができます。デバイスがオンラインになると、データを同期できます。プッシュ同期を設定すると、更新が利用できることが他のデバイスにすぐに通知されます。

Amazon Cognito ID を使用できるリージョンの詳細については、「AWS リージョン別のサービス」を参照してください。

Amazon Cognito Sync の詳細については、以下のトピックを参照してください。

トピック

- Amazon Cognito Sync の使用開始方法
- データの同期
- コールバックの処理
- プッシュ同期
- Amazon Cognito ストリーム
- Amazon Cognito イベント

Amazon Cognito Sync の使用開始方法

> ⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
> このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

AWS のサービスである Amazon Cognito Sync は、アプリケーション関連のユーザーデータのデバイス間での同期を可能にするクライアントライブラリです。これは、モバイルデバイスとウェブの全体でユーザープロファイルデータを同期させるために使用できます。クライアントライブラリはローカルにデータをキャッシュするため、アプリはデバイスの接続状態にかかわらず、データを読み書きすることができます。デバイスがオンラインのときは、データを同期し、プッシュ同期を設定すると、更新が利用できることが他のデバイスにすぐに通知されます。

Amazon Cognito で ID プールをセットアップする

Amazon Cognito Sync では、ユーザー ID を提供するための Amazon Cognito ID プールが必要です。最初にアイデンティティプールをセットアップしてから Amazon Cognito Sync を使用する必要があります。ID プールを作成し、SDKをインストールするには、Amazon Cognito ID プールの開始方法を参照してください。

データを保存し、同期する

ID プールを設定して SDK をインストールしたら、デバイス間でデータを保存して同期を開始できます。詳細については、「データの同期」を参照してください。

データの同期

> ⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。

このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

Amazon Cognito では、キーバリューペアが含まれるデータセットにユーザーデータを保存することができます。Amazon Cognito は、このデータを ID プールの ID と関連付け、アプリがログインやデバイス全体でアクセスできるようにします。このデータを Amazon Cognito サービスとエンドユーザーのデバイス間で同期するには、同期メソッドを呼び出します。 q 各データセットは、最大 1 MB のサイズにすることができます。ID には、最大 20 個のデータセットを関連付けることができます。

Amazon Cognito Sync クライアントは、ID データ用のローカルキャッシュを作成します。アプリは、キーを読み取るか書き込むときに、このローカルキャッシュとやり取りします。このやり取りにより、オフラインの状態でも、デバイス上で行われたすべての変更がすぐに利用できるようになることが保証されます。同期メソッドが呼び出された場合、サービスからの変更はデバイスにプルされ、ローカルの変更はサービスにプッシュされます。この時点で、他のデバイスが変更を利用して同期できるようになります。

Amazon Cognito Sync クライアントの初期化

Amazon Cognito Sync クライアントを初期化するには、まず認証情報プロバイダーを作成する必要があります。認証情報プロバイダーは、アプリケーションによる AWS リソースへのアクセスを可能にするために、一時的な AWS 認証情報を取得します。また、必要なヘッダーファイルをインポートする必要があります。以下のステップを使用して、Amazon Cognito Sync クライアントを初期化します。

Android

1. 「認証情報の取得」の指示に従って、認証情報プロバイダーを作成します。
2. Amazon Cognito パッケージを以下のようにインポートします: `import com.amazonaws.mobileconnectors.cognito.*;`
3. Amazon Cognito Sync を初期化 以下のように、Android アプリのコンテキスト、ID プールの ID、AWS リージョン、および初期化された Amazon Cognito の認証情報プロバイダーを渡します。

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
```

```
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 「認証情報の取得」の指示に従って、認証情報プロバイダーを作成します。
2. 以下のように、AWSCore および Cognito をインポートし、AWSCognito を初期化します。

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. CocoaPods を使用している場合は、<AWSiOSSDKv2/AWSCore.h> を AWSCore.h と置き換えてください。Amazon Cognito のインポートにも同じ構文に従います。

iOS - Swift

1. 「認証情報の取得」の指示に従って、認証情報プロバイダーを作成します。
2. 以下のように AWSCognito をインポートし、初期化します。

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. Amazon Cognito Sync Manager for JavaScript をダウンロードします。
2. プロジェクトに Sync Manager ライブラリを含めます。
3. 「認証情報の取得」の指示に従って、認証情報プロバイダーを作成します。
4. Sync Manager を以下のように初期化します。

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 「認証情報の取得」の指示に従って、CognitoAWSCredentials のインスタンスを作成します。
2. CognitoSyncManager のインスタンスを作成します。以下のように、CognitoAwsCredentials オブジェクトと AmazonCognitoSyncConfig を渡し、少なくともリージョン・セットを含めます。

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 「認証情報の取得」の指示に従って、CognitoAWSCredentials のインスタンスを作成します。
2. CognitoSyncManager のインスタンスを作成します。以下のように、CognitoAwsCredentials オブジェクトと AmazonCognitoSyncConfig を渡し、少なくともリージョン・セットを含めます。

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

データセットについて

Amazon Cognito は、ユーザーのプロファイルデータをデータセットに編成します。各データセットはキーバリューペアの形式で、最大 1 MB のデータを含むことができます。データセットは、同期することができる最も細かいエンティティです。データセットで実行される読み取りと書き込みのオペレーションは、同期メソッドが呼び出されるまでしか、ローカルストアに影響しません。Amazon Cognito は一意の文字列によってデータセットを識別します。以下のように新しいデータセットを作成するか、既存のデータセットを開くことができます。

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

データセットを削除するには、以下のように、最初にそれをローカルストレージから削除するためのメソッドを呼び出し、次に synchronize メソッドを呼び出して Amazon Cognito からそのデータセットを削除します。

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

データセットを削除するには、以下のように、最初にそれをローカルストレージから削除するためのメソッドを呼び出し、次に synchronize メソッドを呼び出して Amazon Cognito からそのデータセットを削除します。

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

データセットを削除するには、最初にそれをローカルストレージから削除するためのメソッドを呼び出し、次に synchronize メソッドを呼び出して Amazon Cognito からそのデータセットを削除します。

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
   // ...
```

```
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

データセットからキーを削除するには、以下のように Remove を設定します。

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

データセットを削除するには、以下のように、最初にそれをローカルストレージから削除するためのメソッドを呼び出し、次に synchronize メソッドを呼び出して Amazon Cognito からそのデータセットを削除します。

```
dataset.Delete();
dataset.SynchronizeAsync();
```

データセットのデータの読み取りと書き込み

Amazon Cognito データセットはディクショナリとして機能し、値はキーでアクセスできます。データセットのキーと値は、データセットがディクショナリであるかのように読み取り、追加、または変更できます。

データセットに書き込む値は、同期メソッドを呼び出すまで、データのローカルにキャッシュされたコピーのみに影響することに注意してください。

Android

```
String value = dataset.get("myKey");
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
```

```
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.stringForKey("myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
});

dataset.put('newKey', 'newValue', function(err, record) {
  console.log(record);
});

dataset.remove('oldKey', function(err, record) {
  console.log(success);
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value
string myValue = dataset.Get("myKey");

// Create a record in a dataset and synchronize with the server
dataset.OnSyncSuccess += SyncSuccessCallback;
dataset.Put("myKey", "myValue");
dataset.SynchronizeAsync();

void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Android

以下のように、remove メソッドを使用して、データセットからキーを削除できます。

```
dataset.remove("myKey");
```

iOS - Objective-C

データセットからキーを削除するには、以下のように removeObjectForKey を設定します。

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

データセットからキーを削除するには、以下のように removeObjectForKey を設定します。

```
dataset.removeObjectForKey("myKey")
```

Unity

データセットからキーを削除するには、以下のように Remove を設定します。

```
dataset.Remove("myKey");
```

Xamarin

Remove を使用して、データセットからキーを削除できます。

```
dataset.Remove("myKey");
```

同期ストアでのローカルデータの同期

Android

synchronize メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

```
dataset.synchronize(syncCallback);
```

synchronize メソッドは、以下に説明する SyncCallback インターフェイスの実装を受け取ります。

synchronizeOnConnectivity() メソッドは、接続が利用可能な場合に同期を試みます。接続が直ちに利用可能な場合、synchronizeOnConnectivity() は synchronize() のように動作します。それ以外の場合、接続の変更をモニタリングし、接続が利用可能になると同期を実行します。synchronizeOnConnectivity() が複数回呼び出された場合、最後の同期リクエストのみが保持され、最後のコールバックのみが実行されます。データセットまたはコールバックがガーベージコレクションの対象になった場合、このメソッドは同期を実行せず、コールバックは実行されません。

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

iOS - Objective-C

synchronize メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

synchronize メソッドは非同期で、応答を処理するために AWSTask オブジェクトを返します。

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

synchronizeOnConnectivity メソッドは、デバイスが接続されているときに同期を試みます。最初に、synchronizeOnConnectivity は接続を確認し、デバイスがオンラインである場合、すぐに synchronize を呼び出して、試行に関連付けられた AWSTask オブジェクトを返します。

デバイスがオフラインの場合、synchronizeOnConnectivity は、1) 次回にデバイスがオンラインになったときに同期をスケジュールし、2) 結果が nil の AWSTask を返します。スケジュールされた同期が有効なのは、データセットオブジェクトのライフサイクルの間のみです。接続が回復する前にアプリが終了された場合、データは同期されません。予定された同期中にイベントが発生した場合に通知を受けるようにする場合は、AWSCognito で見つかった通知のオブザーバーを追加する必要があります。

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

iOS - Swift

synchronize メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

synchronize メソッドは非同期で、応答を処理するために AWSTask オブジェクトを返します。

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in

            if task.isCancelled {
                // Task cancelled.
            } else if task.error != nil {
                // Error while executing task
            } else {
                // Task succeeded. The data was saved in the sync store.
            }
            return task
})
```

synchronizeOnConnectivity メソッドは、デバイスが接続されているときに同期を試みます。最初に、synchronizeOnConnectivity は接続を確認し、デバイスがオンラインである場合、すぐに synchronize を呼び出して、試行に関連付けられた AWSTask オブジェクトを返します。

デバイスがオフラインの場合、synchronizeOnConnectivity は、1) 次回にデバイスがオンラインになったときに同期をスケジュールし、2) 結果が nil の AWSTask オブジェクトを返します。スケジュールされた同期が有効なのは、データセットオブジェクトのライフサイクルの間のみです。接続が回復する前にアプリが終了された場合、データは同期されません。予定された同期中にイベントが

発生した場合に通知を受けるようにする場合は、AWSCognito で見つかった通知のオブザーバーを追加する必要があります。

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

JavaScript

synchronize メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

```
dataset.synchronize();
```

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

Unity

同期メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

```
dataset.Synchronize();
```

同期は非同期で実行され、データセット内で指定できる複数のコールバックの 1 つを呼び出します。

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

Xamarin

synchronize メソッドは、ローカルキャッシュデータと Amazon Cognito Sync ストアに格納されたデータを比較します。リモート変更は Amazon Cognito Sync ストアからプルされ、競合が発生した場合は競合解決が呼び出されます。そして、デバイス上の更新された値がサービスにプッシュされます。データセットを同期するには、その synchronize メソッドを呼び出します。

```
dataset.SynchronizeAsync();
```

データセットの同期と、さまざまなコールバックに関する詳細については、「コールバックの処理」を参照してください。

コールバックの処理

> ⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
> このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

このセクションでは、コールバックを処理する方法について説明します。

Android

SyncCallback インターフェイス

SyncCallback インターフェイスを実装することで、データセットの同期に関するアプリの通知を受け取ることができます。これによりアプリは、ローカルデータの削除、認証されていないプロファイルや認証されたプロファイルの結合、同期の競合の解決に関するアクティブな決定ができます。インターフェイスで必要となる次のメソッドを実装する必要があります。

- `onSuccess()`
- `onFailure()`
- `onConflict()`
- `onDatasetDeleted()`
- `onDatasetsMerged()`

すべてのコールバックを指定しない場合は、すべてのコールバックに対して空のデフォルト実装を提供する、`DefaultSyncCallback` クラスを使用することもできます。

onSuccess

onSuccess() コールバックは、同期ストアから正常にデータセットをダウンロードしたときにトリガーされます。

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

onFailure() は、同期中に例外が発生した場合に呼び出されます。

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

ローカルストアと同期ストアで同じキーが変更された場合に、競合が発生する可能性があります。onConflict() メソッドは、競合の解決を処理します。このメソッドを実装しない場合、Amazon Cognito Sync クライアントはデフォルトで最新の変更を使用します。

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());

        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue);
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
```

```
}
```

onDatasetDeleted

データセットが削除されると、Amazon Cognito クライアントは SyncCallback インターフェイスを使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。onDatasetDeleted( ) メソッドを実装して、クライアント SDK にローカルデータの処理方法を指定します。

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、onDatasetsMerged( ) メソッドを通じて結合について通知されます。

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
    return false;
}
```

iOS - Objective-C

同期の通知

Amazon Cognito クライアントは、同期コール中に多数の NSNotification イベントを生成します。標準の NSNotificationCenter を通じて、これらの通知をモニタリングするよう登録できます。

```
[NSNotificationCenter defaultCenter]
  addObserver:self
  selector:@selector(myNotificationHandler:)
  name:NOTIFICATION_TYPE
  object:nil];
```

Amazon Cognito は、以下にリストされている 5 つの通知タイプをサポートします。

AWSCognitoDidStartSynchronizeNotification

同期操作の開始中に呼び出されます。userInfo は、同期されているデータセットの名前であるキーデータセットを含みます。

AWSCognitoDidEndSynchronizeNotification

同期操作が完了した (正常またはその反対) ときに呼び出されます。userInfo は、同期されているデータセットの名前であるキーデータセットを含みます。

AWSCognitoDidFailToSynchronizeNotification

同期操作が失敗すると呼び出されます。userInfo は、同期されているデータセットの名前であるキーデータセットと、失敗の原因となったエラーを含むキーエラーを含みます。

AWSCognitoDidChangeRemoteValueNotification

ローカル変更が Amazon Cognito に正常にプッシュされたときに呼び出されます。userInfo は、同期されているデータセットの名前であるキーデータセットと、プッシュされたレコードキーの NSArray を含むキーを含みます。

AWSCognitoDidChangeLocalValueFromRemoteNotification

ローカルの値が同期操作が原因で変更されたときに呼び出されます。userInfo は、同期されているデータセットの名前であるキーデータセットと、変更されたレコードキーの NSArray を含むキーを含みます。

競合解決ハンドラ

同期操作中に、ローカルストアと同期ストアで同じキーが変更された場合、競合が発生する可能性があります。競合解決ハンドラを設定していない場合、Amazon Cognito はデフォルトで最新の更新を選択します。

AWSCognitoRecordConflictHandler を実装して割り当てることで、デフォルトの競合解決は変更できます。AWSCognitoConflict 入力パラメータには、ローカルにキャッシュされたデータと、同期ストアの競合するレコードの両方の AWSCognitoRecord オブジェクトが含まれています。AWSCognitoConflict を使用すると、ローカルレコード [conflict resolveWithLocalRecord]、リモートレコード [conflict resolveWithRemoteRecord]、またはまったく新しい値 [conflict resolveWithValue:value] との競合を解決できます。このメソッドから nil が返されると、同期は続行できなくなり、次回に同期処理が開始されると、競合が再び発生します。

クライアントレベルで競合解決ハンドラを設定できます。

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

または、データセットレベルで設定できます。

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

データセットが削除されたハンドラ

データセットが削除されると、Amazon Cognito クライアントは AWSCognitoDatasetDeletedHandler を使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。AWSCognitoDatasetDeletedHandler が実装されていない場合、ローカルデータは自動的に消去されます。消去する前にローカルデータのコピーを維持するか、ローカルデータを維持する場合は、AWSCognitoDatasetDeletedHandler を実装します。

クライアントレベルでデータセットが削除されたハンドラを設定することができます。

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

または、データセットレベルで設定できます。

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

データセットの結合ハンドラ

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、DatasetMergeHandler メソッドを通じて結合について通知されます。ハンドラは、ルートデータセットの名前と、ルートデータセットの結合としてマーキングされているデータセット名の配列を受け取ります。

DatasetMergeHandler が実装されていない場合、これらのデータセットは無視されますが、ID の最大 20 個の合計データセットで引き続き領域を占有します。

クライアントレベルでデータセット結合ハンドラをセットアップすることもできます。

```
client.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
       [merged clear];
       [merged synchronize];
    }
};
```

または、データセットレベルで設定できます。

```
dataset.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
        // do something with the data if it differs from existing dataset
        ...
        // now delete it
        [merged clear];
        [merged synchronize];
    }
};
```

iOS - Swift

同期の通知

Amazon Cognito クライアントは、同期コール中に多数の NSNotification イベントを生成します。標準の NSNotificationCenter を通じて、これらの通知をモニタリングするよう登録できます。

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
   selector: "myNotificationHandler",
   name:NOTIFICATION_TYPE,
   object:nil)
```

Amazon Cognito は、以下にリストされている 5 つの通知タイプをサポートします。

AWSCognitoDidStartSynchronizeNotification

同期操作の開始中に呼び出されます。`userInfo` は、同期されているデータセットの名前であるキーデータセットを含みます。

AWSCognitoDidEndSynchronizeNotification

同期操作が完了した (正常またはその反対) ときに呼び出されます。`userInfo` は、同期されているデータセットの名前であるキーデータセットを含みます。

AWSCognitoDidFailToSynchronizeNotification

同期操作が失敗すると呼び出されます。`userInfo` は、同期されているデータセットの名前であるキーデータセットと、失敗の原因となったエラーを含むキーエラーを含みます。

AWSCognitoDidChangeRemoteValueNotification

ローカル変更が Amazon Cognito に正常にプッシュされたときに呼び出されます。`userInfo` は、同期されているデータセットの名前であるキーデータセットと、プッシュされたレコードキーの NSArray を含むキーを含みます。

AWSCognitoDidChangeLocalValueFromRemoteNotification

ローカルの値が同期操作が原因で変更されたときに呼び出されます。`userInfo` は、同期されているデータセットの名前であるキーデータセットと、変更されたレコードキーの NSArray を含むキーを含みます。

競合解決ハンドラ

同期操作中に、ローカルストアと同期ストアで同じキーが変更された場合、競合が発生する可能性があります。競合解決ハンドラを設定していない場合、Amazon Cognito はデフォルトで最新の更新を選択します。

`AWSCognitoRecordConflictHandler` を実装して割り当てることで、デフォルトの競合解決は変更できます。`AWSCognitoConflict` 入力パラメータには、ローカルにキャッシュされたデータ

と、同期ストアの競合するレコードの両方の AWSCognitoRecord オブジェクトが含まれています。AWSCognitoConflict を使用すると、ローカルレコード [conflict resolveWithLocalRecord]、リモートレコード [conflict resolveWithRemoteRecord]、またはまったく新しい値 [conflict resolveWithValue:value] との競合を解決できます。このメソッドから nil が返されると、同期は続行できなくなり、次回に同期処理が開始されると、競合が再び発生します。

クライアントレベルで競合解決ハンドラを設定できます。

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

または、データセットレベルで設定できます。

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

データセットが削除されたハンドラ

データセットが削除されると、Amazon Cognito クライアントは AWSCognitoDatasetDeletedHandler を使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。AWSCognitoDatasetDeletedHandler が実装されていない場合、ローカルデータは自動的に消去されます。消去する前にローカルデータのコピーを維持するか、ローカルデータを維持する場合は、AWSCognitoDatasetDeletedHandler を実装します。

クライアントレベルでデータセットが削除されたハンドラを設定することができます。

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

または、データセットレベルで設定できます。

```
dataset.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

データセットの結合ハンドラ

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、DatasetMergeHandler メソッドを通じて結合について通知されます。ハンドラは、ルートデータセットの名前と、ルートデータセットの結合としてマーキングされているデータセット名の配列を受け取ります。

DatasetMergeHandler が実装されていない場合、これらのデータセットは無視されますが、ID の最大 20 個の合計データセットで引き続き領域を占有します。

クライアントレベルでデータセット結合ハンドラをセットアップすることもできます。

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

または、データセットレベルで設定できます。

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
```

```
            merged.synchronize()
        }
    }
}
```

JavaScript

同期のコールバック

データセットで synchronize() を実行するときに、オプションで次の各状態を処理するコールバックを指定できます。

```
dataset.synchronize({

   onSuccess: function(dataset, newRecords) {
      //...
   },

   onFailure: function(err) {
      //...
   },

   onConflict: function(dataset, conflicts, callback) {
      //...
   },

   onDatasetDeleted: function(dataset, datasetName, callback) {
      //...
   },

  onDatasetMerged: function(dataset, datasetNames, callback) {
      //...
  }

});
```

onSuccess()

onSuccess( ) コールバックは、同期ストアから正常にデータセットを更新したときにトリガーされます。コールバックを定義しない場合、同期はユーザーの操作なしに成功します。

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
```

```
}
```

onFailure()

onFailure()は、同期中に例外が発生した場合に呼び出されます。コールバックを定義しない場合、同期はユーザーの操作なしに失敗します。

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
}
```

onConflict()

ローカルストアと同期ストアで同じキーが変更された場合に、競合が発生する可能性があります。onConflict() メソッドは、競合の解決を処理します。このメソッドを実行しない場合、競合があると同期は破棄されます。

```
onConflict: function(dataset, conflicts, callback) {

   var resolved = [];

   for (var i=0; i<conflicts.length; i++) {

      // Take remote version.
      resolved.push(conflicts[i].resolveWithRemoteRecord());

      // Or... take local version.
      // resolved.push(conflicts[i].resolveWithLocalRecord());

      // Or... use custom logic.
      // var newValue = conflicts[i].getRemoteRecord().getValue() +
 conflicts[i].getLocalRecord().getValue();
      // resolved.push(conflicts[i].resovleWithValue(newValue);

   }

   dataset.resolve(resolved, function() {
      return callback(true);
   });

   // Or... callback false to stop the synchronization process.
   // return callback(false);
```

```
}
```

onDatasetDeleted()

データセットが削除されると、Amazon Cognito クライアントは onDatasetDeleted( ) コールバックを使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。デフォルトでは、データセットは削除されません。

```
onDatasetDeleted: function(dataset, datasetName, callback) {

   // Return true to delete the local copy of the dataset.
   // Return false to handle deleted datasets outside the synchronization callback.

   return callback(true);

}
```

onDatasetMerged()

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、onDatasetsMerged( ) コールバックを通じて結合について通知されます。

```
onDatasetMerged: function(dataset, datasetNames, callback) {

   // Return true to continue the synchronization process.
   // Return false to handle dataset merges outside the synchronization callback.

   return callback(false);

}
```

Unity

データセットを開くか作成した後は、別のコールバックを設定できます。これは、Synchronize メソッドを使用したときにトリガーされます。この方法により、コールバックを登録することができます。

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
```

```
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

SyncSuccess と SyncFailure は、+= を = の代わりに使用し、複数のコールバックをサブスクライブできるようにします。

OnSyncSuccess

OnSyncSuccess コールバックは、クラウドから正常にデータセットを更新したときにトリガーされます。コールバックを定義しない場合、同期はユーザーの操作なしに成功します。

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

OnSyncFailureは、同期中に例外が発生した場合に呼び出されます。コールバックを定義しない場合、同期はユーザーの操作なしに失敗します。

```
private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}
```

OnSyncConflict

ローカルストアと同期ストアで同じキーが変更された場合に、競合が発生する可能性があります。OnSyncConflict コールバックは競合の解決を処理します。このメソッドを実行しない場合、競合があると同期は破棄されます。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
```

```
    Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Debug.LogWarning("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

データセットが削除されると、Amazon Cognito クライアントは OnDatasetDeleted コールバックを使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。デフォルトでは、データセットは削除されません。

```
private bool HandleDatasetDeleted(Dataset dataset)
  {
      Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
      // Do clean up if necessary
      // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
      return true;
  }
```

OnDatasetMerged

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、OnDatasetsMerged コールバックを通じて結合について通知されます。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
```

```
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;

            //Implement your merge logic here

            mergedDataset.Delete(); //Delete the dataset locally
            mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
 fired again
            mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEvent e2) => {
                localDataset.Synchronize(); //Continue the sync operation that was
 interrupted by the merge
            };
            mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so
 will leave us in an inconsistent state
        };
        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.Synchronize(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

Xamarin

データセットを開くか作成した後は、別のコールバックを設定できます。これは、Synchronize メソッドを使用したときにトリガーされます。この方法により、コールバックを登録することができます。

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

SyncSuccess と SyncFailure は、+= を = の代わりに使用し、複数のコールバックをサブスクライブできるようにします。

OnSyncSuccess

OnSyncSuccess コールバックは、クラウドから正常にデータセットを更新したときにトリガーされます。コールバックを定義しない場合、同期はユーザーの操作なしに成功します。

```
private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

OnSyncFailureは、同期中に例外が発生した場合に呼び出されます。コールバックを定義しない場合、同期はユーザーの操作なしに失敗します。

```
private void HandleSyncFailure(object sender, SyncFailureEventArgs e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync failed");
    }
}
```

OnSyncConflict

ローカルストアと同期ストアで同じキーが変更された場合に、競合が発生する可能性があります。OnSyncConflict コールバックは競合の解決を処理します。このメソッドを実行しない場合、競合があると同期は破棄されます。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Console.WriteLine("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Console.WriteLine("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
```

```
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

データセットが削除されると、Amazon Cognito クライアントは OnDatasetDeleted コールバックを使用して、データセットのローカルキャッシュコピーも削除されるべきかどうかを確認します。デフォルトでは、データセットは削除されません。

```
private bool HandleDatasetDeleted(Dataset dataset)
{
    Console.WriteLine(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
    // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

以前に接続されていない 2 つの ID がリンクされ、すべてのデータセットが結合されます。アプリケーションは、OnDatasetsMerged コールバックを通じて結合について通知されます。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

            //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
```

```
        mergedDataset.SynchronizeAsync(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

プッシュ同期

⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

Amazon Cognito は自動的にアイデンティティとデバイス間の関係を追跡します。プッシュ同期機能を使用すると、ID データが変更されたときに、特定の ID のすべてのインスタンスに確実に通知されます。プッシュ同期は、特定のアイデンティティの同期ストアデータが変更されるたびに、そのアイデンティティに関連付けられているすべてのデバイスが、この変更を伝えるサイレントプッシュ通知を受け取ることを確実にします。

ℹ Note

プッシュ同期は JavaScript、Unity、Xamarin ではサポートされません。

プッシュ同期を使用する前に、まずプッシュ同期用にアカウントをセットアップし、Amazon Cognito コンソールでプッシュ同期を有効にする必要があります。

Amazon Simple Notification Service (Amazon SNS) アプリケーションを作成する

SNS デベロッパーガイドの説明に従って、サポートされているプラットフォーム用の Amazon SNS アプリを作成し、設定します。

Amazon Cognito コンソールでプッシュ同期を有効にする

プッシュ同期は Amazon Cognito コンソールを使って有効にできます。コンソールのホームページから、次の操作を行います。

1. プッシュ同期を有効にする ID プールの名前をクリックします。アイデンティティプールの [Dashboard] (ダッシュボード) ページが表示されます。
2. [ダッシュボード] ページの右上にある、[ID プールの管理] をクリックします。[フェデレーティッドアイデンティティ] ページが表示されます。
3. 下にスクロールし、同期をプッシュしますをクリックして展開します。
4. [Service role] (サービスロール) ドロップダウンメニューで、SNS 通知を送信するアクセス許可を Cognito に付与する IAM ロールを選択します。[Create role] (ロールの作成) をクリックして、AWS IAM コンソールで ID プールに関連付けられたロールを作成または変更します。
5. プラットフォームアプリケーションを選択し、[Save Changes] (変更の保存) をクリックします。
6. アプリケーションへの SNS アクセスの許可

AWS Identity and Access Management コンソールで、完全な Amazon SNS アクセス権を持つように IAM ロールを設定するか、完全な Amazon SNS アクセス権を持つ新しいロールを作成します。次のロール信頼ポリシーの例では、Amazon Cognito Sync に IAM ロールを引き受ける制限付きの機能を付与しています。Amazon Cognito Sync は、条件 `aws:SourceArn` の ID プールと条件 `aws:SourceAccount` のアカウントの両方に代わり行う場合にのみ、その役割を担うことができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
```

```
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
                }
            }
        }
    ]
}
```

IAM ロールの詳細については、「ロール (委任とフェデレーション) 」を参照してください。

アプリケーションでのプッシュ同期の使用: Android

アプリケーションは、Google Play サービスをインポートする必要があります。Android SDK マネージャーから、最新バージョンの Google Play SDK をダウンロードできます。Android のドキュメントの「Android Implementation」に従ってアプリを登録し、GCM から登録 ID を受け取ります。登録 ID を取得したら、以下のスニペットにあるように、デバイスを Amazon Cognito に登録する必要があります。

```
String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}
```

これで、デバイスをサブスクライブして、特定のデータセットから更新を受け取ることができます。

```
Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

データセットからプッシュ通知の受け取りを中止するには、unsubscribe メソッドを呼び出します。CognitoSyncManager オブジェクトのすべてのデータセット (または特定のサブセット) にサブスクライブするには、subscribeAll() を使用します。

```
if (client.isDeviceRegistered()) {
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

Android BroadcastReceiver オブジェクトの実装で、最新バージョンの変更されたデータセットを確認して、アプリをもう一度同期する必要があるかどうか判断できます。

```
@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName;
    long syncCount = update.getSyncCount;

    Dataset dataset = client.openOrCreateDataset(datasetName);

    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
        dataset.synchronize(new SyncCallback() {
            // ...
```

```
        });
    }

}
```

プッシュ通知のペイロードでは、次のキーを利用できます。

- source: cognito-sync。これは、通知間の差別化要素として機能します。
- identityPoolId: ID プールの ID。これは検証や追加の情報用に使用できますが、レシーバーの観点からは不可欠ではありません。
- identityId: プール内のアイデンティティ ID。
- datasetName: 更新されたデータセットの名前。これは openOrCreateDataset 呼び出しの目的で利用できます。
- syncCount: リモートデータセットの同期カウント。これは、ローカルデータセットが古くなっていないこと、および入力同期が新しいことを確認する方法として使用できます。

アプリケーションでプッシュ同期を使用する: iOS - Objective-C

アプリ用のデバイストークンを取得するには、Apple ドキュメントの「Registering for Remote Notifications」を参照してください。APN から NSData オブジェクトとしてデバイストークンを受け取ったら、以下のように、Sync クライアントの registerDevice: メソッドを使用して、デバイスを Amazon Cognito に登録する必要があります。

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
    [[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to registerDevice: %@", task.error);
        } else {
            NSLog(@"Successfully registered device with id: %@", task.result);
        }
        return nil;
      }
    ];
```

デバッグモードでは、デバイスは APN サンドボックスで登録されます。リリースモードでは、APN で登録されます。特定のデータセットからアップデートを受けるには、subscribe メソッドを使用します。

```
[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to dataset: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

データセットからプッシュ通知の受け取りを中止するには、unsubscribe メソッドを呼び出します。

```
[[[syncClient openOrCreateDataset:@”MyDataset”] unsubscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
        } else {
            NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

AWSCognito オブジェクトのすべてのデータセットをサブスクライブするには、subscribeAll を呼び出します。

```
[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to all datasets: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to all datasets: %@", task.result);
        }
        return nil;
      }
    ];
```

subscribeAll を呼び出す前に、少なくとも各データセットで 1 回同期を行い、データセットがサーバー上に存在するようにします。

プッシュ通知に応答するには、アプリの代理で didReceiveRemoteNotification メソッドを実装する必要があります。

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

通知ハンドラを使用して通知を投稿する場合は、データセットへのハンドルがある、アプリケーションの他の場所で通知に応答することができます。次のように通知にサブスクライブしているとします。

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
    name: :@"CognitoPushNotification" object:nil];
```

その場合は、次のように通知に応答できます。

```
- (void)didReceivePushSync:(NSNotification*)notification
    {
        NSDictionary * data = [(NSDictionary *)[notification object]
 objectForKey:@"data"];
        NSString * identityId = [data objectForKey:@"identityId"];
        NSString * datasetName = [data objectForKey:@"datasetName"];
        if([self.dataset.name isEqualToString:datasetName] && [self.identityId
 isEqualToString:identityId]){
            [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
                if(!task.error){
                    NSLog(@"Successfully synced dataset");
                }
                return nil;
            }];
        }
    }
```

プッシュ通知のペイロードでは、次のキーを利用できます。

- source: cognito-sync。これは、通知間の差別化要素として機能します。

- identityPoolId: ID プールの ID。これは検証や追加の情報用に使用できますが、レシーバーの観点からは不可欠ではありません。
- identityId: プール内のアイデンティティ ID。
- datasetName: 更新されたデータセットの名前。これは openOrCreateDataset 呼び出しの目的で利用できます。
- syncCount: リモートデータセットの同期カウント。これは、ローカルデータセットが古くなっていないこと、および入力同期が新しいことを確認する方法として使用できます。

アプリケーションでプッシュ同期を使用する: iOS - Swift

アプリ用のデバイストークンを取得するには、Apple ドキュメントの「Registering for Remote Notifications」を参照してください。APN から NSData オブジェクトとしてデバイストークンを受け取ったら、以下の のように、Sync クライアントの registerDevice: メソッドを使用して、デバイスを Amazon Cognito に登録する必要があります。

```
let syncClient = AWSCognito.default()
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) ->
 AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)

    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

デバッグモードでは、デバイスは APN サンドボックスで登録されます。リリースモードでは、APN で登録されます。特定のデータセットからアップデートを受けるには、subscribe メソッドを使用します。

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to dataset: \(task.result)")
```

```
  }
  return task
})
```

データセットからプッシュ通知の受け取りを中止するには、unsubscribe メソッドを呼び出します。

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully unsubscribed to dataset: \(task.result)")
  }
  return task
})
```

AWSCognito オブジェクトのすべてのデータセットをサブスクライブするには、subscribeAll を呼び出します。

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to all datasets: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to all datasets: \(task.result)")
  }
  return task
})
```

subscribeAll を呼び出す前に、少なくとも各データセットで 1 回同期を行い、データセットがサーバー上に存在するようにします。

プッシュ通知に応答するには、アプリの代理で didReceiveRemoteNotification メソッドを実装する必要があります。

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {
```

```
 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

通知ハンドラを使用して通知を投稿する場合は、データセットへのハンドルがある、アプリケーションの他の場所で通知に応答することができます。次のように通知にサブスクライブしているとします。

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
   name:"CognitoPushNotification",
   object:nil)
```

その場合は、次のように通知に応答できます。

```
func didReceivePushSync(notification: NSNotification) {
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String:
 AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
          dataset.synchronize().continueWithBlock {(task) -> AnyObject! in
              if task.error == nil {
                print("Successfully synced dataset")
              }
              return nil
          }
        }
    }
}
```

プッシュ通知のペイロードでは、次のキーを利用できます。

- `source`: cognito-sync。これは、通知間の差別化要素として機能します。
- `identityPoolId`: ID プールの ID。これは検証や追加の情報用に使用できますが、レシーバーの観点からは不可欠ではありません。
- `identityId`: プール内のアイデンティティ ID。

- `datasetName`: 更新されたデータセットの名前。これは `openOrCreateDataset` 呼び出しの目的で利用できます。
- `syncCount`: リモートデータセットの同期カウント。これは、ローカルデータセットが古くなっていないこと、および入力同期が新しいことを確認する方法として使用できます。

Amazon Cognito ストリーム

> ⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
> このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

Amazon Cognito ストリームは、Amazon Cognito に保存されているデータに対する制御と洞察をデベロッパーに提供します。これで、デベロッパーはデータが更新および同期されるたびにイベントを受け取るように Kinesis ストリームを設定できるようになります。Amazon Cognito は、所有する Kinesis ストリームに各データセットの変更をリアルタイムでプッシュできます。

Amazon Cognito ストリームを使用すると、すべての Sync データを Kinesis に移動することができます。その後、これらをさらなる分析のために Amazon Redshift などのデータウェアハウスツールにストリーミングできます。Kinesis の詳細については、「Amazon Kinesis の使用開始方法」を参照してください。

ストリームの設定

Amazon Cognito ストリームは、Amazon Cognito コンソールで設定できます。Amazon Cognito コンソールで Amazon Cognito ストリームを有効にするには、パブリッシュ先の Kinesis ストリームと、選択されたストリームにイベントを配置するためのアクセス許可を Amazon Cognito に付与する IAM ロールを選択する必要があります。

コンソールのホームページから、次の操作を行います。

1. Amazon Cognito ストリームをセットアップする ID プールの名前をクリックします。アイデンティティプールの [Dashboard] (ダッシュボード) ページが表示されます。

2. [ダッシュボード] ページの右上にある、[ID プールの管理] をクリックします。[Manage Federated Identities] (フェデレーティッド ID の管理) ページが表示されます。
3. 下にスクロールし、[Cognito ストリーム] をクリックして展開します。
4. [ストリーム名] ドロップダウンメニューで、既存の Kinesis ストリームの名前を選択します。または、[ストリームの作成] をクリックしてストリームを作成し、ストリーム名とシャード数を入力します。シャードに関する情報、およびストリームに必要なシャード数の見積りに関するヘルプについては、Kinesis デベロッパーガイドを参照してください。
5. [Publish role] (パブリッシュロール) ドロップダウンメニューで、ストリームをパブリッシュするためアクセス権限を Amazon Cognito に付与する IAM ロールを選択します。[Create role] (ロールの作成) をクリックして、AWS IAM コンソールで ID プールに関連付けられたロールを作成または変更します。
6. [ストリームの状態] ドロップダウンメニューで、[有効] を選択してストリームの更新を有効にします。[Save Changes] (変更の保存) をクリックします。

Amazon Cognito ストリームが正常に設定されると、この ID プール内のデータセットに対する後続の更新がすべてストリームに送信されるようになります。

ストリームのコンテンツ

ストリームに送信される各レコードは、単一の同期を表します。ストリームに送信されるレコードの例を次に示します。

```
{
    "identityPoolId": "Pool Id",
    "identityId": "Identity Id",
    "dataSetName": "Dataset Name",
    "operation": "(replace|remove)",
    "kinesisSyncRecords": [
        {
            "key": "Key",
            "value": "Value",
            "syncCount": 1,
            "lastModifiedDate": 1424801824343,
            "deviceLastModifiedDate": 1424801824343,
            "op": "(replace|remove)"
        },
        ...
    ],
    "lastModifiedDate": 1424801824343,
```

```
    "kinesisSyncRecordsURL": "S3Url",
    "payloadType": "(S3Url|Inline)",
    "syncCount": 1
}
```

Kinesis の最大ペイロードサイズである 1 MB を超える更新については、Amazon Cognito にその更新の完全なコンテンツが格納されている事前署名済みの Amazon S3 の URL が含まれます。

Amazon Cognito ストリームを設定した後で Kinesis ストリームを削除する、または Amazon Cognito Sync がロールを引き受けられないようにロール信頼のアクセス許可を変更する場合、Amazon Cognito ストリームを無効にします。Kinesis ストリームを再作成するか、ロールを修正してから、ストリームを再有効化する必要があります。

一括パブリッシュ

Amazon Cognito ストリームが設定されると、ID プール内の既存のデータに対して一括パブリッシュ操作を実行できるようになります。一括パブリッシュ操作をコンソール経由で開始する、または API 経由で直接開始すると、Amazon Cognito が、更新を受信しているものと同じストリームに対してこのデータのパブリッシュを開始します。

Amazon Cognito は、一括パブリッシュ操作の使用時にストリームに送信されるデータの一意性を保証しません。更新については、更新としてだけではなく、一括パブリッシュの一部として同じものを受け取る場合があります。ストリームからレコードを処理する際は、このことを念頭に置いてください。

すべてのストリームを一括パブリッシュするには、「ストリームの設定」の手順 1～6 に従ってから、[Start bulk publish] (一括パブリッシュを開始) をクリックします。一括パブリッシュ操作は、常に 1 つの進行中操作に制限され、24 時間ごとに 1 つの正常な一括パブリッシュリクエストに制限されます。

Amazon Cognito イベント

> ⚠ Amazon Cognito Sync を初めて使用する場合は、AWS AppSync を使用してください。Amazon Cognito Sync と同様に、AWS AppSync はデバイス間でアプリケーションデータを同期化するためのサービスです。
> このサービスは、アプリの設定やゲームステートといったユーザーデータの同期化を可能にします。また、複数のユーザーが同期し、共有されたデータでリアルタイムにコラボレートできるようにすることで、これらの機能を拡張します。

Amazon Cognito Events は、Amazon Cognito Sync での重要なイベントへの対応として AWS Lambda 関数を実行することを可能にします。Amazon Cognito は、データセットが同期されるときに Sync Trigger イベントを生成します。Sync Trigger イベントを使用して、ユーザーがデータを更新するときにアクションを実行できます。この関数は、クラウド内に保存し、ユーザーの他のデバイスに同期する前にデータを評価して、オプションで操作できます。これは、ユーザーの他のデバイスに対して同期する前に、デバイスから受信するデータを検証したり、プレーヤーが新しいレベルに達したときに賞を与えるなど、受信データに基づいてデータセットの他の値を更新する場合に役立ちます。

以下のステップでは、Amazon Cognito のデータセットが同期されるたびに実行される Lambda 関数のセットアップ手順を説明します。

> **Note**
>
> Amazon Cognito イベントを使用するときは、Amazon Cognito ID から取得した認証情報しか使用できません。関連付けられた Lambda 関数があっても、AWS アカウントの認証情報 (デベロッパー認証情報) を使用して `UpdateRecords` を呼び出すと、Lambda 関数は呼び出されません。

AWS Lambda で関数の作成

Lambda を Amazon Cognito と統合するには、最初に Lambda で関数を作成する必要があります。これを行うには、以下の手順を実行します。

Amazon Cognito での Lambda 関数の選択

1. Lambda コンソールを開きます。
2. [Create a Lambda function] (Lambda 関数の作成) をクリックします。
3. [Select blueprint] (設計図の選択) 画面で、「cognito-sync-trigger」を検索して選択します。
4. [Configure event sources] (イベントソースの設定) 画面で、イベントソースのタイプを [Cognito Sync Trigger] に設定し、ID プールを選択します。[Next] (次へ) をクリックします。

> **Note**
>
> Amazon Cognito Sync トリガーをコンソール外で設定する場合、Amazon Cognito が関数を呼び出せるように、Lambda リソースベースの権限を追加する必要があります。この許可は、Lambda コンソール (「Using resource-based policies for AWS Lambda」を参照) から追加するか、Lambda AddPermission オペレーションを使用して追加できます。

Lambda リソースベースのポリシーの例
次の AWS Lambda リソースベースのポリシーは、Amazon Cognito に Lambda 関数を呼び出す制限付きの機能を付与します。Amazon Cognitoは、条件 `aws:SourceArn` のID プールと条件 `aws:SourceAccount` のアカウントに代わりに関数を呼び出すことができます。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito-my-function",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your identity pool ARN>"
                }
            }
        }
    ]
}
```

5. [Configure function] (関数の設定) 画面で、関数の名前と説明を入力します。[Runtime] (ランタイム) を「Node.js」に設定したままにします。コードは、例では変更しないまま使用します。デフォルトの例では、同期中のデータに変更を加えません。コードは、Amazon Cognito Sync Trigger イベントが発生したことのみを記録します。[Handler] (ハンドラ) の名前は「index.handler」設定のままにしておきます。[Role] (ロール) については、AWS Lambda にアクセスするためのコード許可を付与する IAM ロールを選択します。ロールを変更する方法については、「IAM コンソール」を参照してください。[Advanced settings] (詳細設定) は変更しないまま使用します。[Next] (次へ) をクリックします。
6. [Review] (確認) 画面で詳細を確認し、[Create function] (関数の作成) をクリックします。次のページに新しい Lambda 関数が表示されます。

Lambda で適切な関数を作成したところで、この関数を Amazon Cognito Sync Trigger イベントのハンドラとして選択する必要があります。以下のステップで、このプロセスについて説明します。

コンソールのホームページから、以下の操作を行います。

1. Amazon Cognito イベントをセットアップする ID プールの名前をクリックします。ID プールのダッシュボードページが表示されます。
2. ダッシュボードページの右上にある [Manage Federated Identities] (フェデレーティッド ID の管理) をクリックします。[Manage Federated Identities] (フェデレーティッド ID の管理) ページが表示されます。
3. スクロールダウンし、[Cognito Events] (Cognito イベント) をクリックして展開します。
4. [Sync Trigger] ドロップダウンメニューで、Sync イベントが発生したときにトリガーする Lambda 関数を選択します。
5. [Save Changes] (変更の保存) をクリックします。

これで、データセットが同期されるたびに Lambda 関数が実行されるようになります。次のセクションでは、同期中に関数のデータの読み取りと修正を行う方法について説明します。

Sync trigger 用の Lambda 関数の記述

Sync Trigger は、サービスプロバイダインタフェースが使用するプログラミングパラダイムに従います。Amazon Cognito は、Lambda 関数に対して以下の JSON 形式の入力を提供します。

```
{
  "version": 2,
  "eventType": "SyncTrigger",
  "region": "us-east-1",
  "identityPoolId": "identityPoolId",
  "identityId": "identityId",
  "datasetName": "datasetName",
  "datasetRecords": {
    "SampleKey1": {
      "oldValue": "oldValue1",
      "newValue": "newValue1",
      "op": "replace"
    },
    "SampleKey2": {
      "oldValue": "oldValue2",
      "newValue": "newValue2",
      "op": "replace"
```

```
    },..
  }
}
```

Amazon Cognito では、関数の戻り値が入力と同じ形式であることが想定されています。

Sync Trigger イベント用の関数を記述するときは、次の点に注意してください。

- Amazon Cognito が、UpdateRecords の実行時に Lambda 関数を呼び出すと、関数は 5 秒以内に応答する必要があります。応答しない場合、Amazon Cognito Sync サービスが `LambdaSocketTimeoutException` 例外をスローします。このタイムアウト値を大きくすることはできません。
- `LambdaThrottledException` 例外を取得した場合は、同期オペレーションをもう一度実行して、レコードを更新します。
- Amazon Cognito は、データセットに存在するすべてのレコードを関数に提供します。
- アプリユーザーが更新するレコードには、op フィールドが `replace` として設定されています。削除されたレコードには、op フィールドが `remove` として設定されています。
- アプリユーザーがレコードを更新していない場合でも、任意のレコードを変更できます。
- datasetRecords を除くすべてのフィールドは読み取り専用です。それらを変更しないでください。これらのフィールドを変更すると、レコードを更新することはできません。
- レコードの値を変更するには、値を更新し、op を `replace` に設定します。
- レコードを削除するには、op を `remove` に設定するか、値を null に設定します。
- レコードを追加するには、datasetRecords 配列に新しいレコードを追加します。
- Amazon Cognito は、Amazon Cognito がレコードを更新するときに、レスポンス内の省略されたレコードを無視します。

サンプル Lambda 関数

以下の Lambda 関数のサンプルでは、データへのアクセス、変更、削除の方法を示しています。

```
console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {
```

```
        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' :
 'replace'};
        }

    }
    context.done(null, event);
};
```

Amazon Cognito コンソールの使用

Amazon Cognito コンソール を使用して、ユーザープールと ID プールを作成し、管理することができます。

このガイドでは、Amazon Cognito コンソールで一般的な Amazon Cognito ユーザープールタスクの step-by-step チュートリアルを提供します。

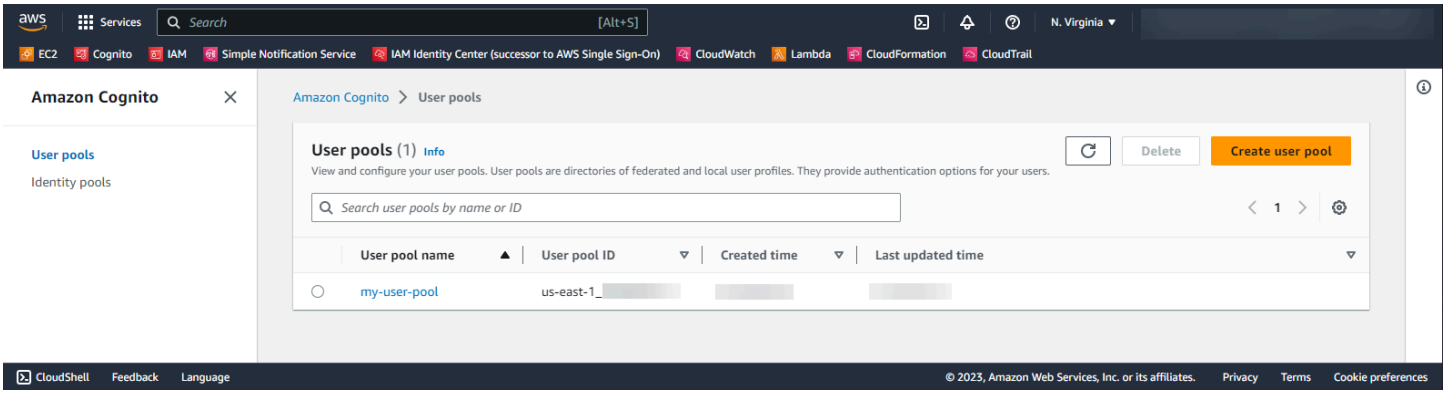
Amazon Cognito コンソールを使用するには

1. Amazon Cognito を使用するには、AWS アカウント にサインアップする必要があります。
2. Amazon Cognito コンソールに移動します。 AWS 認証情報の入力を求められる場合があります。
3. ユーザープールを作成または編集するには、左のナビゲーションペインから [User Pools] (ユーザープール) を選択します。

   詳細については、「ユーザープールの開始方法」を参照してください。
4. ID プールを作成または編集するには、[ID プール] を選択します。Amazon Cognito アイデンティティプールの元のコンソールが表示されます。

   詳細については、「Amazon Cognito ID プールの開始方法」を参照してください。

Amazon Cognito コンソールは の一部であり AWS Management Console、アカウントと請求に関する情報を提供します。詳細については、「AWS Management Consoleの操作」を参照してください。



トピック

- ユーザープールコンソール

- アイデンティティプールコンソール

ユーザープールコンソール

Amazon Cognito コンソールのユーザープールビューで、リストからユーザープールを選択して詳細を表示します。詳細ビューの、コンソールの上部にある [ユーザープールの概要] には、ユーザープールの基本情報が表示されます。以下のタブでは、ユーザープールの設定を関連関数別に整理しています。

[ユーザー]

[ユーザー] タブには、CSV ファイルからのユーザーおよびユーザーインポートに関する情報が含まれています。このタブでは、ユーザーを追加、削除、編集できます。

リファレンス

- ユーザープール内のユーザーを管理する
- CSV ファイルからユーザープールへのユーザーのインポート

グループ

[グループ] タブには、ユーザーグループに関する情報が含まれています。グループのメンバーシップを追加、変更、変更したり、アイデンティティプールを統合するためにグループに関連付けられている IAM ロールを変更したりできます。

リファレンス

- ユーザープールにグループを追加する

サインインエクスペリエンス

[サインインエクスペリエンス] タブには、ユーザーがユーザープールにサインインする方法に関する情報が含まれています。このタブには、サードパーティ ID プロバイダー、ユーザー名オプション、パスワードポリシー、多要素認証 (MFA) 設定、パスワードを忘れたときの動作、デバイスの記憶などが表示されています。ID プロバイダーを追加および変更したり、ユーザープールの全体的なサインイン動作を変更したりできます。

リファレンス

- サードパーティー経由のユーザープールへのサインインの追加
- ログイン属性のカスタマイズ

- ユーザープールのパスワードの追加要件
- ユーザープールに MFA を追加します
- ユーザーアカウントの復旧
- ユーザープール内のユーザーデバイスの使用

サインインエクスペリエンス

[サインアップエクスペリエンス] タブには、セルフサービスサインアップ、必須属性、電話番号とメールアドレスの確認、カスタム属性に関する情報が含まれています。

リファレンス

- ユーザーアカウントのサインアップと確認
- ユーザープール属性
- サインアップ時に連絡先情報を検証する

メッセージング

[メッセージング] タブには、ユーザーへの e メールや SMS メッセージの送信に使用する AWS のサービス 、および送信に使うメッセージ形式に関する情報が含まれています。

リファレンス

- Amazon Cognito ユーザープールの E メール設定
- Amazon Cognito ユーザープール用の SMS メッセージ設定
- SMS と E メール検証メッセージおよびユーザー招待メッセージの設定

アプリ統合

[アプリ統合] タブには、ユーザープールアプリクライアント、ユーザープールサービスエンドポイントに割り当てるドメイン、API リソースサーバー、ホストされた UI、および高度なセキュリティに関する情報が含まれています。各アプリケーションクライアントをドリルダウンして以下を設定できます。

1. トークン設定
2. コールバック URL
3. 認証のフロー
4. 属性権限
5. アプリ固有の高度なセキュリティとホストされた UI 設定
6. Amazon Pinpoint 分析

リファレンス

- ユーザープールアプリクライアント
- Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用
- ユーザープールのドメインを設定する
- スコープ、M2M 、およびリソースサーバーによる API 認証
- ユーザープールにアドバンストセキュリティを追加する
- Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用

ユーザープールのプロパティ

ユーザープールのプロパティタブには、Lambda トリガー、 AWS WAF ウェブ ACL 保護、削除保護、リソースタグなど、ユーザーに直接関係しないユーザープール設定に関する情報が含まれています。

リファレンス

- Lambda トリガーを使用したユーザープールワークフローのカスタマイズ
- AWS WAF ウェブ ACL とユーザープールの関連付け
- ユーザープールの削除保護
- AWS リソースのタグ付け

アイデンティティプールコンソール

Amazon Cognito コンソールのアイデンティティプールビューで、リストからアイデンティティプールを選択して詳細を表示します。詳細ビューの、コンソールの上部にある [アイデンティティプールの概要] には、ユーザープールの基本情報が表示されます。以下のタブでは、ユーザープールの設定を関連関数別に整理しています。

ユーザー統計

[ユーザー統計] タブには、アイデンティティプールで ID を生成したユーザーに関する統計情報が表示されます。このタブでアイデンティティプール設定を行うことはできません。

ID ブラウザ

[ID ブラウザ] タブには、アイデンティティプールでユーザーが生成した個々の ID に関する情報が含まれています。ID は表示および削除できます。

リファレンス

- Amazon Cognito ID プールの開始方法

ユーザーアクセス

[ユーザーアクセス] タブには、アイデンティティプールにリンクした ID プロバイダー、開発者プロバイダー、ID に割り当てられたデフォルトの IAM ロール、および認証されていないゲストアクセス設定に関する情報が含まれています。各 ID プロバイダーをドリルダウンして以下を設定できます。

1. IAM ロール選択によるロールベースのアクセス制御
2. アクセス制御用の属性による属性ベースのアクセス制御

リファレンス

- ID プール外部 ID プロバイダー
- IAM ロール
- 認証された ID と認証されていない ID
- デベロッパーが認証したアイデンティティ (アイデンティティプール)
- ロールベースアクセスコントロールの使用
- アクセスコントロールへの属性の使用

ID プールのプロパティ

[ID プールのプロパティ] タブには、基本 (クラシック) 認証やリソースタグなど、さまざまなアイデンティティプール設定に関する情報が表示されます。

- ID プール (フェデレーティッドアイデンティティ) の認証フロー
- AWS リソースのタグ付け

Amazon Cognito のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ — クラウドで AWS サービスを実行するインフラストラクチャを保護する責任 AWS は AWS にあります。 AWS また、 では、安全に使用できるサービスも提供しています。コンプライアンスAWS プログラムコンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。Amazon Cognito に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Amazon Cognito の使用時に責任共有モデルを適用する方法を理解するために役立ち、セキュリティおよびコンプライアンス上の目的を満たすように Amazon Cognito を設定する方法を説明します。また、Amazon Cognito リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

内容

Amazon Cognito でのデータ保護

責任 AWS 共有モデル、Amazon Cognito (Amazon Cognito) でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての AWS クラウドを実行するグローバルインフラストラクチャを保護する責任があります。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用する AWS サービスのセキュリティ設定および管理タスクが含まれます。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。

データ保護の目的で、 AWS アカウントの認証情報を保護し、 AWS Identity and Access Management (IAM) を使用して個々のユーザーアカウントを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 サービス内のすべての AWS デフォルトのセキュリティコントロールを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これにより、Amazon S3 に保存される個人データの検出と保護が支援されます。

顧客のアカウント番号などの機密の識別情報は、[名前] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または AWS SDK を使用して Amazon Cognito AWS CLIまたは他の のサービスを使用する場合も同様です。 AWS SDKs Amazon Cognito、またはその他のサービスに入力されたデータは、いずれも診断ログへの包含のために取得される可能性があります。外部サーバーへの URL を指定するときは、そのサーバーへのリクエストを検証するための認証情報を URL に含めないでください。

データ暗号化

データ暗号化は、通常、保管時の暗号化と転送中の暗号化の 2 つのカテゴリに分類されます。

保管時の暗号化

Amazon Cognito 内のデータは、業界標準に従って保管時に暗号化されます。

転送時の暗号化

マネージドサービスである Amazon Cognito は AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon Cognito にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Cognito ユーザープールおよび ID プールには、IAM 認証、非認証、およびトークン認証による API オペレーションがあります。認証されていない API オペレーションやトークン認証された API オペレーションは、お客様 (アプリケーションのエンドユーザー) による使用を目的としています。認証されていない API オペレーションとトークン認証された API オペレーションは、保管中および転送中に暗号化されます。詳細については、「Amazon Cognito ユーザープールの認証済みおよび未認証の API オペレーション」を参照してください。

> **Note**
>
> Amazon Cognito は、コンテンツを内部で暗号化します。お客様提供のキーはサポートしていません。

Amazon Cognito 向けの Identity and access management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証するか (サインインさせるか)

と、誰に Amazon Cognito リソースの使用を許可するか (アクセス許可を付与するか) を制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon Cognito で行う作業によって異なります。

サービスユーザー – ユーザーが Amazon Cognito サービスを使用してジョブを実行する場合、必要な認証情報とアクセス許可は管理者が用意します。さらに多くの Amazon Cognito 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon Cognito の機能にアクセスできない場合は、「Amazon Cognito のアイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon Cognito リソースを担当しているユーザーには、通常、Amazon Cognito への完全なアクセス権限があります。サービスユーザーがアクセスする Amazon Cognito の機能とリソースを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。Amazon Cognito で IAM を利用する方法の詳細については、「Amazon Cognito で IAM が機能する仕組み」を参照してください。

IAM 管理者 – IAM 管理者は、通常、Amazon Cognito へのアクセスを管理するポリシーの作成方法について詳細情報が必要になります。IAM で使用できる Amazon Cognito の ID ベースのポリシー例については、「Amazon Cognito の ID ベースのポリシー例」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 にサインインする方法 AWS アカウントAWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWSにおける多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、『IAM ユーザーガイド』の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールが引き受けられ、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、『AWS IAM Identity Center ユーザーガイド』の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

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

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

IAM ロール

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

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

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

す。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの「 AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) AWS がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

Amazon Cognito で IAM が機能する仕組み

IAM を使用して Amazon Cognito へのアクセスを管理する前に、Amazon Cognito で利用できる IAM の機能を確認します。

Amazon Cognito で使用できる IAM の機能

IAM 機能	Amazon Cognito サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	Yes
ポリシー条件キー	Yes
ACL	No
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	Yes
プリンシパル権限	いいえ
サービスロール	あり
サービスリンクロール	はい

Amazon Cognito およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

Amazon Cognito の ID ベースのポリシー

アイデンティティベースポリシーをサポートする	Yes

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon Cognito の ID ベースのポリシー例

Amazon Cognito の ID ベースのポリシー例については、「Amazon Cognito の ID ベースのポリシー例」を参照してください。

Amazon Cognito 内のリソースベースのポリシー

リソースベースのポリシーのサポート	No

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon Cognito のポリシーアクション

ポリシーアクションに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon Cognito アクションのリストについては、「サービス認可リファレンス」の「Amazon Cognito ID で定義されるアクション」を参照してください。

Amazon Cognito のポリシーアクションでは、アクションの前に、次のプレフィックスを使用します。

```
cognito-identity
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
```

```
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

署名された API と署名されていない API

AWS 認証情報を使用して Amazon Cognito API リクエストに署名すると、 AWS Identity and Access Management (IAM) ポリシーでリクエストを制限できます。 AWS 認証情報で署名する必要がある API リクエストには、サーバー側での `AdminInitiateAuth` を使用したサインイン、および `UpdateUserPool` などの Amazon Cognito リソースを作成、表示、または変更するアクションが含まれます。署名付き API リクエストの詳細については、AWS 「 API リクエストの署名」を参照してください。

Amazon Cognito は、一般公開するアプリケーション用のコンシューマー ID 製品であるため、次の署名されていない API にアクセスできます。アプリケーションは、ユーザーと見込みユーザーに代わってこれらの API リクエストを行います。一部の API (`InitiateAuth` など) は、新しい認証セッションを開始するために、事前の承認を必要としません。一部の API (`VerifySoftwareToken` など) は、アクセストークンやセッションキーを認証に使用して、既存の認証済みセッションを持つユーザーの MFA 設定を完了します。署名されていない、承認済みの Amazon Cognito ユーザープール API は、Amazon Cognito API リファレンスで示しているように、リクエスト構文での `Session` または `AccessToken` パラメータをサポートしています。署名されていない Amazon Cognito アイデンティティ API は、Amazon Cognito フェデレーティッドアイデンティティ API リファレンスで示しているように、`IdentityId` パラメータをサポートしています。

Amazon Cognito ユーザープール API オペレーションの認証モデルとロールの詳細については、「Amazon Cognito ユーザープールの認証済みおよび未認証の API オペレーション」を参照してください。

Amazon Cognito アイデンティティプール API オペレーション

- `GetId`
- `GetOpenIdToken`
- `GetCredentialsForIdentity`
- `UnlinkIdentity`

Amazon Cognito ユーザープール API オペレーション

- `AssociateSoftwareToken`

- ChangePassword
- ConfirmDevice
- ConfirmForgotPassword
- ConfirmSignUp
- DeleteUser
- DeleteUserAttributes
- ForgetDevice
- ForgotPassword
- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- RevokeToken
- SetUserMFAPreference
- SetUserSettings
- SignUp
- UpdateAuthEventFeedback
- UpdateDeviceStatus
- UpdateUserAttributes
- VerifySoftwareToken
- VerifyUserAttribute

Amazon Cognito の ID ベースのポリシー例については、「Amazon Cognito の ID ベースのポリシー例」を参照してください。

Amazon Cognito のポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

`Resource` JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource`要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

操作のリスト化など、リソースレベルの許可をサポートしないアクションの場合は、ワイルドカード (*) を使用して、ステートメントがすべてのリソースに適用されることを示します。

```
"Resource": "*"
```

Amazon リソースネーム (ARN)

Amazon Cognito フェデレーティッド ID の ARN

Amazon Cognito ID プール (フェデレーティッド ID) では、以下の例にあるように、Amazon リソースネーム (ARN) 形式を使用して IAM ユーザーのアクセスを特定の ID プールに制限することができます。ARN の詳細については、「IAM 識別子」を参照してください。

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync の ARN

Amazon Cognito Sync では、カスタマーがアイデンティティプール ID、アイデンティティ ID、およびデータセット名でアクセスを制限することもできます。

ID プールで動作する API の場合、サービス名が `cognito-identity` ではなく `cognito-sync` である場合を除いて、ID プールの ARN 形式が Amazon Cognito フェデレーティッド ID のものと同じになります。

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

RegisterDevice などの単一のアイデンティティで動作する API については、以下の ARN 形式で個々のアイデンティティを参照できます。

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID
```

データセットで動作する API (例えば、UpdateRecords および ListRecords) の場合、次の ARN 形式を使用して個別のデータセットを参照できます。

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID/dataset/DATASET_NAME
```

Amazon Cognito ユーザープールの ARN

Amazon Cognito ユーザープールでは、以下の ARN 形式を使用して、ユーザーのアクセスを特定のユーザープールに制限することが可能です。

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

Amazon Cognito リソースのタイプと ARN のリストについては、「サービス認可リファレンス」の「Amazon Cognito ID で定義されるリソースタイプ」を参照してください。各リソースの ARN を指定できるアクションについては、「Amazon Cognito で定義されるアクション」を参照してください。

Amazon Cognito の ID ベースのポリシー例については、「Amazon Cognito の ID ベースのポリシー例」を参照してください。

Amazon Cognito のポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定するか、1 つの Condition 要素に複数のキーを指定すると、 AWS は AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Amazon Cognito の条件キーのリストについては、「サービス認可リファレンス」の「Amazon Cognito ID の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Cognito で定義されるアクション」を参照してください。

Amazon Cognito の ID ベースのポリシー例については、「Amazon Cognito の ID ベースのポリシー例」を参照してください。

Amazon Cognito のアクセスコントロールリスト (ACL)

ACL のサポート	No

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon Cognito の属性ベースのアクセス制御 (ABAC)

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初

の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon Cognito での一時的な認証情報の使用

一時的な認証情報のサポート	はい

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。これは、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することを推奨しています。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon Cognito のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) をサポート	いいえ

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon Cognito のサービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの「 AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

Amazon Cognito のサービスロールの詳細については、「プッシュ同期を有効にする」と「プッシュ同期」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、Amazon Cognito の機能が破損する可能性があります。Amazon Cognito が指示する場合以外は、サービスロールを編集しないでください。

Amazon Cognito のサービスにリンクされたロール

サービスリンクロールのサポート	はい

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

Amazon Cognito のサービスにリンクされたロールの作成または管理の詳細については、「Amazon Cognito のサービスリンクロールの使用」を参照してください。

Amazon Cognito の ID ベースのポリシー例

デフォルトでは、ユーザーおよびロールには Amazon Cognito リソースを作成または変更するアクセス許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

Amazon Cognito が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「Amazon Cognito のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon Cognito コンソールの使用
- 自分の権限の表示をユーザーに許可する
- 特定の ID プールにコンソールアクセスを制限する
- プール内のすべての ID に対して特定のデータセットへのアクセスを許可する

ポリシーのベストプラクティス

ID ベースのポリシーは、アカウント内で誰が Amazon Cognito リソースを作成、アクセス、または削除できるを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

> **Note**
>
> Amazon Cognito コンソールの元のバージョンと新しいバージョンでは、Amazon Cognito リソースを表示および変更するときの基本的な動作が異なります。条件 `aws:ViaAWSService` が true のときだけに `cognito-idp` サービスプレフィックスのアクションに許可を付与した場合、該当する IAM プリンシパルは、元のコンソールでは Amazon Cognito リソース

に対して有効かもしれませんが、新しいコンソールでは使用できません。Amazon Cognito コンソールで作業を行うには、IAM ポリシーの Amazon Cognito アクセス許可に対して `aws:ViaAWSService` 条件を設定しないでください。

Amazon Cognito コンソールの使用

Amazon Cognito コンソールにアクセスするには、最小限の許可のセットが必要です。これらのアクセス許可により、 の Amazon Cognito リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き Amazon Cognito コンソールを使用できるようにするには、エンティティに Amazon Cognito `ConsoleAccess`または `ReadOnly` AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

特定の ID プールにコンソールアクセスを制限する

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:ListIdentityPools"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:*"
      ],
      "Resource": "arn:aws:cognito-identity:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cognito-sync:*"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    }
  ]
}
```

プール内のすべての ID に対して特定のデータセットへのアクセスを許可する

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:ListRecords",
        "cognito-sync:UpdateRecords"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678/identity/*/dataset/UserProfile"
    }
  ]
}
```

Amazon Cognito のアイデンティティとアクセスのトラブルシューティング

以下の情報は、Amazon Cognito と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

- Amazon Cognito でアクションを実行する権限がない
- iam を実行する権限がありません。PassRole
- 管理者として Amazon Cognito へのアクセスを他のユーザーに許可したい
- AWS アカウント以外のユーザーに Amazon Cognito リソースへのアクセスを許可したい

Amazon Cognito でアクションを実行する権限がない

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なcognito-identity:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cognito-identity:GetWidget on resource: my-example-widget
```

この場合、cognito-identity:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、Amazon Cognito にロールを渡すことを許可するようにポリシーを更新する必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例では、marymajor という名前の IAM ユーザーがコンソールを使用して Amazon Cognito でアクションを実行しようとした際に、エラーが発生しています。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

管理者として Amazon Cognito へのアクセスを他のユーザーに許可したい

Amazon Cognito へのアクセスを他のユーザーに許可するには、アクセスを必要とするユーザーまたはアプリケーション用の IAM エンティティ (ユーザーまたはロール) を作成する必要があります。ユーザーまたはアプリケーションは、そのエンティティの認証情報を使用して AWSにアクセスします。次に、Amazon Cognito で適切なアクセス許可を付与するポリシーをエンティティにアタッチする必要があります。

すぐにスタートするには、「IAM ユーザーガイド」の「IAM が委任した初期のユーザーおよびグループの作成」を参照してください。

AWS アカウント以外のユーザーに Amazon Cognito リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon Cognito がこれらの機能をサポートしているかどうかについては、「Amazon Cognito で IAM が機能する仕組み」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、IAM ユーザーガイドの IAM ロールとリソースベースのポリシーとの相違点を参照してください。

Amazon Cognito のサービスリンクロールの使用

Amazon Cognito は AWS Identity and Access Management (IAM) サービスにリンクされたロール を使用します。サービスにリンクされたロールは、 がロールを AWS のサービス 引き受けることを許可する信頼ポリシーを持つ一意のタイプの IAM ロールです。サービスにリンクされたロールは Amazon Cognito によって事前定義されており、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスリンクロールでは、必要な許可を手動で追加する必要がないため、Amazon Cognito の設定が簡単になります。Amazon Cognito は、そのサービスリンクロールの許可を定義します。別途定義されている場合を除き、Amazon Cognito しかそのロールを引き受けることができません。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これは、Amazon Cognito リソースを保護します。リソースにアクセスするための許可を誤って削除できなくなるからです。

サービスにリンクされたロールをサポートする他のサービスを確認するには、「IAM と連携する AWS のサービス」を開き、サービスにリンクされたロール列で「はい」が表示されているサービスを探します。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

Amazon Cognito のサービスリンクロール許可

Amazon Cognito は、以下のサービスリンクロールを使用します。

- AWSServiceRoleForAmazonCognitoIdpEmailService — Amazon Cognito ユーザープールサービスが Amazon SES ID を使用して E メールを送信できるようにします。

- AWSServiceRoleForAmazonCognitoIdp — Amazon Cognito ユーザープールが Amazon Pinpoint プロジェクトのイベントを発行し、エンドポイントを設定できるようにします。

AWSServiceRoleForAmazonCognitoIdpEmailService

`AWSServiceRoleForAmazonCognitoIdpEmailService` サービスリンクロールは、ロールの引き受けについて以下のサービスを信頼します。

- `email.cognito-idp.amazonaws.com`

このロールの許可ポリシーは、Amazon Cognito が指定されたリソースで以下のアクションを実行できるようにします。

で許可されるアクション AWSServiceRoleForAmazonCognitoIdpEmailService：

- アクション: `ses:SendEmail` および `ses:SendRawEmail`
- リソース: *

このポリシーは、Amazon Cognito が指定されたリソースで以下のアクションを完了することを拒否します。

拒否されるアクション

- アクション: `ses:List*`
- リソース: *

これらの許可を使用すると、Amazon Cognito は、ユーザーへの E メールの送信に Amazon SES で検証済みの E メールアドレスのみを使用することができます。Amazon Cognito は、ユーザーがユーザープールのクライアントアプリで特定のアクション (サインアップやパスワードのリセットなど) を実行するときに、ユーザーに E メールを送信します。

IAM エンティティ (ユーザー、グループ、ロールなど) がサービスリンクロールの作成、編集、削除を行うことを許可する許可を設定する必要があります。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールのアクセス許可」を参照してください。

AWSServiceRoleForAmazonCognitoIdp

AWSServiceRoleForAmazonCognitoIdp サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `email.cognito-idp.amazonaws.com`

このロールの許可ポリシーは、Amazon Cognito が指定されたリソースで以下のアクションを実行できるようにします。

で許可されるアクション AWSServiceRoleForAmazonCognitoIdp

- アクション: `cognito-idp:Describe`

- リソース: *

この許可を使用すると、Amazon Cognito はユーザーに代わって `Describe` Amazon Cognito API オペレーションを呼び出すことができます。

> **Note**
>
> `createUserPoolClient` と `updateUserPoolClient` を使用して Amazon Cognito を Amazon Pinpoint に統合すると、リソースのアクセス許可がインラインポリシーとして SLR に追加されます。インラインポリシーは、`mobiletargeting:UpdateEndpoint` および `mobiletargeting:PutEvents` 許可を提供します。これらの許可は、Amazon Cognito が、Cognito に統合された Pinpoint プロジェクトのためにイベントを発行し、エンドポイントを設定できるようにします。

Amazon Cognito のサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。、、 AWS CLIまたは Amazon Cognito API で E メール配信を処理するために Amazon SES 設定を使用するようにユーザープールを設定する AWS Management Consoleと、Amazon Cognito によってサービスにリンクされたロールが作成されます。 Amazon Cognito

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。Amazon SES 設定を使用してメール配信を処理するようにユーザープールを設定すると、Amazon Cognito がサービスリンクロールをもう一度作成します。

Amazon Cognito がこのロールを作成する前に、ユーザープールのセットアップに使用する IAM 許可に `iam:CreateServiceLinkedRole` アクションを含める必要があります。IAM での許可の更新に関する詳細については、IAM ユーザーガイドの「IAM ユーザーの許可の変更」を参照してください。

Amazon Cognito のサービスリンクロールの編集

AmazonCognitoIdp またはサービス AmazonCognitoIdpEmailService にリンクされたロールは で編集できません AWS Identity and Access Management。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

Amazon Cognito のサービスリンクロールの削除

サービスリンクロールを必要とする機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。ロールを削除すると、Amazon Cognito が積極的にモニタリングまたは維持しているエンティティのみを保持します。 AmazonCognitoIdp または AmazonCognitoIdpEmailService サービスにリンクされたロールを削除する前に、ロールを使用するユーザープールごとに次のいずれかを実行する必要があります。

- ユーザープールを削除する。
- ユーザープールの E メール設定を更新して、デフォルトの E メール機能を使用する。デフォルト設定では、サービスリンクロールが使用されません。

ロールを使用するユーザープール AWS リージョン を使用して、各 で アクションを実行することを忘れないでください。

> **Note**
>
> リソースを削除しようとするときに Amazon Cognito サービスがロールを使用している場合は、削除が失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。

Amazon Cognito ユーザープールを削除する

1. にサインイン AWS Management Console し、 で Amazon Cognito コンソールを開きますhttps://console.aws.amazon.com/cognito。
2. [Manage User Pools] (ユーザープールの管理) をクリックします。
3. [Your User Pools] (ユーザープール) ページで、削除するユーザープールを選択します。
4. [Delete pool] (プールの削除) をクリックします。
5. [Delete user pool] (ユーザープールの削除) ウィンドウで「**`delete`**」と入力し、[Delete pool] (プールの削除) をクリックします。

デフォルトの E メール機能を使用するために Amazon Cognito ユーザープールを更新する

1. にサインイン AWS Management Console し、 で Amazon Cognito コンソールを開きますhttps://console.aws.amazon.com/cognito。

2. [Manage User Pools] (ユーザープールの管理) をクリックします。
3. [Your User Pools] (ユーザープール) ページで、更新するユーザープールを選択します。
4. 左側のナビゲーションメニューで [Message customizations] (メッセージのカスタマイズ) をクリックします。
5. [Do you want to send emails through your Amazon SES Configuration?] (Amazon SES の設定を通じて E メールを送信しますか?) で [No - Use Cognito (Default)] (いいえ - Cognito を使用します (デフォルト)) を選択します。
6. E メールアカウントオプションの設定を終了したら、[Save changes] (変更の保存) をクリックします。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、 AmazonCognitoIdp または AmazonCognitoIdpEmailService サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

Amazon Cognito サービスリンクロールがサポートされるリージョン

Amazon Cognito は、サービスが利用可能なすべての でサービスにリンク AWS リージョン されたロールをサポートします。詳細については、「AWS リージョン およびエンドポイント」を参照してください。

Amazon Cognito でのロギングとモニタリング

モニタリングは、Amazon Cognito およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。現在 Amazon Cognito では、組織と、その組織内で発生するアクティビティを監視することができるように、以下の AWS のサービス をサポートしています。

- AWS CloudTrail – を使用すると、Amazon Cognito コンソールから、および Amazon Cognito API オペレーションへのコード呼び出しから API コールをキャプチャ CloudTrail できます。例えば、ユーザーが認証すると、 はリクエストの IP アドレス、リクエスト者、リクエスト日時などの詳細を記録 CloudTrail できます。
- Amazon CloudWatch Logs – CloudWatch Logs を使用すると、ユーザーアクティビティの詳細なログをロググループに送信できます。例えば、ユーザーアクティビティの詳細ログを確認して、ユーザーへの E メールや SMS メッセージの配信のトラブルシューティングを行うことができます。

- Amazon CloudWatch メトリクス – CloudWatch メトリクスを使用すると、ほぼリアルタイムでイベントが発生した場合に、モニタリング、レポート、自動アクションを実行できます。例えば、提供されたメトリクスに CloudWatch ダッシュボードを作成して Amazon Cognito ユーザープールをモニタリングしたり、提供されたメトリクスに CloudWatch アラームを作成して、設定されたしきい値に違反したときに通知したりできます。
- Amazon CloudWatch Logs Insights – CloudWatch Logs Insights を使用すると、Amazon Cognito CloudTrail ログファイルをモニタリング CloudWatch するために にイベントを送信する CloudTrail ように を設定できます。

トピック

コストのモニタリング

Amazon Cognito では、使用量の以下のディメンションに対して料金が発生します。

- ユーザープールの月間アクティブユーザー (MAUs)
- OIDC または SAML フェデレーションでサインインしたユーザープール MAUs
- 高度なセキュリティ機能を備えたユーザープールMAUs
- アクティブなユーザープールアプリクライアントと、クライアント認証情報付与によるマシンツーマシン (M2M) 認証のリクエストボリューム
- 一部のカテゴリのユーザープール APIs のデフォルトクォータを超える購入使用量

さらに、E メールメッセージ、SMS メッセージ、Lambda トリガーなどのユーザープールの機能では、依存サービスにコストが発生する可能性があります。完全な概要については、Amazon Cognito の料金」を参照してください。

コストの表示と予測

AWS Billing and Cost Management コンソール で AWS コストを表示およびレポートできます。Amazon Cognito の最新の料金は、請求と支払いセクションで確認できます。請求 、サービス別の料金 で、 をフィルタリングCognitoして使用状況を表示します。詳細については、AWS Billing ユーザーガイドの「請求書の表示」を参照してください。

API リクエストレートをモニタリングするには、Service Quotas コンソールで使用率メトリクスを確認します。例えば、クライアント認証情報リクエストは ClientAuthentication 、リクエストのレートとして表示されます。請求書では、これらのリクエストは、それらを生成したアプリクライアントに関連付けられています。この情報により、マルチテナントアーキテクチャ のテナントに公平にコストを割り当てることができます。

一定期間の M2M リクエストの数を取得するには、分析のためにAWS CloudTrail イベントを CloudWatch Logs に送信することもできます。クライアント認証情報の付与 CloudTrail Token_POSTを使用してイベントをクエリします。次の CloudWatch Insights クエリはこの数を返します。

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

のコスト管理

Amazon Cognito は、ユーザー数、機能の使用状況、リクエスト量に基づいて請求します。Amazon Cognito のコストを管理するためのヒントを以下に示します。

非アクティブなユーザーをアクティブ化しない

ユーザーをアクティブにする一般的なオペレーションは、サインイン、サインアップ、パスワードのリセットです。詳細なリストについては、「」を参照してください月次のアクティブユーザー。Amazon Cognito は、非アクティブなユーザーを請求書にカウントしません。ユーザーをアクティブに設定する操作は避けてください。AdminGetUser API オペレーションの代わりに、ListUsers オペレーションを使用してユーザーをクエリします。非アクティブなユーザーに対して、ユーザープールオペレーションの大量の管理テストを実行しないでください。

フェデレーティッドユーザーのリンク

SAML 2.0 または OpenID Connect (OIDC) ID プロバイダーでサインインするユーザーは、ローカルユーザー よりもコストが高くなります。これらのユーザーをローカルユーザープロファイル にリンクできます。リンクされたユーザーは、フェデレーティッドユーザーに付属する属性とアクセスを使用して、ローカルユーザーとしてサインインできます。SAML または OIDC のユーザーが 1 か月以内に、リンクされたローカルアカウントでのみサインイン IdPs すると、ローカルユーザーとして課金されます。

リクエストレートの管理

ユーザープールがクォータの上限に近づいている場合は、ボリュームを処理するための追加容量の購入を検討してください。アプリケーション内のリクエストの量を減らすことができる場合があります。詳細については、「クォータ制限のリクエストレートを最適化する」を参照してください。

必要な場合にのみ新しいトークンをリクエストする

クライアント認証情報の付与によるマシンツーマシン (M2M) 認証は、大量のトークンリクエストに達する可能性があります。新しいトークンリクエストはそれぞれ、リクエストレートのクォータと請求書のサイズに影響します。コストを最適化するには、トークンの有効期限設定とトークン処理をアプリケーションの設計に含めます。

- アクセストークンをキャッシュして、アプリケーションが新しいトークンをリクエストしたときに、以前に発行されたトークンのキャッシュバージョンを受け取るようにします。この方法を実装すると、キャッシュプロキシは、以前に取得したトークンの有効期限を認識せずにアクセストークンをリクエストするアプリケーションに対するガードとして機能します。キャッシュトークンは、Lambda 関数や Docker コンテナなどの有効期間の短いマイクロサービスに最適です。
- トークンの有効期限を考慮したトークン処理メカニズムをアプリケーションに実装します。以前のトークンの有効期限が切れるまで、新しいトークンをリクエストしないでください。各アプリケーションの機密性と可用性のニーズを評価し、適切な有効期間でアクセストークンを発行するようにユーザープールアプリクライアントを設定します。カスタムトークンの有効期間は、認証情報のリクエストの頻度を永続的に管理できる存続期間の長い APIs とサーバーに最適です。

未使用のクライアント認証情報アプリケーションクライアントを削除する

M2M 認証は、トークンリクエストのレートと、クライアント認証情報を付与するアプリケーションクライアントの数という 2 つの要素に基づいて請求します。M2M 認証用のアプリケーションクライアントが使用されていない場合は、クライアント認証情報を発行するための認可を削除するか、削除します。アプリケーションクライアント設定の管理の詳細については、「」を参照してくださいユーザープールアプリクライアント。

高度なセキュリティを管理する

ユーザープールでアドバンストセキュリティ機能を設定すると、アドバンストセキュリティ請求レートがユーザープール内のすべての MAUs に適用されます。高度なセキュリティ機能を必要としないユーザーがいる場合は、別のユーザープールに分割します。

および Service Quotas でのクォータ CloudWatch と使用状況の追跡

Amazon Cognito ユーザープールをモニタリングできます。 CloudWatch Service Quotas また、Service Quotas で ID プールの使用状況をモニタリングすることもできます。 CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに処理します。では CloudWatch、特定のしきい値を監視し、それらのしきい値に達したときに通知を送信したりアクションを実行したりするアラームを設定できます。サービスクォータの CloudWatch アラームを作成するには、「アラームの作成 CloudWatch」を参照してください。Amazon Cognito のメトリクスは 5 分間隔で利用できます。の保持期間の詳細については CloudWatch、「Amazon CloudWatch FAQ」ページを参照してください。

Service Quotas を使用して、Amazon Cognito ユーザープールとアイデンティティプールのクォータの使用状況を表示および管理できます。Service Quotas コンソールには、サービスクォータの表示、サービスクォータ引き上げのリクエスト、および現在の使用状況の表示という 3 つの機能があります。最初の機能を使用してクォータを表示し、クォータが調整可能かどうかを確認できます。2 番目の機能を使用して、Service Quotas の引き上げをリクエストできます。最後の機能は、クォータの使用率の表示に使用できます。この機能を利用できるのは、アカウントがしばらくの間アクティブな状態であった場合のみです。Service Quotas コンソールでのクォータの表示に関する詳細については、「Viewing service quotas」を参照してください。

> **Note**
>
> Amazon Cognito のメトリクスは 5 分間隔で利用可能になります。の保持期間の詳細については CloudWatch、「Amazon CloudWatch FAQ」ページを参照してください。

クロスアカウントオブザーバビリティで CloudWatchモニタリングアカウントとして AWS アカウント 設定された にサインインしている場合は、そのモニタリングアカウントを使用してサービスクォータを視覚化し、そのモニタリングアカウントにリンクされているソースアカウントのメトリクスのアラームを設定できます。詳細については、CloudWatch 「クロスアカウントオブザーバビリティ」を参照してください。

トピック

- Amazon Cognito ユーザープールの追加アクティビティのログ記録
- Amazon Cognito ユーザープールのメトリクス
- Amazon Cognito ユーザープールのディメンション
- Service Quotas コンソールを使用してメトリクスを追跡する

- CloudWatch コンソールを使用してメトリクスを追跡する
- クォータの CloudWatch アラームを作成する

Amazon Cognito ユーザープールの追加アクティビティのログ記録

ユーザープールを設定して、追加のアクティビティの詳細なログを CloudWatch ロググループに送信できます。これらのログは のログよりも細かく AWS CloudTrail、ユーザープールのトラブルシューティングに役立ちます。この機能を有効にすると、Amazon Cognito からログを送信する先のロググループを選択できます。ユーザーアクティビティのログ記録は、ユーザープールが Amazon SNS と Amazon SES で配信した E メールや SMS メッセージのステータスを確認する場合に便利です。

現在、ユーザープールから配信できるのは、エラーレベルのユーザー通知ログのみです。

詳細なログ記録は、ユーザープールの以下のログ機能を置き換えたり変更したりするものではありません。

1. CloudTrail サインアップやサインインなどの日常的なユーザーアクティビティの ログ。
2. CloudWatch メトリクスを使用した大規模なユーザーアクティビティの分析。

これとは別に、ユーザーインポートジョブと Lambda トリガーからのログも CloudWatch Logs で確認できます。Amazon Cognito と Lambda は、詳細なアクティビティログ用に指定したロググループとは異なるロググループに、これらのログを保存します。

SetLogDeliveryConfiguration API リクエストで Amazon Cognito ユーザープール API を使用して詳細なアクティビティログを設定できます。GetLogDeliveryConfiguration API リクエストでユーザープールのログ記録設定を表示できます。

これらのリクエストは、次のアクセス許可を持つ AWS 認証情報で承認する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
```

```
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

ユーザープールのイベントの例を次に示します。このログスキーマは変更される可能性があります。一部のフィールドは NULL 値がログとして記録される場合があります。

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
```

```
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

Amazon Cognito からのログの配信はベストエフォートに基づきます。ユーザープールが配信するログのボリュームと CloudWatch Logs のサービスクォータは、ログの配信に影響を与える可能性があります。

CloudWatch ログ配信が有効になっている場合、ログ料金が適用されます。詳細については、「Amazon CloudWatch 料金表」の「販売されたログ」を参照してください。

サイズが 5,120 文字を超えるリソースポリシーを持つロググループにログを送信するには、`/aws/vendedlogs` で始まるパスでロググループを設定します。詳細については、「特定のサービスからのログ記録の有効化 AWS」を参照してください。

Amazon Cognito ユーザープールのメトリクス

以下の表には、Amazon Cognito ユーザープールに使用できるメトリクスがリストされています。Amazon Cognito の Amazon CloudWatch メトリクスの名前空間は `AWS/Cognito` です。詳細については、「Amazon CloudWatch ユーザーガイド」の「名前空間」を参照してください。

> **Note**
>
> 過去 2 週間に新しいデータポイントがなかったメトリクスは、コンソールに表示されません。また、コンソールの [All metrics] (すべてのメトリクス) タブにある検索ボックスにそれらのメトリクス名またはディメンション名を入力しても表示されません。さらに、list-metrics コマンドの結果にも返されません。これらのメトリクスを取得する最善の方法は、AWS CLI の `get-metric-data`または `get-metric-statistics` コマンドを使用することです。

メトリクス	説明
`SignUpSuccesses`	Amazon Cognito ユーザープールに対して正常に行われたユーザー登録リクエストの合計数を提供します。正常に行われたユーザー登録リクエストは 1 の値を生成し、失敗したリクエストは 0 の値を生成します。スロットリングされた

メトリクス	説明
	リクエストも失敗したリクエストと見なされるため、スロットリングされたリクエストも 0 の数を生成します。 正常に行われたユーザー登録リクエストの割合を確認するには、このメトリクスの Average 統計を使用します。ユーザー登録リクエストの合計数を計上するには、このメトリクスの Sample Count 統計を使用します。正常に行われたユーザー登録リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。失敗したユーザー登録リクエストの合計数を CloudWatch Mathカウントするには、式を使用して Sum 統計から Sample Count 統計を減算します。 このメトリクスは、各ユーザープールクライアントのユーザープールごとにパブリッシュされます。ユーザー登録が管理者によって実行された場合、メトリクスはユーザープールクライアントを Admin とした形でパブリッシュされます。 このメトリクスは、ユーザーのインポートおよびユーザーの移行には生成されません。 メトリクスディメンション: UserPool、UserPoolClient 単位: 数

メトリクス	説明
SignUpThrottles	Amazon Cognito ユーザープールに対して行われたユーザー登録リクエストで、スロットリングされたものの合計数を提供します。ユーザー登録リクエストがスロットリングされるたびに、1 の数が発行されます。 スロットリングされたユーザー登録リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。 このメトリクスは、各クライアントのユーザープールごとにパブリッシュされます。スロットリングされたリクエストが管理者によって行われたものである場合、メトリクスはユーザープールクライアントを Admin とした形でパブリッシュされます。 メトリクスディメンション: UserPool、UserPoolClient 単位: 数

メトリクス	説明
SignInSuccesses	Amazon Cognito ユーザープールに対して正常に行われたユーザー認証リクエストの合計数を提供します。認証トークンがユーザーに発行されると、ユーザー認証は正常に行われたとみなされます。認証が正常に行われると 1 の値が生成され、リクエストが失敗すると 0 の値が生成されます。スロットリングされたリクエストも失敗したリクエストと見なされるため、スロットリングされたリクエストも 0 の数を生成します。 正常に行われたユーザー認証リクエストの割合を調べるには、このメトリクスの Average 統計を使用します。ユーザー認証リクエストの合計数を計上するには、このメトリクスの Sample Count 統計を使用します。正常に行われたユーザー認証リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。失敗したユーザー認証リクエストの合計数を CloudWatch Mathカウントするには、式を使用して Sum 統計から Sample Count 統計を減算します。 このメトリクスは、各クライアントのユーザープールごとにパブリッシュされます。リクエストで無効なユーザープールクライアントが提供された場合、メトリクスの対応するユーザープールクライアント値には、リクエストで送信された実際の無効な値ではなく、固定値の Invalid が含まれます。 このメトリクスには Amazon Cognito トークンを更新するリクエストが含まれないことに注意してください。Refresh トークン統計の提供には、別のメトリクスがあります。

メトリクス	説明
	メトリクスディメンション: UserPool、UserPoolClient 単位: 数
SignInThrottles	Amazon Cognito ユーザープールに対して行われたユーザー認証リクエストで、スロットリングされたものの合計数を提供します。認証リクエストがスロットリングされるたびに、1 の数が発行されます。 スロットリングされたユーザー認証リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。 このメトリクスは、各クライアントのユーザープールごとにパブリッシュされます。リクエストで無効なユーザープールクライアントが提供された場合、メトリクスの対応するユーザープールクライアント値には、リクエストで送信された実際の無効な値ではなく、固定値の Invalid が含まれます。 このメトリクスに Amazon Cognito トークンを更新するリクエストは含まれません。Refresh トークン統計の提供には、別のメトリクスがあります。 メトリクスディメンション: UserPool、UserPoolClient 単位: 数

メトリクス	説明
TokenRefreshSuccesses	Amazon Cognito ユーザープールに対して行われた Amazon Cognito を更新するリクエストで、正常に行われたものの合計数を提供します。Amazon Cognito トークンの更新リクエストが正常に行われると 1 の値が生成され、失敗したリクエストには 0 の値が生成されます。スロットリングされたリクエストも失敗したリクエストと見なされるため、スロットリングされたリクエストも 0 の数を生成します。 正常に行われた Amazon Cognito トークンの更新リクエストの割合を調べるには、このメトリクスの Average 統計を使用します。Amazon Cognito トークンを更新するリクエストの合計数を計上するには、このメトリクスの Sample Count 統計を使用します。正常に行われた Amazon Cognito トークンの更新リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。Amazon Cognito トークンの更新に失敗したリクエストの合計数をカウントするには、 Math式を使用して CloudWatch Sum 統計から Sample Count 統計を減算します。 このメトリクスは、ユーザープールクライアントごとにパブリッシュされます。リクエストに無効なユーザープールクライアントがある場合、ユーザープールクライアントの値には固定値の Invalid が含まれます。 メトリクスディメンション: UserPool、UserPoolClient 単位: 数

メトリクス	説明
TokenRefreshThrottles	Amazon Cognito ユーザープールに対して行われた Amazon Cognito トークンを更新するリクエストで、スロットリングされたものの合計数を提供します。Amazon Cognito トークンの更新リクエストがスロットリングされるたびに、1 の数が発行されます。 Amazon Cognito トークンのスロットリングされた更新リクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。 このメトリクスは、各クライアントのユーザープールごとにパブリッシュされます。リクエストで無効なユーザープールクライアントが提供された場合、メトリクスの対応するユーザープールクライアント値には、リクエストで送信された実際の無効な値ではなく、固定値の Invalid が含まれます。 メトリクスディメンション: UserPool、UserPoolClient 単位: 数

メトリクス	説明
FederationSuccesses	Amazon Cognito ユーザープールに対して正常に行われた ID フェデレーションリクエストの合計数を提供します。ID フェデレーションは、Amazon Cognito がユーザーに認証トークンを発行した時点で成功とみなされます。ID フェデレーションリクエストが正常に行われると 1 の値が生成され、失敗すると 0 の値が生成されます。スロットリングされたリクエストや、認証コードは生成されるがトークンは生成されないリクエストでは、値が 0 になります。 正常に行われた ID フェデレーションリクエストの割合を調べるには、このメトリクスの Average 統計を使用します。ID フェデレーションリクエストの合計数を計上するには、このメトリクスの Sample Count 統計を使用します。正常に行われた ID フェデレーションリクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。失敗した ID フェデレーションリクエストの合計数を CloudWatch Mathカウントするには、 式を使用して Sum 統計から Sample Count 統計を減算します。 メトリクスディメンション:UserPool、UserPoolClient 、IdentityProvider 単位: 数

メトリクス	説明
FederationThrottles	Amazon Cognito ユーザープールに対して行われた ID フェデレーションリクエストで、スロットリングされたものの合計数を提供します。ID フェデレーションリクエストがスロットリングされるたびに、1 の数が発行されます。 スロットリングされた ID フェデレーションリクエストの合計数を計上するには、このメトリクスの Sum 統計を使用します。 メトリクスディメンション:UserPool、UserPoolClient 、IdentityProvider 単位: 数
CallCount	顧客が実行したコールで、カテゴリに関連するものの合計数を提供します。このメトリクスには、スロットリングコール、失敗したコール、正常に行われたコールなどのすべてのコールが含まれます。 このメトリクスは [Usage] (使用状況) の nameSpace で利用できます。 カテゴリクォータは、 AWS アカウントとリージョン内のすべてのユーザープールにまたがるアカウントごとに適用されます。 カテゴリ内のコールの合計数は、このメトリクスの Sum 統計を使用して計上できます。 メトリクスディメンション: Type、Resource、Class 単位: 数

メトリクス	説明
ThrottleCount	スロットルされたコールで、カテゴリに関連するものの合計数を提供します。 このメトリクスは [Usage] (使用状況) の nameSpace で利用できます。 このメトリクスは、アカウントレベルでパブリッシュされます。 カテゴリ内のコールの合計数は、このメトリクスの Sum 統計を使用して計上できます。 メトリクスディメンション: Type、Resource、Class 単位: 数

Amazon Cognito ユーザープールのディメンション

Amazon Cognito によってパブリッシュされた使用状況メトリクスの絞り込みには、以下のディメンションが使用されます。ディメンションは、CallCount および ThrottleCount メトリクスのみに適用されます。

ディメンション	説明
Service	リソースを含む AWS サービスの名前。Amazon Cognito の使用状況メトリクスの場合、このディメンションの値は Cognito user pool です。
Type	報告されるエンティティのタイプ。Amazon Cognito の使用状況メトリクスに対する有効な値は API のみです。
Resource	実行中のリソースのタイプ。有効な値はカテゴリ名のみです。

ディメンション	説明
Class	追跡されているリソースのクラス。Amazon Cognito は class ディメンションを使用しません。

Service Quotas コンソールを使用してメトリクスを追跡する

Service Quotas を使用して、一元化された場所から Amazon Cognito ユーザープールおよびアイデンティティプールのクォータを表示および管理できます。Service Quotas コンソールを使用して、特定のクォータに関する詳細の表示、クォータ使用率のモニタリング、クォータ引き上げのリクエストを行うことができます。一部のクォータタイプでは、クォータの使用率を追跡するアラームを作成できます CloudWatch。追跡できる Amazon Cognito メトリクスの詳細については、「クォータの使用状況を追跡する」を参照してください。

Amazon Cognito ユーザープールおよびアイデンティティプールのサービスクォータ使用率を表示するには、次の手順を実行します。

1. Service Quotas コンソール を開きます。
2. ナビゲーションペインで、[AWS services] (AWS のサービス) を選択します。
3. [AWS のサービス] リストで、[Amazon Cognito ユーザープール] または [Amazon Cognito フェデレーティッド ID] を検索して選択します。[Service Quota] (サービスクォータ) ページが表示されます。
4. CloudWatch モニタリングをサポートするクォータを選択します。例えば、Amazon Cognito ユーザープールで [`Rate of UserAuthentication requests`] を選択します。
5. [Monitoring] (モニタリング) までスクロールダウンします。このセクションは、 CloudWatch モニタリングをサポートするクォータに対してのみ表示されます。
6. [Monitoring] (モニタリング) では、現在のサービスクォータ使用率をグラフに表示できます。
7. [Monitoring] (モニタリング) で、1 時間、3 時間、12 時間、1 日、3 日、または 1 週間を選択します。
8. グラフ内の任意の領域を選択して、サービスクォータ使用率の割合を表示します。ここから、グラフをダッシュボードに追加するか、アクションメニューを使用してメトリクス で表示 を選択すると、 CloudWatch コンソールの関連するメトリクスが表示されます。

CloudWatch コンソールを使用してメトリクスを追跡する

を使用して Amazon Cognito ユーザープールメトリクスを追跡および収集できます CloudWatch。CloudWatch ダッシュボードには、使用するすべての AWS サービスに関するメトリクスが表示されます。を使用して CloudWatch メトリクスアラームを作成できます。アラームは、通知を送信する、またはモニタリングしている特定のリソースに変更を行うように設定できます。でサービスクォータメトリクスを表示するには CloudWatch、次のステップを実行します。

1. CloudWatch コンソールを開きます。
2. ナビゲーションペインで メトリクスを選択します。
3. [All metrics] (すべてのメトリクス) でメトリクスとディメンションを選択します。
4. メトリクスの横にあるチェックボックスをオンにします。グラフにメトリクスが表示されます。

> **Note**
>
> 過去 2 週間に新しいデータポイントがなかったメトリクスは、コンソールに表示されません。また、コンソールの [All metrics] (すべてのメトリクス) タブにある検索ボックスにそれらのメトリクス名またはディメンション名を入力しても表示されず、list-metrics コマンドの結果にも返されません。これらのメトリクスを取得する最適な方法は、 AWS CLI で `get-metric-data` または `get-metric-statistics` コマンドを使用することです。

クォータの CloudWatch アラームを作成する

Amazon Cognito は、 `CallCount`および `ThrottleCount` API AWS のサービスクォータに対応する CloudWatch 使用状況メトリクスを提供します。 APIs での使用状況の追跡の詳細については、CloudWatch「」を参照してくださいクォータの使用状況を追跡する。

Service Quotas コンソールで、使用量がサービスクォータに近づいた場合に警告するアラームを作成できます。Service Quotas コンソールを使用して CloudWatch アラームを設定する方法については、Service Quotas CloudWatch 」を参照してください。 Service Quotas

を使用した Amazon Cognito API コールのログ記録 AWS CloudTrail

Amazon Cognito は AWS CloudTrail、Amazon Cognito のユーザー、ロール、または AWS サービスによって実行されたアクションを記録するサービスである と統合されています。 は、Amazon Cognito コンソールからの呼び出しや Amazon Cognito API オペレーションへのコード呼び出しを含

む、Amazon Cognito の API Amazon Cognito コールのサブセットをイベントとして CloudTrail キャプチャします。 Amazon Cognito 証跡を作成する場合は、Amazon Cognito の CloudTrail イベントなど、Amazon S3 バケットにイベントを配信することを選択できます。 Amazon S3 証跡を設定しない場合でも、 CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。で収集された情報を使用して CloudTrail、Amazon Cognito に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の設定とアクティブ化の方法など CloudTrail、 の詳細については、AWS CloudTrail 「 ユーザーガイド」を参照してください。

特定の CloudTrail イベントに対して Amazon CloudWatch アラームを作成することもできます。例えば、ID プールの設定が変更された場合にアラームをトリガー CloudWatch するように を設定できます。詳細については、 CloudTrail 「イベントの CloudWatch アラームの作成: 例」を参照してください。

トピック

- の Amazon Cognito 情報 CloudTrail
- Amazon Cognito のサインインイベントの概要
- Amazon Logs Insights を使用した Amazon Cognito CloudTrail CloudWatch イベントの分析

の Amazon Cognito 情報 CloudTrail

CloudTrail を作成すると、 がオンになります AWS アカウント。Amazon Cognito でサポートされているイベントアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴 を含む CloudTrail イベントの表示」を参照してください。

Amazon Cognito のイベントなど、 AWS アカウントのイベントの継続的な記録については、証跡を作成します。 CloudTrail 証跡はログファイルを Amazon S3 バケットに配信します。デフォルトでは、コンソールで証跡を作成すると、すべての リージョンに証跡が適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントを ログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、以下をご覧ください。

- 証跡を作成するための概要
- CloudTrail がサポートするサービスと統合

- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからのログファイルの受信 CloudTrail

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity 要素」を参照してください。

の機密データ AWS CloudTrail

ユーザープールと ID プールはユーザーデータを処理するため、Amazon Cognito は CloudTrail イベント内の一部のプライベートフィールドを値 で隠しますHIDDEN_FOR_SECURITY_REASONS。Amazon Cognito がイベントに設定しないフィールドの例については、「Amazon Cognito のサインインイベントの概要」を参照してください。Amazon Cognito は、パスワードやトークンなど、一般的にユーザー情報を含む一部のフィールドだけを非表示にします。Amazon Cognito は、API リクエスト内の非公開フィールドに入力された個人識別情報の自動検出やマスキングを行いません。

Amazon Cognito ユーザープール

Amazon Cognito は、ユーザープールアクションページにリストされているすべてのアクションの口グ記録を CloudTrail 、ログファイルのイベントとしてサポートします。Amazon Cognito は、ユーザープールイベントを管理イベント CloudTrail として に記録します。

Amazon Cognito ユーザープール CloudTrail エントリの `eventType`フィールドは、アプリが Amazon Cognito ユーザープール API または OpenID Connect、SAML 2.0、またはホストされた UI のリソースを提供するエンドポイントにリクエストを行ったかどうかを示します。API リクエストには `AwsApiCall` の `eventType` があり、エンドポイントリクエストには `AwsServiceEvent` の `eventType` があります。

Amazon Cognito は、以下のホストされた UI リクエストを のイベントとしてホストされた UI に記録します CloudTrail。

でのホストされた UI オペレーション CloudTrail

操作	説明
Login_GET , CognitoAuthentication	ユーザーは ログインエンドポイント に認証情報を表示または送信します。
OAuth2_Authorize_GET , Beta_Authorize_GET	ユーザーは 認可エンドポイント を表示します。
OAuth2Response_GET , OAuth2Response_POST	ユーザーは IdP トークンを /oauth2/idpresponse エンドポイントに送信します。
SAML2Response_POST , Beta_SAML2Response_POST	ユーザーは IdP SAML アサーションを /saml2/idpresponse エンドポイントに送信します。
Login_OIDC_SAML_POST	ユーザーは、ログインエンドポイント でユーザー名を入力し、IdP identifier と一致させます。
Token_POST , Beta_Token_POST	ユーザーは、認証コードを トークンエンドポイント に送信します。
Signup_GET , Signup_POST	ユーザーは、サインアップ情報を /signup エンドポイントに送信します。
Confirm_GET , Confirm_POST	ユーザーは、ホストされた UI で確認コードを送信します。
ResendCode_POST	ユーザーは、ホストされた UI で確認コードの再送リクエストを送信します。
ForgotPassword_GET , ForgotPassword_POST	ユーザーは、パスワードのリセットリクエストを /forgotPassword エンドポイントに送信します。

操作	説明
ConfirmForgotPassword_GET , ConfirmForgotPassword_POST	ユーザーは、ForgotPassword リクエストを確認するコードを /confirmForgotPassword エンドポイントに送信します。
ResetPassword_GET , ResetPassword_POST	ユーザーは、ホストされた UI で新しいパスワードを送信します。
Mfa_GET, Mfa_POST	ユーザーは、ホストされた UI で多要素認証 (MFA) コードを送信します。
MfaOption_GET , MfaOption_POST	ユーザーは、ホストされた UI で MFA の好みの方法を選択します。
MfaRegister_GET , MfaRegister_POST	ユーザーは、MFA の登録時にホストされた UI で多要素認証 (MFA) コードを送信します。
Logout	ユーザーは、/logout エンドポイントでサインアウトします。
SAML2Logout_POST	ユーザーは、/saml2/logout エンドポイントでサインアウトします。
Error_GET	ユーザーは、ホストされた UI でエラーページを表示します。
UserInfo_GET , UserInfo_POST	ユーザーまたは IdP は、UserInfo エンドポイント と情報を交換します。
Confirm_With_Link_GET	ユーザーは、Amazon Cognito が E メールメッセージで送信したリンクに基づいて確認を送信します。
Event_Feedback_GET	ユーザーは、高度なセキュリティ機能イベントに関するフィードバックを Amazon Cognito に送信します。

> **Note**
>
> Amazon Cognito はUserSub、ユーザーに固有のリクエストのUserName CloudTrail ログには記録しますが、記録しません。ListUsers API を呼び出し、sub のフィルターを使用することで、所定の UserSub のユーザーを見つけることができます。

Amazon Cognito アイデンティティプール

データイベント

Amazon Cognito は、次の Amazon Cognito ID イベントをデータイベント CloudTrail として に記録します。 データイベントは、デフォルトでログ記録 CloudTrail されない大量のデータプレーン API オペレーションです。追加の変更がイベントデータに適用されます。

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

これらの API オペレーションの CloudTrail ログを生成するには、証跡でデータイベントをアクティブ化し、Cognito ID プールのイベントセレクタを選択する必要があります。詳細については、「AWS CloudTrail ユーザーガイド」の「証跡のデータイベントの記録」を参照してください。

次の CLI コマンドを使用して、ID プールのイベントセレクターをトレイルに追加することもできます。

```
aws cloudtrail put-event-selectors --trail-name <trail name> --advanced-event-selectors
 \
"{\
   \"Name\": \"Cognito Selector\",\
   \"FieldSelectors\": [\
      {\
         \"Field\": \"eventCategory\",\
         \"Equals\": [\
            \"Data\"\
         ]\
      },\
```

```
      {\
         \"Field\": \"resources.type\",\
         \"Equals\": [\
            \"AWS::Cognito::IdentityPool\"\
         ]\
      }\
   ]\
}"
```

管理イベント

Amazon Cognito は、残りの Amazon Cognito ID プール API オペレーションを管理イベント .logs 管理イベント API オペレーションとしてデフォルトでログに記録します。 CloudTrail

Amazon Cognito が に記録する Amazon Cognito ID プール API オペレーションのリストについては CloudTrail、Amazon Cognito ID プール API リファレンス」を参照してください。

Amazon Cognito Sync

Amazon Cognito は、すべての Amazon Cognito Sync API オペレーションを管理イベントとしてログ記録します。Amazon Cognito が に記録する Amazon Cognito Sync API オペレーションのリストについては CloudTrail、Amazon Cognito Sync API リファレンス」を参照してください。

Amazon Cognito のサインインイベントの概要

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できます。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

トピック

- ホストされた UI サインアップの CloudTrail イベント例
- SAML リクエストの CloudTrail イベント例
- トークンエンドポイントへのリクエストの CloudTrail イベント例
- の CloudTrail イベントの例 CreateIdentityPool
- の CloudTrail イベントの例 GetCredentialsForIdentity
- の CloudTrail イベントの例 GetId
- の CloudTrail イベントの例 GetOpenIdToken

- の CloudTrail イベントの例 GetOpenIdTokenForDeveloperIdentity
- の CloudTrail イベントの例 UnlinkIdentity

ホストされた UI サインアップの CloudTrail イベント例

次の CloudTrail イベント例は、ユーザーがホストされた UI を通じてサインアップしたときに Amazon Cognito が記録する情報を示しています。

Amazon Cognito は、新しいユーザーがアプリのサインインページに移動すると、次のイベントをログに記録します。


```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-04-06T05:38:12Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Login_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "errorCode": "",
    "errorMessage": "",
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200.0
        },
        "requestParameters":
        {
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "response_type":
            [
                "token"
            ],
            "client_id":
```

```
            [
                "1example23456789"
            ]
        }
    },
    "eventID": "382ae09a-151d-4116-8f2b-6ac0a804a38c",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito は、新しいユーザーがアプリのサインインページで [Sign up] を選択すると、次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
```

```
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito は、新しいユーザーがユーザー名を選択し、E メールアドレスを入力し、アプリのサインインページからパスワードを選択すると、次のイベントをログに記録します。Amazon Cognito は、ユーザーの ID に関する識別情報を に記録しません CloudTrail。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
```

```
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
```

```
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito は、新規ユーザーがサインアップ後にホストされた UI のユーザー確認ページにアクセスすると、次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
```

```
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito は、ホストされた UI のユーザー確認ページで、Amazon Cognito が送信したコードを E メールメッセージでユーザーが入力したときに、次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
```

```
        {
            "status": 302
        },
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
```

```
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

SAML リクエストの CloudTrail イベント例

Amazon Cognito は、SAML IdP で認証されたユーザーが SAML アサーションを `/saml2/idpresponse` エンドポイントに送信すると、次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
```

```
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

トークンエンドポイントへのリクエストの CloudTrail イベント例

次の例は、トークンエンドポイント へのリクエストからのイベントの例です。

Amazon Cognito は、認証して認証コードを受け取ったユーザーがコードを /oauth2/token エンドポイントに送信すると、以下のイベントを記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
```

```
    },
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
```

```
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

バックエンドシステムが /oauth2/token エンドポイントにアクセストークンの client_credentials リクエストを送信すると、Amazon Cognito は次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
```

```
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito は、アプリが更新トークンを新しい ID に交換し、/oauth2/token エンドポイントとトークンにアクセスすると、次のイベントをログに記録します。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
```

```
            "grant_type":
            [
                "refresh_token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

の CloudTrail イベントの例 CreateIdentityPool

次の例は、CreateIdentityPool アクションに対するリクエストのログエントリを示しています。リクエストは Alice という IAM ユーザーによって行われました。

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "USER_AGENT",
    "requestParameters": {
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

の CloudTrail イベントの例 GetCredentialsForIdentity

次の例は、GetCredentialsForIdentity アクションに対するリクエストのログエントリを示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-credentials-for-identity",
    "requestParameters": {
        "logins": {
```

```
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

の CloudTrail イベントの例 GetId

次の例は、GetId アクションに対するリクエストのログエントリを示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-id",
```

```
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

の CloudTrail イベントの例 GetOpenIdToken

次の例は、GetOpenIdToken アクションに対するリクエストのログエントリを示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
```

```
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

の CloudTrail イベントの例 GetOpenIdTokenForDeveloperIdentity

次の例は、GetOpenIdTokenForDeveloperIdentity アクションに対するリクエストのログエントリを示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
```

```
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

の CloudTrail イベントの例 UnlinkIdentity

次の例は、UnlinkIdentity アクションに対するリクエストのログエントリを示しています。

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```


Amazon Logs Insights を使用した Amazon Cognito CloudTrail CloudWatch イベントの分析

Amazon Logs Insights を使用して、Amazon Cognito CloudTrail CloudWatch イベントを検索および分析できます。ログにイベントを送信するように証跡を設定すると、 CloudWatch CloudTrail は証跡設定に一致するイベントのみを送信します。

Amazon Cognito CloudTrail イベントをクエリまたは調査するには、 CloudTrail コンソールで証跡設定で 管理イベント オプションを選択し、 AWS リソースで実行された管理オペレーションをモニタ

リングできるようにします。アカウント内でのエラー、異常なアクティビティ、または異常なユーザー動作を特定したい場合は、オプションで追跡設定の [Insights イベント] オプションを選択することができます。

サンプル Amazon Cognito クエリ

Amazon CloudWatch コンソールでは、次のクエリを使用できます。

一般的なクエリ

最近追加された 25 件のログイベントを検索します。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

例外を含む、最近追加された 25 件のログイベントのリストを取得します。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

例外とエラーのクエリ

Amazon Cognito ユーザープール sub と共に、最近追加されたエラーコード NotAuthorizedException を伴う 25 件のログイベントを検索します。

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
```

eventName と対応する sourceIPAddress を持つレコードの数を検索します。

```
filter eventSource = "cognito-idp.amazonaws.com"
| stats count(*) by sourceIPAddress, eventName
```

NotAuthorizedException エラーをトリガーした上位 25 の IP アドレスを検索します。

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
```

```
| sort count desc | limit 25
```

ForgotPassword API を呼び出した上位 25 の IP アドレスを検索します。

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

Amazon Cognito のコンプライアンス検証

サードパーティーの監査者は、複数の コンプライアンスプログラムの一環として Amazon Cognito のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポート AWS Artifactのダウンロード」の」を参照してください。

Amazon Cognito の使用時におけるユーザーのコンプライアンス責任は、データの機密性、会社のコンプライアンス目的、および適用される法律と規制に応じて異なります。 AWS は、コンプライアンスに役立つ以下のリソースを提供しています。

- 「セキュリティ & コンプライアンス クイックリファレンスガイド」– これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、 AWSでセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。
- HIPAA セキュリティとコンプライアンスの設計ホワイトペーパー — このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価 AWS Config」 – は、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Cognito の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

トピック

- リージョンデータに関する考慮事項

リージョンデータに関する考慮事項

Amazon Cognito ユーザープールはそれぞれ 1 つの AWS リージョンで作成され、そのリージョンにのみユーザープロファイルデータを保存します。ユーザープールは、オプション機能の設定方法に応じて、ユーザーデータを別の AWS リージョンに送信できます。

- デフォルトの`no-reply@verificationemail.com` E メールアドレス設定は Amazon Cognito ユーザープールでの E メールアドレスのルーティング検証に使用されます。E メールは関連付けられているユーザープールと同じリージョン経由でルーティングされます。
- Amazon Cognito ユーザープールで Amazon Simple Email Service (Amazon SES) を設定するために別の E メールアドレスが使用されている場合、その E メールアドレスは Amazon SES の E メールアドレスに関連付けられた AWS リージョンを介してルーティングされます。
- Amazon Cognito ユーザープールからの SMS メッセージは、「E メールまたは電話による検証の設定」で特に指定されていない限り、同じリージョンの Amazon SNS 経由で送信されます。
- Amazon Cognito ユーザープールで Amazon Pinpoint 分析が使用される場合、イベントデータは米国東部 (バージニア北部) リージョンにルーティングされます。

> **Note**
>
> Amazon Pinpoint は、北米、欧州、アジア、オセアニアの複数の AWS リージョンで利用できます。Amazon Pinpoint リージョンには、Amazon Pinpoint API が含まれています。Amazon Pinpoint リージョンが Amazon Cognito でサポートされている場合、Amazon Cognito は同じ Amazon Pinpoint リージョン内の Amazon Pinpoint プロジェクトにイベントを送信します。リージョンが Amazon Pinpoint でサポートされていない場合、Amazon Cognito は us-east-1 でのイベントの送信のみをサポートします。Amazon Pinpoint の詳細なリージョン情報については、「Amazon Pinpoint エンドポイントとクォータ」および「Amazon Cognito ユーザープールを使用して Amazon Pinpoint 分析を使用する」を参照してください。

Amazon Cognito のインフラストラクチャセキュリティ

マネージドサービスである Amazon Cognito は AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon Cognito にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Cognito ユーザープールの設定と脆弱性分析

AWS は、ゲストオペレーティングシステム (OS) やデータベースのパッチ適用、ファイアウォール設定、ディザスタリカバリなどの基本的なセキュリティタスクを処理します。これらの手順は適切な第三者によって確認され、証明されています。詳細については、以下のリソース を参照してください。

- Amazon Cognito のコンプライアンス検証
- 責任共有モデル

AWS Amazon Cognito の マネージドポリシー

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを記述するよりも、AWS 管理ポリシーを使用する方が簡単です。チームに必要な許可のみを提供する IAM カスタマーマネージドポリシーを作成するには、時間と専門知識が必要です。すぐに開始するには、 AWS マネージドポリシーを使用できます。これらのポリシーは一般的なユースケースを対象としており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、「IAM ユーザーガイド」の「AWS 管理ポリシー」を参照してください。

AWS サービスは、 AWS 管理ポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスでは、新しい機能を利用できるようにするために、 AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、ポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS マネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が破損することはありません。

さらに、 は、複数の サービスにまたがる職務機能の マネージドポリシー AWS をサポートします。例えば、 ReadOnlyAccess AWS 管理ポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能のポリシーの一覧および詳細については、「IAM ユーザーガイド」の「AWS のジョブ機能のマネージドポリシー」を参照してください。

IAM コンソールからは、Amazon Cognito への許可を付与するために使用できる多数のポリシーを利用できます。

- `AmazonCognitoPowerUser` – ID プールとユーザープールのあらゆる側面へのアクセスと管理のための許可。このポリシーのアクセス許可を表示するには、「」を参照してくださいAmazonCognitoPowerUser。
- `AmazonCognitoReadOnly` - ID プールとユーザープールへの読み取り専用アクセスのための許可。このポリシーのアクセス許可を表示するには、「」を参照してくださいAmazonCognitoReadOnly。
- `AmazonCognitoDeveloperAuthenticatedIdentities` – 認証システムが Amazon Cognito と統合するのための許可。このポリシーのアクセス許可を表示するには、「」を参照してくださいAmazonCognitoDeveloperAuthenticatedIdentities。

これらのポリシーは Amazon Cognito チームによって維持されるため、新しい API が追加されても、ユーザーは引き続き同じアクセスレベルを維持することができます。

> **Note**
>
> 新しいアイデンティティプールを作成すると、認証済みユーザーアクセスとゲストユーザーアクセス用の新しいロールを自動的に作成できます。新しい IAM ロールでアイデンティティプールを作成する管理者には、ロールを作成するための IAM アクセス許可も必要です。

認証されていないゲストアクセスを持つアイデンティティプールは、認証されていないユーザーにセッションポリシー`AmazonCognitoUnAuthedIdentitiesSessionPolicy`として追加の AWS 管理ポリシー を適用します。この AWS 管理ポリシーには、管理目的での使用はありません。代わりに、アイデンティティプールの拡張認証フローでゲストユーザーに適用できるアクセス許可の範囲が制限されます。詳細については、「IAM ロール」を参照してください。

Amazon Cognito による AWS マネージドポリシーの更新

Amazon Cognito の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページへの変更に関する自動アラートについては、Amazon Cognito の「ドキュメント履歴」ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` – 新しいポリシー	ID プール内のゲストユーザーの特権スコープダウンのための AWS マネージドポリシーを追加しました。	2023 年 7 月 14 日
`AmazonCognitoPowerUser` と `AmazonCognitoReadOnly` — 既存のポリシーに対する更新	パワーユーザーが Amazon Cognito ユーザープールへの AWS WAF ウェブ ACLs の関連付けを表示および管理できるようにする新しいアクセス許可を追加しました。 読み取り専用ユーザーが Amazon Cognito ユーザープールへの AWS WAF ウェブ ACLs の関連付けを表示できるようにする新しいアクセス許可を追加しました。	2022 年 7 月 19 日
`AmazonCognitoPowerUser` – 既存ポリシーへの更新。	Amazon Cognito による Amazon Simple Email Service `PutIdentityPolicy` および `ListConfigurationSets` のオペレーションの呼び出しを可能にする新しい許可が追加されました。 この変更により、Amazon Cognito ユーザープールは、Amazon SES 送信認可ポリシーを更新し、ユーザープールの E メール送信設定で Amazon SES の設定を適用できるようになります。	2021 年 11 月 17 日

変更	説明	日付
`AmazonCognitoPowerUser` – 既存ポリシーへの更新。	Amazon Cognito による Amazon Simple Notification Service の `GetSMSSandboxAccountStatus` オペレーションの呼び出しを可能にする新しい許可が追加されました。 この変更により、Amazon Cognito ユーザープールは、ユーザープールを通じてすべてのエンドユーザーにメッセージを送信するために、Amazon Simple Notification Service のサンドボックスから乗り換える必要があるかどうかを判断できるようになります。	2021 年 6 月 1 日
Amazon Cognito が変更の追跡を開始	Amazon Cognito が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 3 月 1 日

Amazon Cognito リソースのタグ付け

タグは、ユーザーまたはAWSがAWSリソースに割り当てるメタデータラベルです。各タグは、キーと値から構成されます。ユーザーが割り当てるタグでは、ユーザーがキーと値を定義します。たとえば、1 つのリソースのキーを `stage` と定義し、値を `test` と定義します。

タグは、以下のことに役立ちます。

- AWS リソースを識別および整理します。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てることができます。タグは、どのリソースが関連しているかを示すのに役立ちます。例えば、Amazon DynamoDB テーブルに割り当てるタグと同じものを Amazon Cognito ユーザープールに割り当てることができます。
- AWS のコストを追跡します。これらのタグは、AWS Billing and Cost Management ダッシュボードで有効にできます。AWS では、コスト配分タグを使用してコストを分類し、毎月のコスト配分レポートを提供します。詳細については、AWS Billing ユーザーガイドの「コスト配分タグの使用」を参照してください。
- リソースに割り当てられたタグに基づいて、リソースへのアクセスをコントロールします。AWS Identity and Access Management (IAM) ポリシーの条件でキーと値を指定することによってアクセスをコントロールできます。例えば、ユーザープールの更新をユーザーに許可しても、そのユーザープールに `owner` タグと、そのユーザーの名前の値がある場合のみに限定することができます。詳細については、IAM ユーザーガイドの「タグを使用したアクセス制御」を参照してください。

AWS Command Line Interface または Amazon Cognito API を使用して、ユーザーと ID プールの両方のタグの追加、編集、または削除を行うことができます。Amazon Cognito コンソールを使用して、ユーザープールのタグを管理することもできます。

タグの使用方法のヒントについては、AWS の回答ブログのAWS タグ付け戦略記事を参照してください。

以下のセクションでは、Amazon Cognito のタグに関する詳細が説明されています。

Amazon Cognito でサポートされるリソース

Amazon Cognito の以下のリソースがタグ付けをサポートしています。

- ユーザープール

- ID プール

タグの制限

Amazon Cognito リソースのタグには、以下の制限が適用されます。

- リソースに割り当てることができるタグの最大数: 50
- キーの最大長 – 128 文字 (Unicode)
- 値の最大長 – 256 文字 (Unicode)
- キーと値の有効な文字 – a ~ z、A ~ Z、0 ~ 9、スペース、および特殊文字 (_ . : / = + - @)
- キーと値は大文字と小文字が区別されます
- `aws:` をキーのプレフィックスとして使用しないでください。AWS 用に予約済みです。

Amazon Cognito コンソールを使用したタグの管理

Amazon Cognito コンソールを使用して、ユーザープールに割り当てられたタグを管理することができます。

ユーザープールにタグを追加するには

1. [Amazon Cognito console] (Amazon Connect コンソール) に移動します。プロンプトが表示されたら、AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。
4. [User pool properties] (ユーザープールのプロパティ) タブを選択し、Tags (タグ) を探します。
5. [Add tags] (タグを追加する) を選択し、最初のタグを追加します。以前このユーザープールにタグを割り当てたことがある場合は、[Manage tags] (タグの管理) を選び、[Add another] (別のタグを追加) を選択します。
6. [Tag Key] (タグキー) と [Tag Value]] (タグ値) に値を入力します。
7. 追加するタグごとに、[Add another tag] (別のタグを追加) を選択します。
8. タグの追加を完了したら、[Save changes] (変更の保存) を選択します。

[Manage tags] (タグの管理) ページで、既存のタグのキーと値を編集することもできます。タグを削除するには、[Remove] (削除) を選択します。

AWS CLI の例

AWS CLI は、Amazon Cognito ユーザープールと ID プールに割り当てるタグの管理に役立つコマンドを提供します。

タグの割り当て

次のコマンドを使用して、既存のユーザープールと ID プールにタグを割り当てます。

Example **`tag-resource`** ユーザープール向けのコマンド

コマンドの `cognito-idp` セット内の `tag-resource` 使用して、ユーザープールにタグを割り当てます。

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

このコマンドには次のパラメータが含まれます。

- `resource-arn` – タグを割り当てるユーザープールの Amazon リソースネーム (ARN)。ARN を調べるには、Amazon Cognito コンソールでユーザープールを選択し、[General settings] (一般設定) タブの [Pool ARN] (プール ARN) 値を表示します。
- `tags` — 形式*key*=*value*のタグのキーバリューペアです。

一度に複数のタグを割り当てるには、カンマ区切りリストで指定します。

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example **`tag-resource`** ID プール向けのコマンド

コマンドの `cognito-identity` セット内の `tag-resource` 使用して、ID プールにタグを割り当てます。

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

このコマンドには次のパラメータが含まれます。

- `resource-arn` – タグを割り当てる ID プールの Amazon リソースネーム (ARN)。ARN を調べるには、Amazon Cognito コンソールで ID プールを選択し、[Edit identity pool] (ID プールの編集) をクリックします。その後、[Identity pool ID] (ID プールの ID) で [Show ARN] (ARN の表示) をクリックします。
- `tags` — 形式*key*=*value*のタグのキーバリューペアです。

一度に複数のタグを割り当てるには、カンマ区切りリストで指定します。

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

タグの表示

次のコマンドを使用して、ユーザープールと ID プールに割り当てたタグを表示します。

Example **`list-tags-for-resource`** ユーザープール向けのコマンド

コマンドの `cognito-idp` セット内の `list-tags-for-resource` 使用して、ユーザープールにタグを割り当てます。

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example **`list-tags-for-resource`** ID プール向けのコマンド

コマンドの `cognito-identity` セット内の `list-tags-for-resource` 使用して、ID プールにタグを割り当てます。

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

タグの削除

以下のコマンドを使用して、ユーザープールと ID プールからタグを削除します。

Example **`untag-resource`** ユーザープール向けのコマンド

コマンドの `cognito-idp` セット内の `untag-resource` 使用して、ユーザープールにタグを割り当てます。

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

--tag-keysパラメータには、1 つ以上のタグキーを指定します。タグ値は含みません。キーをスペースで区切ります。

Example **untag-resource** ID プール向けのコマンド

コマンドの cognito-identity セット内の untag-resource 使用して、ID プールにタグを割り当てます。

```
$ aws cognito-identity untag-resource \
> --resource-arn identity-pool-arn \
> --tag-keys Stage CostCenter Owner
```

--tag-keysパラメータには、1 つ以上のタグキーを指定します。タグ値は含みません。

> **Important**
>
> ユーザーまたは ID プールの削除後、削除されたプールに関連するタグは、削除から最大 30 日までコンソールまたは API コールに表示できます。

リソース作成時のタグの適用

次のコマンドを使用して、ユーザープールまたは ID プールを作成するときにタグを割り当てます。

Example **create-user-pool**コマンドとタグ

create-user-pool コマンドを使用してユーザープールを作成するときは、--user-pool-tags パラメータと共にタグを指定できます。

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

タグのキーバリューペアは、*key*=*value*形式が必須です。複数のタグを追加する場合は、カンマで区切りを付けたリストで指定します。

Example **create-identity-pool** コマンドとタグ

create-identity-pool コマンドを使用して ID プールを作成するときは、--identity-pool-tags パラメータと共にタグを指定できます。

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

タグのキーバリューペアは、*key*=*value*形式が必須です。複数のタグを追加する場合は、カンマで区切りを付けたリストで指定します。

Amazon Cognito API を使用したタグの管理

以下のアクションを Amazon Cognito API で使用することで、ユーザープールと ID プールのタグを管理できます。

ユーザープールタグの API アクション

次の API アクションを使用して、ユーザープールのタグを割り当て、表示、削除します。

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

ID プールタグの API アクション

次の API アクションを使用して、ID プールのタグを割り当て、表示、削除します。

- TagResource
- ListTagsForResource
- UntagResource
- CreateIdentityPool

Amazon Cognito のクォータ

Amazon Cognito には、アカウントで実行できる操作の最大数のデフォルトクォータ (以前は制限と呼ばれていました) があります。Amazon Cognito では、Amazon Cognito リソースの最大数と最大サイズのクォータがあります。

各 Amazon Cognito クォータは、1 つの 内の 1 つの AWS リージョン でリクエストの最大量を表します AWS アカウント。例えば、アプリは、米国東部 (バージニア北部) のすべてのユーザープールに対して、`UserAuthentication` オペレーションのデフォルトクォータ (RPS) レートまでの API リクエストを行うことができます。アジアパシフィック (東京) のアプリは、自分のリージョンのすべてのユーザープールに対して同じ量のリクエストを生成できます。 AWS は、一度に 1 つのリージョンでクォータ引き上げリクエストのみを付与できます。米国東部 (バージニア北部) でクォータを引き上げても、アジアパシフィック (東京) の最大リクエストレートには影響しません。

トピック

API リクエストレートクォータについて

クォータのカテゴリ

Amazon Cognito は API オペレーションの最大リクエストレートを適用します。Amazon Cognito が提供する API オペレーションの詳細については、「Amazon Cognito API およびエンドポイントリファレンス」を参照してください。ユーザープールの場合、これらのオペレーションは、`UserAuthentication` や `UserCreation` などの一般的なユースケースのカテゴリに分類されます。カテゴリ別のユーザープール API オペレーションのリストについては、「」を参照してくださいAmazon Cognito ユーザープール API オペレーションカテゴリとリクエストレートクォータ。

Service Quotas コンソール では、カテゴリユーザープールと ID プールごとにクォータの使用状況を追跡できます。Amazon Cognito ユーザープールのリクエストレートがクォータを超える場合は、追

加の容量を購入できます。Service Service Quotas コンソール で、ユーザープールのクォータの使用状況をカテゴリ別に追跡し、クォータの引き上げを購入できます。

操作のクォータは、カテゴリ内のすべての操作について、1 秒あたりのリクエスト (RPS) の最大数として定義されます。Amazon Cognito ユーザープールサービスは、各カテゴリのすべての操作にクォータを適用します。例えば、カテゴリ `UserCreation` には `SignUp`、`ConfirmSignUp`、`AdminCreateUser`、および `AdminConfirmSignUp` の 4 つのオペレーションが含まれます。これには合計で 50 RPS のクォータが割り当てられています。同時に複数のオペレーションが実行される場合は、オペレーションごとに最大 50 の RPS を別々に、または合わせて呼び出すことができます。

> **Note**
>
> カテゴリクォータはユーザープールにのみ適用されます。Amazon Cognito は、各 ID プールクォータを 1 つのオペレーションに適用します。カテゴリごとおよびオペレーションごとのリクエストレートクォータの両方で、 は、1 AWS アカウント つのリージョン内の のすべてのユーザープールまたはアイデンティティプールからのすべてのリクエストの集計レート AWS を測定します。

リクエストレートの特別な取り扱いを使用した Amazon Cognito ユーザープール API オペレーション

オペレーションクォータは、カテゴリレベルでのリクエストの合計数で測定され、適用されますが、特別な取扱いルールが適用される `AdminRespondToAuthChallenge` および `RespondToAuthChallenge` オペレーションに対しては例外になります。

`UserAuthentication` カテゴリには、Amazon Cognito ユーザープール API の `AdminInitiateAuth`、、`InitiateAuth`、 の 4 つのオペレーションが含まれますAdminRespondToAuthChallengeRespondToAuthChallenge。さらに、ホストされた UI のユーザー認証がこのクォータに影響します。`InitiateAuth` および `AdminInitiateAuth` オペレーションは、カテゴリクォータごとに測定され、施行されます。マッチングオペレーション `RespondToAuthChallenge` および `AdminRespondToAuthChallenge` は、別のクォータの対象であり、`UserAuthentication` カテゴリの制限の 3 倍です。この昇格されたクォータは、アプリケーションで設定された複数の認証チャレンジに対応します。このクォータは、ほとんどのユースケースに十分対応できます。アプリが認証チャレンジに最大 3 つの応答を行うと、追加のリクエストはUserAuthenticationカテゴリクォータにカウントされます。多要素認証 (MFA)、デバイス認

証、カスタム認証はすべて、ユーザープールに組み込むことができるチャレンジプロンプトの例です。

例えば、UserAuthenticationカテゴリのクォータが 80 RPS の場合、RespondToAuthChallenge または を最大 240 RPS (3 * 80 RPS) のレートAdminRespondToAuthChallengeで呼び出すことができます。ユーザープールが認証ごとに 4 ラウンドのチャレンジを要求し、1 秒あたり 70 人のユーザーがサインインする場合、合計RespondToAuthChallengeは 280 RPS (70 x 4) で、クォータを 40 RPS 上回ります。40 RPS が 70 回 の InitiateAuth コールに追加されたことで、UserAuthentication カテゴリの総使用量 は 110 RPS (40 + 70) になります。この値は 80 RPS に設定されたカテゴリクォータを 30 RPS 超えるため、Amazon Cognito はアプリケーションからのリクエストを調整します。

月次のアクティブユーザー

Amazon Cognito がユーザープールの請求を計算すると、毎月のアクティブユーザー (MAU) ごとに料金が請求されます。クォータ引き上げリクエストの計画では、現在の MAU 数と予測される MAU 数を考慮してください。暦月内に、ユーザーに関連した ID オペレーションがある場合、そのユーザーは MAU としてカウントされます。ユーザーをアクティブにするアクティビティには、次のものが含まれます。

- ユーザーのサインアップまたは管理者作成
- サインイン
- サインアウト
- ユーザーアカウントの確認または属性検証
- パスワードのリセット
- ユーザー属性、グループのメンバーシップ、または MFA 設定を変更する
- ユーザーの詳細属性をクエリする
- ユーザーのアクティブ化、非アクティブ化、または削除

> **Note**
>
> ユーザーのカテゴリクエリ詳細属性には API オペレーション が含まれますがAdminGetUser、 は含まれませんListUsers。大規模なユーザープールの詳細な user-by-user クエリは、 AWS 請求に大きな影響を与える可能性があります。超過料金を回避するに

は、 を使用してユーザーデータを収集ListUsersするか、外部データベースにユーザー情報を保存します。

API リクエストレートクォータ

クォータ要件を特定する

> **Important**
>
> UserAuthentication、、 などのカテゴリの Amazon Cognito クォータを増やす場合はUserCreationAccountRecovery、他の のクォータを増やす必要がある場合があります AWS のサービス。例えば、Amazon Cognito が Amazon Simple Notification Service (Amazon SNS) または Amazon Simple Email Service (Amazon SES) で送信するメッセージは、これらのサービスでリクエストレートのクォータが不十分な場合に失敗することがあります。

クォータ要件を計算するには、特定の期間内にアプリケーションとやり取りするアクティブなユーザーの数を判断します。例えば、アプリケーションに 8 時間の期間内に平均 100 万人のアクティブユーザーによるサインインがあることを想定する場合、1 秒あたり平均 35 人のユーザーを認証できる必要があります。

さらに、平均的なユーザーセッションが 2 時間であると仮定し、トークンが 1 時間後に期限切れになるように設定している場合、各ユーザーは、セッション中にトークンを 1 回更新する必要があります。この負荷をサポートするために UserAuthentication カテゴリに必要な平均クォータは 70 RPS になります。

8 時間のユーザーサインイン頻度の変動を考慮して 3:1 peak-to-average の比率を前提とする場合、必要なクォータは UserAuthentication 200 RPS です。

> **Note**
>
> ユーザーアクションごとに複数のオペレーションを呼び出す場合は、カテゴリレベルで個々のオペレーションのコールレートを合計する必要があります。

クォータ制限のリクエストレートを最適化する

API レート制限を増やすと AWS 請求コストが増加するため、クォータの引き上げをリクエストする前に、使用量モデルの調整を検討してください。以下は、リクエストレートを最適化するアプリケーションアーキテクチャの例です。

バックオフ待機期間後に再試行する

API コールごとにエラーをキャッチし、バックオフ期間後に再試行することができます。バックオフアルゴリズムは、ビジネスのニーズと負荷に応じて調整できます。Amazon SDK には、組み込みの再試行ロジックがあります。詳細については、「 で構築するツール」を参照してください AWS。

頻繁に更新される属性に外部データベースを使用する

アプリケーションが、カスタム属性の読み取りと書き込みにユーザープールへの複数のコールを必要とする場合は、外部ストレージを使用します。優先データベースを使用してカスタム属性を保存する、またはキャッシュレイヤーを使用してサインイン時にユーザープロファイルをロードすることができます。このプロファイルは、ユーザープールからユーザープロファイルを再ロードしないでも、必要に応じてキャッシュから参照することができます。

クライアント側で JSON ウェブトークン (JWTsを検証する

アプリケーションは、JWT トークンを信頼する前にそれらを検証する必要があります。API リクエストをユーザープールに送信しなくても、クライアント側でトークンの署名と有効性を検証できます。トークンが検証されたら、トークン内のクレームを信頼して、さらに多くの `getUser` API コールを実行せずにクレームを使用することができます。詳細については、「JSON Web トークンの検証」を参照してください。

待合室でウェブアプリケーションへのトラフィックを調整する

試験の受験やライブイベントへの参加など、時間が限られているイベント中に多数のユーザーがサインインするトラフィックが予想される場合は、セルフスロットリングメカニズムを使用してリクエストトラフィックを最適化できます。例えば、セッションが利用可能になるまでユーザーが待機できる待合室を設定して、使用可能な容量がある間にリクエストを処理できます。待合室のリファレンスアーキテクチャについては、「AWS Virtual Waiting Room」を参照してください。

JWTsキャッシュする

アクセストークンの有効期限が切れるまで再利用します。API Gateway でトークンキャッシュを使用するフレームワークの例については、「」を参照してくださいキャッシュトークン。ユー

ザー情報をクエリする API リクエストを生成する代わりに、有効期限が切れるまで ID トークンをキャッシュし、キャッシュからユーザー属性を読み取ります。

での API リクエストレートの操作の詳細については AWS、「ワークロードでの API スロットリングの管理とモニタリング」を参照してください。 AWS 請求書にコストを追加する Amazon Cognito オペレーションの最適化については、「」を参照してください のコスト管理。

クォータの使用状況を追跡する

Amazon Cognito は、アカウントレベルで各 API オペレーションカテゴリ CloudWatch の `CallCount`および `ThrottleCount`メトリクスを Amazon で生成します。お客様が実行するカテゴリ関連のコールの合計数を追跡するには、`CallCount` を使用できます。カテゴリ関連のスロットルされたコールの合計数は、`ThrottleCount` を使用して追跡できます。カテゴリ内のコールの合計数を計上するには、`Sum` 統計で `CallCount` および `ThrottleCount` メトリクスを使用できます。詳細については、「 CloudWatch 使用状況メトリクス」を参照してください。

サービスクォータをモニタリングする場合、使用率 は使用中のサービスクォータの割合です。例えば、クォータの値が 200 リソースで 150 リソースが使用中の場合、使用率は 75% です。使用状況は、サービスクォータに対する使用中のリソースまたはオペレーションの数です。

CloudWatch メトリクスによる使用状況の追跡

を使用して、Amazon Cognito ユーザープールの使用率メトリクスを追跡および収集できます CloudWatch。ダッシュボードには AWS のサービス 、使用するすべての CloudWatchに関するメトリクスが表示されます。では CloudWatch、メトリクスアラームを作成して通知したり、モニタリングする特定のリソースを変更したりできます。 CloudWatch メトリクスの詳細については、CloudWatch 「使用状況メトリクスの追跡」を参照してください。

Service Quotas メトリクスを使用した使用率の追跡

Amazon Cognito ユーザープールは、Service Quotas と統合されています。Service Quotas は、Service Quotas の使用状況を表示および管理するためのコンソールインターフェイスです。Service Quotas コンソールでは、特定のクォータの値の検索、モニタリング情報の表示、クォータの引き上げのリクエスト、 CloudWatch アラームの設定を行うことができます。アカウントがアクティブになってからしばらくすると、リソース使用率のグラフを表示できます。

Amazon Amazon Cognito Cognito ID プールの Service Quotas コンソールの Applied account-level quota value 列には、現在のクォータが表示されます。 Amazon Cognito 使用率 列には、現在のクォータ使用率が表示されます。調整可能な Amazon Cognito ユーザープール requests-per-second

(RPS) クォータには、現在の使用状況が表示されます。Service Quotas コンソールで CloudWatch メトリクスに移動して、選択したクォータメトリクスを詳しく確認することもできます。Service Quotas コンソールでのクォータの表示に関する詳細については、「Viewing service quotas」を参照してください。

毎月のアクティブユーザーを追跡MAUs)

ユーザープール内の月間アクティブユーザー数 (MAUsは、リクエストレートクォータの増加に関する計画に重要なデータを提供します。API リクエストレートを、特定の期間にアクティブだったユーザーの数と比較できます。その知識があれば、アプリケーションのアクティブユーザーの増加が使用量モデルのクォータにどのように影響するかを計算できます。例えば、米国西部 (オレゴン) のアプリケーションを組み合わせると、1 か月で 200 万人のアクティブユーザーがいて、`UserAuthentication`カテゴリがデフォルトの 120 リクエスト/秒 (RPS) のクォータで時折スロットリングエラーを受け取ったとします。前月、広告キャンペーンが成功する前に 100 万 MAUs があり、アプリケーションが 80 RPS を超えることはありませんでした。新しいテレビスポットの結果として同様のスパイクが予想される場合は、調整されたクォータが 160 RPS の次の 100 万人のユーザーに対応するために、追加の 40 RPS を購入できます。

MAUs を確認するには

AWS Billing コンソールにアクセスし、最近の請求書を確認します。サービス 別の料金では、Cognito でフィルタリングして、その請求期間の MAUsの内訳を表示できます。

クォータ引き上げのリクエスト

Amazon Cognito には、各 のユーザープールと ID プールの で実行できる 1 秒あたりの最大オペレーション数のクォータがあります AWS リージョン。調整可能な Amazon Cognito ユーザープール API リクエストレートクォータの引き上げを購入できます。現在のクォータを確認し、Service Quotas コンソールまたは Service Quotas API オペレーション `ListAWSDefaultServiceQuotas`および を使用して引き上げを購入します`RequestServiceQuotaIncrease`。

- Service Quotas コンソールを使用してクォータの引き上げを購入するには、「Service Quotas ユーザーガイド」の「API クォータの引き上げのリクエスト」を参照してください。 Service Quotas
- AWS は、クォータ引き上げリクエストを 10 日以内に完了することを目標としています。ただし、いくつかの考慮事項により、リクエストの処理時間が 10 日を超える場合があります。例えば、一部のリクエストでは Amazon Cognito で追加のハードウェア容量をプロビジョニングする必要があり、リクエストボリュームが季節的に増加すると遅延が発生する場合があります。

- Service Quotas でクォータが利用できない場合は、Service limit increase フォームを使用してください。

> **Important**
>
> クォータは、調整可能なもののみを引き上げることができます。クォータ容量を増やす必要があります。クォータ引き上げの料金については、Amazon Cognito の料金」を参照してください。

Amazon Cognito ユーザープール API オペレーションカテゴリとリクエストレートクォータ

Amazon Cognito には、認証モデルが異なる API オペレーションのクラスが重複しているため、各オペレーションはカテゴリに属します。各カテゴリには、アカウント内の 1 つの AWS リージョン 内のすべてのユーザープールで、すべてのメンバー API オペレーションに対して独自のプールされたクォータがあります。調整可能なカテゴリのクォータについてのみ、引き上げをリクエストできます。詳細については、「クォータ引き上げのリクエスト」を参照してください。クォータ調整は、単一リージョンのアカウント内のユーザープールに適用されます。Amazon Cognito では、一部のカテゴリ[3]でのオペレーションを制限し、ユーザープールごとのリクエスト数/秒 (RPS) を 5 にしています。デフォルトのクォータ (RPS) は、 内のすべてのユーザープールに追加に適用されます AWS アカウント。

> **Note**
>
> 各カテゴリのクォータは、月間アクティブユーザー数 (MAU) で測定されます。MAU が 200 万未満の AWS アカウント は、デフォルトのクォータ内で動作できます。MAUs が 100 万未満で、Amazon Cognito がリクエストをスロットリングしている場合は、アプリの最適化を検討してください。詳細については、「クォータ制限のリクエストレートを最適化する」を参照してください。

カテゴリオペレーションクォータは、1 つの AWS リージョン内のユーザープールのすべてのユーザーに適用されます。Amazon Cognito は、アプリが 1 人のユーザーに対して生成できるリクエストの数にも制限を設けています。ユーザーごとの API リクエストを以下のテーブルに示すとおり、制限する必要があります。

Amazon Cognito ユーザープールのユーザーあたりのリクエストレートクォータ

操作	ユーザーあたりのオペレーション数 (1 秒あたりのオペレーション数)
読み込みユーザープロファイル 例: `GetUser`、`GetDevice`	10
書き込みユーザープロファイル 例: `UpdateUserAttributes` 、`SetUserSettings`	10

次の表に示すとおり、カテゴリ別の API リクエストを制限する必要があります。

Amazon Cognito ユーザープールのカテゴリごとのリクエストレートクォータ

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserAuthentication` • InitiateAuth • `InitiateAuth` または トークンエンドポイント によるトークンの更新 • RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1]	ユーザーを認証 (サインイン) するオペレーション。 これらのオペレーションは リクエストレートの特別な取り扱いを使用した Amazon Cognito ユーザープール API オペレーション の対象となります。	120	はい

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
• ホストされた UI のサインインと MFA (認証コード付与または暗黙的な付与)[2]			
`UserCreation` • SignUp • ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	Amazon Cognito ローカルユーザーを作成または確認するオペレーション。これは、Amazon Cognito ユーザープールによって直接作成され、検証されるユーザーです。	50	はい
`UserFederation` サードパーティー ID プロバイダーを使用してユーザーを Amazon Cognito ユーザープールにフェデレート (認証) するオペレーション。	ユーザープールフェデレーションエンドポイントに IdP レスポンスを送信するオペレーション。IdP トークンを生成する OIDC またはソーシャルプロバイダーのオペレーション、およびすべての SAML リクエストが、このクォータに寄与します。	25	はい

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserAccountRecovery` • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • ホストされた UI のパスワードリセット	ユーザーのアカウントを回復する、またはユーザーのパスワードを変更もしくは更新するオペレーション。	30	いいえ
`UserRead` • AdminGetUser • GetUser	ユーザープールからユーザーを取得する操作。	120	はい

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserUpdate` • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	ユーザーとユーザー属性を管理するために使用するオペレーション。	25	いいえ
`UserToken` • RevokeToken	トークン管理のオペレーション	120	はい

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	記憶されているデバイスやグループメンバーシップなどのユーザーリソース情報を Amazon Cognito から取得するオペレーション。	50	はい

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUserMFAPreference • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAPreference • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	記憶されているデバイスやグループメンバーシップなど、ユーザーのリソース情報を更新するオペレーション。	25	いいえ
`UserList` • ListUsers • ListUsersInGroup	ユーザーのリストを返す操作。	30	いいえ

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserPoolRead` • DescribeUserPool • ListUserPools	ユーザープールを読み取る操作。	15	いいえ
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	ユーザープールを作成、更新、または削除するオペレーション。	15	いいえ

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserPoolResourceRead` • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization	ユーザープールからグループやリソースサーバーなどのリソースに関する情報を取得するオペレーション。[3]	20	いいえ

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer	ユーザープールのグループやリソースサーバーなどのリソースを変更するオペレーション。[3]	15	いいえ

カテゴリ	説明	デフォルトクォータ (RPS)	調整可能
• UpdateUserPoolDomain • SetRiskConfiguration • SetUICustomization • TagResource • UntagResource			
UserPoolClientRead • DescribeUserPoolClient • ListUserPoolClients	ユーザープールクライアントに関する情報を取得するオペレーション。[3]	15	いいえ
UserPoolClientUpdate • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	ユーザープールクライアントを作成、更新、削除するオペレーション。[3]	15	いいえ
ClientAuthentication トークンエンドポイントに対する client_credentials 付与タイプのリクエスト	machine-to-machine リクエストの承認に使用される認証情報を生成するオペレーション	150	いいえ

[1] が ChallengeNameの RespondToAuthChallengeまたは AdminRespondToAuthChallengeレスポンスは、UserAccountRecoveryカテゴリにNEW_PASSWORD_REQUIREDカウントされます。他のすべてのチャレンジレスポンスは、UserAuthenticationカテゴリにカウントされます。

[2] サインイン中のホストされた UI オペレーションごとに、クォータに 1 つのリクエストが付与されます。例えば、サインインして MFA コードを入力したユーザーは 2 つのリクエストとしてカウントされます。認証コード付与でのトークン利用には、UserAuthenticationカテゴリのクォータと同じレートで追加のクォータ割り当てが適用されます。

[3] このカテゴリの個々のオペレーションには、1 つのユーザープールに対して 5 RPS を超えるレートでオペレーションが呼び出されないようにする制約があります。

Amazon Cognito ID プール (フェデレーティッドアイデンティティ) API オペレーションリクエストレートのクォータ

操作	説明	デフォルトクォータ (RPS)[1]	調整可能	クォータ引き上げの対象
GetId	ID プールからアイデンティティ ID を取得します。	25	はい	アカウントチームにお問い合わせください。
GetOpenIdToken	クラシックワークフローの ID プールから OpenID トークンを取得します。	200	はい	アカウントチームにお問い合わせください。
GetCredentialsForIdentity	拡張ワークフローの ID プールから AWS 認証情報を取得します。	200	はい	アカウントチームにお問い合わせください。

操作	説明	デフォルトクォータ (RPS)[1]	調整可能	クォータ引き上げの対象
GetOpenIdTokenForDeveloperIdentity	デベロッパーワークフローの ID プールから OpenID トークンを取得します。	50	はい	アカウントチームにお問い合わせください。
ListIdentities	アイデンティティプール内の ID のリストを取得します。	5	はい	アカウントチームにお問い合わせください。
DeleteIdentities	アイデンティティプールから登録した ID を 1 つ以上削除します。	10	[Yes (はい)]	アカウントチームにお問い合わせください。
TagResource	アイデンティティプールにタグを適用します。	5	はい	アカウントチームにお問い合わせください。
UntagResource	アイデンティティプールからタグを削除します。	5	はい	アカウントチームにお問い合わせください。
ListTagsForResource	アイデンティティプールに適用されたタグのリストを表示します。	10	[Yes (はい)]	アカウントチームにお問い合わせください。

[1] デフォルトのクォータは、 の にある ID プールの最小リクエストレートクォータ AWS リージョンです AWS アカウント。リージョンによっては RPS クォータが高くなる場合があります。

リソースの番号とサイズのクォータ

リソースクォータは、Amazon Cognito のリソース、入力フィールド、期間、およびその他の機能の最大数またはサイズです。

一部のリソースクォータの調整は、サービスクォータコンソールまたはサービス制限の引き上げフォームからリクエストできます。Service Quotas コンソールからクォータの引き上げをリクエストするには、Service Quotas ユーザーガイドの「Requesting a quota increase」を参照してください。Service Quotas でクォータが利用できない場合は、Service limit increase フォームを使用してください。

> **Note**
>
> リージョン あたりのユーザープールなどの AWS アカウント レベルのリソースクォータは、各 の Amazon Cognito リソースに適用されます AWS リージョン。 例えば、米国東部 (バージニア北部) に 1,000 個のユーザープールを持ち、欧州 (ストックホルム) に別の 1,000 個のユーザープールを持つことができます。

次の表は、デフォルトのリソースクォータと、それらが調整可能かどうかを示しています。

Amazon Cognito ユーザープールのリソースクォータ

リソース	クォータ	調整可能	最大クォータ
ユーザープールあたりのアプリケーションクライアント	1,000	はい	10,000
リージョンごとのユーザープール	1,000	はい	10,000
ユーザープールあたりのアイデンティティプロバイダー	300	はい	1,000

リソース	クォータ	調整可能	最大クォータ
ユーザープールあたりのリソースサーバー	25	はい	300
ユーザープールあたりのユーザー数	40,000,000	はい	アカウントチームにお問い合わせください。
トークン生成前の Lambda トリガーの合計変更数[1]	5,000	はい	アカウントチームにお問い合わせください。
ユーザープールあたりのカスタム属性数	50	いいえ	該当なし
属性あたりの文字数	2,048 バイト	いいえ	該当なし
カスタム属性名の文字数	20	いいえ	該当なし
パスワードポリシーに最低限必要なパスワード文字数	6～99	いいえ	該当なし
ごとに毎日送信される E メールメッセージ AWS アカウント[2]	50	いいえ	該当なし
E メール件名の文字数	140	いいえ	該当なし
E メールメッセージの文字数	20,000	いいえ	該当なし
SMS 検証メッセージの文字数	140	いいえ	該当なし
パスワードの文字数	256	いいえ	該当なし

リソース	クォータ	調整可能	最大クォータ
ID プロバイダー名の文字数	32	いいえ	該当なし
ID プロバイダーあたり識別子数	50	いいえ	該当なし
ユーザーにリンクされた ID 数	5	いいえ	該当なし
アプリケーションクライアントあたりのコールバック URL 数	100	いいえ	該当なし
アプリケーションクライアントあたりのログアウト URL 数	100	いいえ	該当なし
リソースサーバーあたりのスコープ	100	いいえ	該当なし
アプリケーションクライアントあたりのスコープ数	50	いいえ	該当なし
アカウントあたりのカスタムドメイン数	4	いいえ	該当なし
各ユーザーが所属できるグループ数	100	いいえ	該当なし
ユーザープールあたりのグループ	10,000	いいえ	該当なし

[1] このクォータは、トークン生成前の Lambda トリガー からのトークンで発生する場合があります。アクセストークンおよび ID トークンの既存および追加のクレームとスコープの数の合計は、こ

のクォータ以下になる必要があります。抑制されたクレームとスコープは、このクォータには加算されません。

[2] このクォータは、Amazon Cognito ユーザープールでデフォルトの E メール機能を使用している場合にのみ適用されます。E メールの配信ボリュームがより高い場合は、Amazon SES の E メール設定を使用するようにユーザープールを設定します。詳細については、「Amazon Cognito ユーザープールの E メール設定」を参照してください。

Amazon Cognito ユーザープールセッションの有効性パラメータ

トークン	クォータ
ID トークン	5 分 ~ 1 日
更新トークン	1 時間 ~ 3,650 日
アクセストークン	5 分 ~ 1 日
ホストされた UI セッション cookie	1 時間
認証セッショントークン	3 分 ~ 15 分

Amazon Cognito ユーザープールコードセキュリティリソースクォータ (調整不可)

リソース	クォータ
サインアップ確認コードの有効期間	24 時間
ユーザー属性検証コードの有効期間	24 時間
多要素認証 (MFA) コードの有効期間	3 ~ 15 分
パスワードを忘れた場合のコードの有効期間	1 時間
ユーザーあたりの `ConfirmForgotPassword` および `ForgotPassword` リクエストの最大数/時間[1]	5 ~ 20

リソース	クォータ
ユーザーあたりの ResendConfirmationCode リクエストの最大数/時間	5
ユーザーあたりの ConfirmSignUp リクエストの最大数/時間	15
ユーザーあたりの ChangePassword リクエストの最大数/時間	5
ユーザーあたりの GetUserAttributeVerificationCode リクエストの最大数/時間	5
ユーザーあたりの VerifyUserAttribute リクエストの最大数/時間	15

[1] Amazon Cognito は、パスワード更新リクエストのリスク要因を評価し、評価したリスクレベルに応じたクォータを割り当てます。詳細については、「パスワードを忘れた場合の対応」を参照してください。

Amazon Cognito ユーザープールへのユーザーインポートジョブのリソースクォータ

リソース	クォータ	調整可能	最大クォータ
ユーザープールあたりのユーザーインポートジョブ	1,000	はい	アカウントチームにお問い合わせください。
ユーザーインポート CSV 行あたりの最大文字数	16,000	いいえ	該当なし
最大 CSV ファイルサイズ	100 MB	いいえ	該当なし
CSV ファイルあたりのユーザーの最大数	500,000	いいえ	該当なし

Amazon Cognito ID プール (フェデレーティッドアイデンティティ) リソースのクォータ

リソース	クォータ	調整可能	最大クォータ
アカウントあたりのアイデンティティプール	1,000	はい	該当なし
ID プールあたりの Amazon Cognito ユーザープールプロバイダー数	50	はい	1,000
ID プール名の文字長	128 バイト	いいえ	該当なし
ログインプロバイダー名の文字長	2,048 バイト	いいえ	該当なし
ID プールあたりのアイデンティティ数	無制限	いいえ	該当なし
ロールマッピングを指定できる ID プロバイダー数	10	いいえ	該当なし
単一のリストまたは lookup コールからの結果の数	60	いいえ	該当なし
ロールベースのアクセス制御 (RBAC) ルール	25	いいえ	該当なし

Amazon Cognito Sync リソースクォータ

リソース	クォータ	調整可能	最大クォータ
アイデンティティあたりのデータセット	20	はい	アカウントチームにお問い合わせください。
データセットあたりのレコード	1,024	はい	アカウントチームにお問い合わせください。
1 つのデータセットのサイズ	1 MB	はい	アカウントチームにお問い合わせください。
データセット名の文字数	128 バイト	いいえ	該当なし
正常なリクエスト後の一括発行の待機時間	24 時間	いいえ	該当なし

Amazon Cognito API およびエンドポイントリファレンス

以下のリファレンスでは、Amazon Cognito の各機能のサービスエンドポイントについて説明しています。Amazon Cognito ユーザープールには、ユーザープールドメインを持つユーザープールエンドポイントとユーザープール API のオプションがあります。Amazon Cognito ユーザープールのユーザープール API での API オペレーションクラスの内訳については、「Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用」を参照してください。

AWS リージョン によるユーザープール API のサービスエンドポイントの完全なリストについては、AWS 全般リファレンスの「サービスエンドポイント」を参照してください。

トピック

- ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス
- Amazon Cognito ユーザープール API リファレンス
- Amazon Cognito ID プール (フェデレーティッド ID) API リファレンス
- Amazon Cognito Sync API リファレンス

ユーザープールフェデレーションエンドポイントとホストされた UI リファレンス

ユーザープールにドメインを割り当てると、Amazon Cognito はここにリストされているパブリックウェブページをアクティブにします。ドメインは、すべてのアプリクライアントの中央アクセスポイントとして機能します。これらには、ユーザーがサインアップしてサインイン (ログインエンドポイント) およびサインアウト (ログアウトエンドポイント) できるホストされた UI が含まれます。これらのリソースの詳細については、「Amazon Cognito でホストされた UI およびフェデレーションエンドポイントの設定と使用」を参照してください。

これらのページには、ユーザープールがサードパーティーの SAML、OpenID Connect (OIDC)、および OAuth 2.0 ID プロバイダー () と通信できるようにするパブリックウェブリソースも含まれていますIdPs。フェデレーション ID プロバイダーを使用してユーザーをサインインするには、ユーザーは、インタラクティブなホストされた UI ログインエンドポイント または OIDC 認可エンドポイント に対してリクエストを開始する必要があります。Authorize エンドポイントは、ホストされている UI または IdP サインインページにユーザーをリダイレクトします。

アプリは、Amazon Cognito ユーザープール API を使用してローカルユーザーをサインインすることもできます。ローカルユーザーは、外部 IdP を介したフェデレーションなしに、ユーザープールディレクトリにのみ存在します。

Amazon Cognito は、ホストされた UI とフェデレーションエンドポイントに加えて JavaScript、Android、iOS などの SDKs と統合します。SDK は、Amazon Cognito API サービスエンドポイントにユーザープール API オペレーションを実行するためのツールを提供します。サービスエンドポイントの詳細については、「Amazon Cognito Identity エンドポイントとクォータ」を参照してください。

> **Warning**
>
> Amazon Cognito domain のエンドエンティティまたは中間 Transport Layer Security (TLS) 証明書を固定しないでください。 AWS は、すべてのユーザープールエンドポイントとプレフィックスドメインのすべての証明書を管理します。Amazon Cognito 証明書をサポートする信頼チェーン内の認証局 (CA) は、動的にローテーションし、更新されます。アプリを中間証明書またはリーフ証明書に固定すると、 が証明書を AWS ローテーションすると、アプリは予告なく失敗する可能性があります。
> 代わりに、アプリケーションを利用できるすべての Amazon ルート証明書に固定します。詳細については、AWS Certificate Manager ユーザーガイドの「証明書のピンニング」にあるベストプラクティスと推奨事項を参照してください。

トピック

- ホストされた UI エンドポイントのリファレンス
- OAuth 2.0、OpenID Connect、および SAML 2.0 フェデレーションエンドポイントリファレンス
- OAuth 2.0 許可
- Amazon Cognito ユーザープールでの認証コード付与での PKCE の使用
- ホストされた UI とフェデレーションエラーレスポンス

ホストされた UI エンドポイントのリファレンス

Amazon Cognito はユーザープールにドメインを追加すると、このセクションのホストされた UI エンドポイントをアクティブにします。これらは、ユーザーがユーザープールのコア認証オペレーションを完了できるウェブページです。これらには、パスワード管理、多要素認証 (MFA)、および属性

検証用のページが含まれます。ホストされた UI サインアップのエクスペリエンスについて詳しくは、「ホストされた UI でのサインアップとサインイン」を参照してください。

ホストされた UI を構成するウェブページは、顧客とのインタラクティブなユーザーセッションのためのフロントエンドウェブアプリケーションです。アプリはユーザーのブラウザでホストされた UI を呼び出す必要があります。Amazon Cognito は、この章のウェブページに対するプログラムによるアクセスをサポートしていません。JSON レスポンスを返す OAuth 2.0、OpenID Connect、および SAML 2.0 フェデレーションエンドポイントリファレンス のフェデレーションエンドポイントは、アプリコードで直接クエリできます。認可エンドポイント は、ホストされた UI または IdP サインインページのどちらかにリダイレクトされます。また、ユーザーのブラウザで開く必要があります。

このガイドのトピックでは、頻繁に使用されるホスト UI エンドポイントについて詳しく説明します。Amazon Cognito は、ユーザープールにドメインを割り当てる際に以下のウェブページを利用できるようにします。

ホストされた UI エンドポイント

エンドポイント URL	説明	アクセス方法
https://############ /login	ユーザープールのローカルユーザーとフェデレーションユーザーにサインインします。	認可エンドポイント、/logout、と /confirmforgotPassword などのエンドポイントからリダイレクトします。ログインエンドポイント を参照してください。
https://############ /logout	ユーザープールユーザーをサインアウトします。	直接リンク。ログアウトエンドポイント を参照してください。
https://############ /ConfirmUser	E メールリンクを選択してユーザーアカウントを検証したユーザーを確認します。	ユーザーが E メールメッセージ内のリンクを選択しました。
https://############ /signup	新規ユーザーをサインアップします。/login ページは、ユーザーが [Sign up] (サインアップ) を選択すると、/signup に移動します。	/oauth2/authorize と同じパラメータで直接リンクします。

エンドポイント URL	説明	アクセス方法
https://############ /confirm	ユーザープールがサインアップしたユーザーに確認コードを送信すると、ユーザーにコードの入力を求めます。	/signup からのリダイレクトのみ。
https://*Your user pool domain*/forgotPassword	ユーザーにユーザー名の入力を求め、パスワードリセットコードを送信します。/login ページは、ユーザーが [Forgot your password?] (パスワードをお忘れですか?) を選択するとユーザーを /forgotPassword に誘導します。	1. /login のパスワードを忘れた場合のリンクから。 2. /oauth2/authorize と同じパラメータで直接リンクします。
https://*Your user pool domain*/confirmforgotPassword	ユーザーにパスワードリセットコードと新しいパスワードの入力を求めます。/forgotPassword ページは、ユーザーが [Reset your password] (パスワードのリセット) を選択するとユーザーを /confirmforgotPassword ページに誘導します。	/forgotPassword からのリダイレクトのみ。
https://############ /resendcode	ユーザープールにサインアップしたユーザーに、新しい確認コードを送信します。	/confirm の [新しいコードを送信] リンクからのリダイレクトのみ。

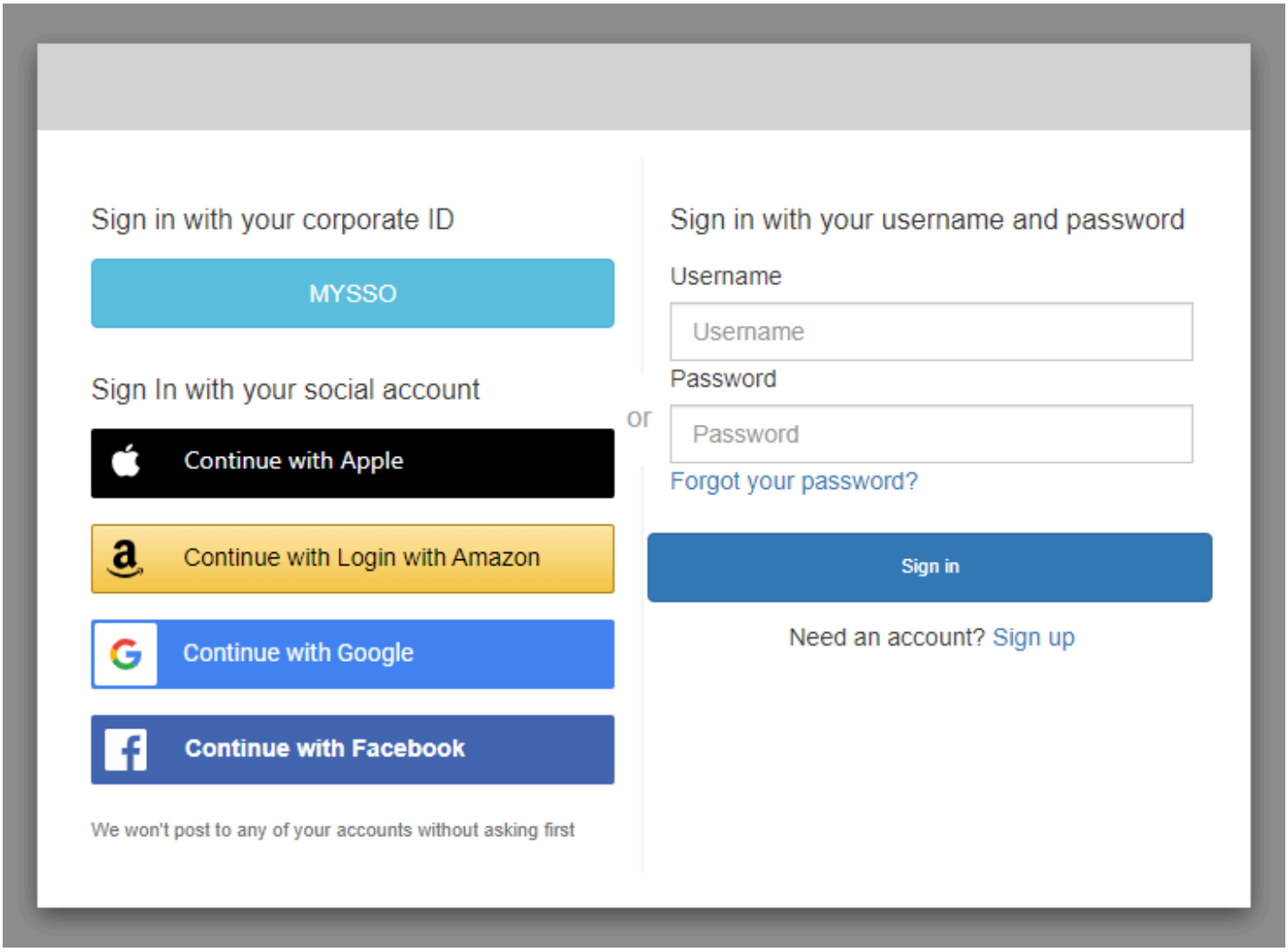
トピック

- ログインエンドポイント
- ログアウトエンドポイント

ログインエンドポイント

ログインエンドポイントは認証サーバーであり、認可エンドポイント からのリダイレクト先です。これは、ID プロバイダーを指定しない場合のホスト UI へのエントリポイントです。ログインエンドポイントへのリダイレクトを生成すると、ログインページにロードされ、ユーザーにクライアントに設定されている認証オプションが提示されます。

> **Note**
>
> ログインエンドポイントはホスト UI のコンポーネントです。アプリで、ログインエンドポイントにリダイレクトするフェデレーションページとホストされた UI ページを呼び出します。ユーザーがログインエンドポイントに直接アクセスすることはベストプラクティスではありません。



GET/ログイン

/login エンドポイントは、ユーザーの最初のリクエストの HTTPS GET のみをサポートします。アプリは Chrome や Firefox などのブラウザでページを呼び出します。/login から にリダイレクトすると認可エンドポイント、最初のリクエストで指定したすべてのパラメータが渡されます。ログインエンドポイントは authorize エンドポイントのすべてのリクエストパラメータをサポートします。ログインエンドポイントに直接アクセスすることもできます。ベストプラクティスとしては、すべてのユーザーのセッションを /oauth2/authorize から開始してください。

例 - ユーザーにサインインを促す

この例では、ログイン画面が表示されます。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=ad398u21ijw3s9w3939&
                redirect_uri=https://YOUR_APP/redirect_uri&
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

例 - レスポンス

認証サーバーは認証コードとステートを伴ってアプリにリダイレクトします。サーバーは、コードとステートをフラグメントではなく、クエリ文字列のパラメータで返す必要があります。

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

ユーザーが開始したサインインリクエスト

ユーザーは、/login エンドポイントをロードした後で、ユーザー名とパスワードを入力して [サインイン] を選択できます。これを行うと、GET リクエストと同じヘッダーリクエストパラメータと、ユーザー名、パスワード、およびデバイスのフィンガープリントを含むリクエスト本文で、HTTPS POST リクエストが生成されます。

ログアウトエンドポイント

/logout エンドポイントは、リダイレクションエンドポイントです。ユーザーをサインアウトし、アプリクライアントの認可されたサインアウト URL または/loginエンドポイントにリダイレク

トします。/logout エンドポイントへの GET リクエストで使用可能なパラメータは、Amazon Cognito がホストする UI のユースケースに合わせて調整されます。

ユーザーをホストされた UI にリダイレクトして再度ログインするには、リクエストに redirect_uri パラメータを追加します。redirect_uri パラメータを含む logout リクエストには、client_id、response_type、scope などの ログインエンドポイント への後続のリクエスト用のパラメータも含める必要があります。

ログアウトエンドポイントは、顧客とのインタラクティブなユーザーセッション用のフロントエンドウェブアプリケーションです。アプリは、このエンドポイントと他のホストされた UI エンドポイントをユーザーのブラウザで呼び出す必要があります。

選択したページにユーザーをリダイレクトするには、アプリクライアントに許可されているサインアウト URL を追加します。logout エンドポイントへのユーザーのリクエストに、logout_uri および client_id パラメーターを追加します。logout_uri の値が、許可されているサインアウト URL の 1 つである場合、Amazon Cognito はユーザーをその URL にリダイレクトします。

SAML 2.0 のシングルログアウト (SLO) では IdPs、Amazon Cognito はまず IdP 設定で定義した SLO エンドポイントにユーザーをリダイレクトします。IdP がユーザーを にリダイレクトするとsaml2/logout、Amazon Cognito はリクエストlogout_uriから redirect_uriまたは へのリダイレクトをもう 1 つ返します。詳細については、「SAML サインアウトフロー」を参照してください。

ログアウトエンドポイントは、OIDC またはソーシャル ID プロバイダー () からユーザーをサインアウトしませんIdPs。外部 IdP とのセッションからユーザーをサインアウトするには、そのプロバイダーのサインアウトページに移動します。

GET /logout

/logout エンドポイントは HTTPS GET のみをサポートします。ユーザープールクライアントは通常、このリクエストをシステムブラウザ経由で行います。ブラウザは通常、Android では Custom Chrome Tab、iOS では Safari View Control です。

リクエストパラメータ

client_id

アプリのアプリクライアント ID。アプリクライアント ID を取得するには、ユーザープールにアプリを登録する必要があります。詳細については、「ユーザープールアプリクライアント」を参照してください。

必須。

logout_uri

logout_uri パラメータを使用して、ユーザーをカスタムサインアウトページにリダイレクトします。この値を、ユーザーがサインアウトした後にユーザーをリダイレクトするアプリクライアントの[sign-out URL] (サインアウト URL) に設定します。logout_uri は、client_id パラメータでのみ使用してください。詳細については、「ユーザープールアプリクライアント」を参照してください。

logout_uri パラメータを使用して、ユーザーを別のアプリクライアントのサインインページにリダイレクトすることもできます。他のアプリクライアントのサインインページを、アプリクライアントで許可されたコールバック URL として設定します。`/logout` エンドポイントへのリクエストで、logout_uri パラメータの値を URL エンコードされたサインインページに設定します。

Amazon Cognito では、`/logout` エンドポイントへのリクエストに logout_uri または redirect_uri パラメータのいずれかが必要です。logout_uri パラメータは、ユーザーを別のウェブサイトにリダイレクトします。`/logout` エンドポイントへのリクエストに logout_uri パラメータと redirect_uri パラメータの両方が含まれている場合、Amazon Cognito は logout_uri パラメータのみを使用し、redirect_uri パラメータをオーバーライドします。

redirect_uri

redirect_uri パラメータを使用して、ユーザーをサインインページにリダイレクトし、認証を行います。その値を、サインインした後にユーザーをリダイレクトするアプリクライアントの[Allowed callback URL] (許可されたコールバック URL) に設定します。`/login` エンドポイントに渡す client_id、scope、state、response_type のパラメータを追加します。

Amazon Cognito では、`/logout` エンドポイントへのリクエストに logout_uri または redirect_uri パラメータのいずれかが必要です。ユーザーを`/login`エンドポイントにリダイレクトしてトークンを再認証し、アプリに渡すには、redirect_uri パラメータを追加します。logout_uri パラメータと redirect_uri パラメータの両方が`/logout`エンドポイントへのリクエストに含まれている場合、Amazon Cognito は redirect_uri パラメータをオーバーライドし、logout_uri パラメータのみを処理します。

response_type

ユーザーがサインインした後に Amazon Cognito から受信する OAuth 2.0 レスポンス。`code` と `token` は、response_type パラメータの有効な値です。

redirect_uri パラメータを使用する場合は必須です。

state

アプリケーションがリクエストに状態パラメータを追加すると、/oauth2/logoutエンドポイントがユーザーをリダイレクトするときに Amazon Cognito はその値をアプリケーションに返します。

この値をリクエストに追加して CSRF 攻撃から保護します。

state パラメータの値を、URL でエンコードされた JSON 文字列に設定することはできません。state パラメータでこの形式に一致する文字列を渡すには、文字列を base64 にエンコードしてから、アプリケーションでデコードします。

redirect_uri パラメータを使用する場合は強くお勧めします。

scope

redirect_uri パラメータを使用してサインアウトした後に Amazon Cognito にリクエストする OAuth 2.0 スコープ。Amazon Cognito は、/logout エンドポイントへのリクエストの scope パラメータを使用してユーザーを /login エンドポイントにリダイレクトします。

redirect_uri パラメータを使用する場合は任意。scope パラメータを含めない場合、Amazon Cognito は scope パラメータを使用してユーザーを /login エンドポイントにリダイレクトします。Amazon Cognito がユーザーをリダイレクトし、scope に自動的にデータを入力する場合、パラメータには、アプリクライアントに対して許可されているすべてのスコープが含まれます。

リクエストの例

例 - ログアウトしてユーザーをクライアントにリダイレクトする

logout_uri と を除きclient_id、このエンドポイントで使用できるすべてのクエリパラメータはに渡されます認可エンドポイント。Amazon Cognito は、リクエストに logout_uri と client_id が含まれている場合、他のすべてのリクエストパラメータを無視して、値が logout_uri である URL にユーザーセッションをリダイレクトします。この URL は、アプリケーションクライアントの承認済みサインアウト URL である必要があります。

次は、サインアウトと https://www.example.com/welcome へのリダイレクトのリクエスト例です。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

例 - ログアウトし、別のユーザーとしてサインインするようにユーザーに求める

リクエストに logout_uri が省略されているが、許可エンドポイントへの正しい形式のリクエストを構成するパラメータが指定されている場合、Amazon Cognito はホストされた UI サインインにユーザーをリダイレクトします。ログアウトエンドポイントは、元のリクエストのパラメータをリダイレクト先に追加します。ログアウトエンドポイントへのリクエストの redirect_uri パラメータは、サインアウト URL ではなく、承認エンドポイントにパススルーするサインイン URL です。

以下は、ユーザーをサインアウトし、サインインページにリダイレクトし、サインインhttps://www.example.com後に認証コードを に提供するリクエストの例です。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

OAuth 2.0、OpenID Connect、および SAML 2.0 フェデレーションエンドポイントリファレンス

Amazon Cognito はユーザープールにドメインを追加すると、このセクションのエンドポイントをアクティブにします。フェデレーションエンドポイントはユーザーインタラクティブではありません。アプリケーションがサードパーティーの OAuth 2.0、OIDC、SAML 2.0 ID プロバイダー (IdPs) と通信するためのサービスロールを実行します。

このガイドのトピックでは、頻繁に使用される OAuth 2.0 および OIDC エンドポイントをいくつか説明します。Amazon Cognito はユーザープールにドメインを割り当てる際に以下のエンドポイントを作成します。

ユーザープールのフェデレーションエンドポイント

エンドポイント URL	説明	アクセス方法
https://############ / oauth2/authorize	ユーザーをホストされた UI にリダイレクトするか、IdP でサインインするようにリダイレクトします。	ユーザー認証を開始するためにカスタマーブラウザで呼び出されます。認可エンドポイント を参照してください。

エンドポイント URL	説明	アクセス方法
https://########### / oauth2/token	認証コードまたはクライアント認証情報リクエストに基づいてトークンを返します。	トークンを取得するためにアプリからリクエストされます。トークンエンドポイントを参照してください。
https://########### / oauth2/userInfo	OAuth 2.0 のスコープとアクセストークンのユーザー ID に基づいてユーザー属性を返します。	ユーザープロファイルを取得するためにアプリからリクエストされます。UserInfo エンドポイント を参照してください。
https://########### / oauth2/revoke	更新トークンと関連するアクセストークンを取り消します。	トークンの取り消しをアプリケーションからリクエストされました。エンドポイントの取り消し を参照してください。
https://cognito-idp.#### #.amazonaws.com/####### *ID*/.well-known/openid-configuration	ユーザープールの OIDC アーキテクチャのディレクトリ。	ユーザープール発行者のメタデータを見つけるためにアプリからリクエストされます。
https://cognito-idp.#### #.amazonaws.com/####### *ID*/.well-known/jwks.json	Amazon Cognito トークンの検証に使用できるパブリックキー。	JWTsを検証するためにアプリからリクエストされます。
https://########### / oauth2/idpresponse	ソーシャル ID プロバイダーは、認証コードを使用してユーザーをこのエンドポイントにリダイレクトする必要があります。Amazon Cognito は、フェデレーションユーザーを認証する際に、このコードをトークンに引き換えます。	OIDC IdP サインインから IdP クライアントのコールバック URL としてリダイレクトされます。

エンドポイント URL	説明	アクセス方法
https://########## /saml2/idpresponse	SAML 2.0 ID プロバイダーと統合するためのアサーションコンシューマーレスポンス (ACS) URL。	SAML 2.0 IdP から ACS URL、または IdP が開始するサインイン の送信元ポイントとしてリダイレクトされます[1]。
https://########## /saml2/logout	SAML 2.0 ID プロバイダーと統合するための Single Logout (SLO) URL。	シングルログアウト (SLO) URL として SAML 2.0 IdP からリダイレクトされます。POST バインディングのみを受け入れます。

[1] IdP が開始する SAML サインインの詳細については、「」を参照してくださいIdP が開始する SAML サインインの使用。

OpenID および OAuth 標準の詳細については、「OpenID 1.0」および「OAuth 2.0」を参照してください。

トピック

- 認可エンドポイント
- トークンエンドポイント
- UserInfo エンドポイント
- エンドポイントの取り消し
- saml2/idpresponse エンドポイント

認可エンドポイント

`/oauth2/authorize` エンドポイントは、2 つのリダイレクト先をサポートするリダイレクトエンドポイントです。URL に `identity_provider` または `idp_identifier` のパラメータを含めると、ユーザーはその ID プロバイダー (IdP) のサインインページにそのままリダイレクトされます。それ以外の場合は、リクエストに含まれるものと同じ URL パラメータを持つ ログインエンドポイント にリダイレクトされます。

認証エンドポイントは、ホストされている UI または IdP サインインページのどちらかにリダイレクトされます。このエンドポイントにおけるユーザーセッションの送信先は、ユーザーがブラウザで直接操作する必要があるウェブページです。

認証エンドポイントを使用するには、ユーザープールに以下のユーザープールの詳細に関する情報を提供するパラメータを使用して、ユーザーのブラウザを `/oauth2/authorize` で呼び出します。

- サインインするアプリクライアント。
- 終了させたいコールバック URL。
- ユーザーのアクセストークンでリクエストする OAuth 2.0 スコープ。
- サインインに使用するサードパーティーの IdP (任意)。

また、Amazon Cognito が受信クレームの検証に使用する `state` および `nonce` パラメータを提供することもできます。

GET **`/oauth2/authorize`**

`/oauth2/authorize` エンドポイントは `HTTPS GET` のみをサポートします。アプリは通常、このリクエストをユーザーのブラウザで開始します。`/oauth2/authorize` エンドポイントへのリクエストは、HTTPS でのみ行うことができます。

OpenID Connect (OIDC) 標準における認可エンドポイントの定義の詳細については、「Authorization Endpoint」(認可エンドポイント) を参照してください。

リクエストパラメータ

`response_type`

(必須) レスポンスタイプ。`code` または `token` を指定する必要があります。

`response_type` が `code` のリクエストに成功すると、認証コード付与を返します。認証コード付与とは、Amazon Cognito がリダイレクト URL に追加する `code` パラメータです。アプリはトークンエンドポイント と、アクセス、ID、更新の各トークンとコードを交換することができます。セキュリティ上のベストプラクティスとして、またユーザーに更新トークンを受け取るには、アプリで認証コード付与を使用します。

`response_type` が `token` のリクエストに成功すると、暗黙的な付与を返します。暗黙的な付与とは、Amazon Cognito がリダイレクト URL に追加する ID とアクセストークンのことです。暗黙的な付与は、トークンと潜在的な識別情報をユーザーに公開するため、安全性が低くなります。アプリクライアントの設定で、暗黙的な付与のサポートを無効にできます。

client_id

(必須) アプリクライアント ID。

client_id の値は、リクエストを行うユーザープール内のアプリクライアントの ID である必要があります。アプリクライアントは、Amazon Cognito ローカルユーザーまたは少なくとも 1 つのサードパーティー IdP によるサインインをサポートしている必要があります。

redirect_uri

(必須) Amazon Cognito がユーザーを承認した後に認証サーバーがブラウザをリダイレクトする URL。

リダイレクト Uniform Resource Identifier (URI) には次の属性が必要です。

- 絶対 URI である。
- URI を事前にクライアントに登録しておく必要があります。
- フラグメントコンポーネントが含まれていない。

「OAuth 2.0 - リダイレクトエンドポイント」を参照してください。

Amazon Cognito では、テスト目的のコールバック URL として設定できる http://localhost を除き、リダイレクト URI に HTTPS を使用することが必要です。

Amazon Cognito では、myapp://example のようなアプリのコールバック URL もサポートしています。

state

(オプション、推奨) アプリケーションがリクエストに状態パラメータを追加すると、/oauth2/authorizeエンドポイントがユーザーをリダイレクトするときに Amazon Cognito はその値をアプリケーションに返します。

この値をリクエストに追加して CSRF 攻撃から保護します。

state パラメータの値を、URL でエンコードされた JSON 文字列に設定することはできません。state パラメータでこの形式に一致する文字列を渡すには、文字列を base64 にエンコードしてから、アプリケーションでデコードします。

identity_provider

(オプション) このパラメータを追加して、ホストされた UI をバイパスし、ユーザーをプロバイダーのサインインページにリダイレクトします。identity_provider パラメータの値はユーザープールに表示される ID プロバイダー (IdP) の名前です。

- ソーシャルプロバイダーの場合、ID_provider 値 `Facebook`、`Google`、`LoginWithAmazon`、および を使用できます`SignInWithApple`。
- Amazon Cognito ユーザープールの場合は、値 を使用します`COGNITO`。
- SAML 2.0 および OpenID Connect (OIDC) ID プロバイダー (IdPs) の場合は、ユーザープールの IdP に割り当てた名前を使用します。

`idp_identifier`

(オプション) このパラメータを追加して、identity_provider 名の代替名を持つプロバイダーにリダイレクトします。Amazon Cognito コンソールのサインインエクスペリエンスタブ IdPs から SAML 2.0 と OIDC の識別子を入力できます。

`scope`

(オプション) クライアントに関連付けられているシステム予約スコープまたはカスタムスコープの組み合わせにすることができます。スコープはスペースで区切る必要があります。システム予約されたスコープは `openid`、`email`、`phone`、`profile`、および `aws.cognito.signin.user.admin` です。使用されるスコープは、いずれもクライアントに関連付けられている必要があり、関連付けられていなければ、ランタイム時に無視されます。

クライアントがスコープをリクエストしない場合、認証サーバーはクライアントに関連付けられているすべてのスコープを使用します。

ID トークンは `openid` スコープがリクエストされた場合にのみ返されます。Amazon Cognito ユーザープールに対してアクセストークンを使用できるのは、`aws.cognito.signin.user.admin` スコープがリクエストされている場合のみです。`phone`、`email`、および `profile` スコープは、`openid` スコープがリクエストされた場合にのみリクエストできます。これらのスコープは ID トークン内でクレームを記述します。

`code_challenge_method`

(オプション) チャレンジの生成に使用したハッシュプロトコル。PKCE RFC では S256 および plain の 2 つのメソッドが定義されていますが、Amazon Cognito 認証サーバーでは S256 のみがサポートされています。

`code_challenge`

(オプション) から生成したチャレンジ`code_verifier`。

`code_challenge_method` パラメータを指定した場合にのみ必須です。

nonce

(オプション) リクエストに追加できるランダムな値。指定したノンス値は、Amazon Cognito が発行する ID トークンに含まれます。リプレイ攻撃を防ぐために、アプリは ID トークンの nonce クレームを検査し、生成したものと比較することができます。nonce クレームの詳細については、「OpenID Connect standard」(OpenID Connect 標準) の「ID token validation」(ID トークンの検証) を参照してください。

肯定的なレスポンスを持つリクエストの例

次の例は、/oauth2/authorizeエンドポイントへの HTTP リクエストの形式を示しています。

認証コード付与

これは、認証コード付与のリクエスト例です。

例 – GET リクエスト

次のリクエストは、ユーザーがredirect_uri送信先でアプリケーションに渡す認証コードを取得するセッションを開始します。このセッションは、ユーザー属性と Amazon Cognito セルフサービス API オペレーションへのアクセスの範囲をリクエストします。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

例 – レスポンス

Amazon Cognito 認証サーバーは、承認コードとステートを伴ってリダイレクトしアプリに戻ります。認証コードは 5 分間有効です。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

PKCE を使用した認可コード付与

これは、PKCE を使用した認証コード付与のリクエスト例です。

例 - GET リクエスト

次のリクエストは、前のリクエストに code_challenge パラメータを追加します。トークンのコード交換を完了するには、/oauth2/tokenエンドポイントへのリクエストに code_verifierパラメータを含める必要があります。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=a1b2c3d4...
```

例 – レスポンス

認証サーバーは、認証コードと状態を使用してアプリケーションにリダイレクトします。コードと状態は、フラグメントではなくクエリ文字列パラメータで返される必要があります。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

openid スコープを使用しないトークン付与

これは、暗黙的な許可を生成し、JWTs をユーザーのセッションに直接返すリクエストの例です。

例 – GET リクエスト

次のリクエストは、認証サーバーからの暗黙的な付与用です。Amazon Cognito からのアクセストークンは、セルフサービス API オペレーションを承認します。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

例 – レスポンス

Amazon Cognito 認証サーバーはアクセストークンを伴ってリダイレクトし、アプリに戻ります。openid スコープがリクエストされなかったため、Amazon Cognito は ID トークンを返しません。また、Amazon Cognito はこのフローで更新トークンを返しません。Amazon Cognito は、クエリ文字列ではなくフラグメントでアクセストークンと状態を返します。

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_uri#access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

openid スコープを使用したトークン付与

これは、暗黙的な許可を生成し、JWTs をユーザーのセッションに直接返すリクエストの例です。

例 – GET リクエスト

次のリクエストは、認証サーバーからの暗黙的な付与用です。Amazon Cognito からのアクセストークンは、ユーザー属性とセルフサービス API オペレーションへのアクセスを許可します。

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin+openid+profile
```

例 – レスポンス

認証サーバーは、アクセストークンと ID トークンを使用してアプリケーションにリダイレクトバックします (openidスコープが含まれていたため)。

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

否定応答の例

Amazon Cognito はリクエストを拒否することがあります。負のリクエストには、HTTP エラーコードと、リクエストパラメータの修正に使用できる説明が付属しています。以下は、否定的なレスポンスの例です。

- client_id と redirect_uriが有効で、リクエストパラメータが正しくフォーマットされていない場合、認証サーバーはエラーをクライアントの にリダイレクトredirect_uriし、URL パラメータにエラーメッセージを追加します。以下は、誤ったフォーマットの例です。
  - リクエストにはresponse_typeパラメータは含まれません。
  - 認証リクエストは code_challengeパラメータを提供しましたが、code_challenge_methodパラメータは提供していません。
  - code_challenge_method パラメータの値は ではありませんS256。

  以下は、誤ったフォーマットのリクエスト例に対するレスポンスです。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
  ```

- クライアントが tokenで codeまたは をリクエストしてもresponse_type、これらのリクエストに対するアクセス許可がない場合、Amazon Cognito 認証サーバーはredirect_uri次のようにクライアントの unauthorized_clientに戻ります。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=unauthorized_client
  ```

- クライアントが無効、不明、または有効ではない形式のスコープをリクエストする場合は、以下にあるように、Amazon Cognito 認証サーバーが invalid_scope をクライアントの redirect_uri に返します。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
  ```

- サーバーに予期しないエラーがある場合、認証サーバーはクライアントの server_errorに戻りますredirect_uri。HTTP 500 エラーはクライアントに送信されないため、エラーはユーザーのブラウザに表示されません。認証サーバーは次のエラーを返します。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
  ```

- Amazon Cognito がサードパーティーの へのフェデレーションを通じて認証する場合IdPs、Amazon Cognito では次のような接続の問題が発生する可能性があります。
  - IdP からトークンをリクエストしているときに接続タイムアウトが発生した場合、認証サーバーは以下のようにエラーをクライアントの redirect_uri にリダイレクトします。

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Timeout+occurred+in+calling+IdP+token
    +endpoint
    ```

- ID トークンの検証のためにjwks_uriエンドポイントを呼び出すときに接続タイムアウトが発生すると、認証サーバーは次のようにエラーでクライアントの にリダイレクトしますredirect_uri。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=error_description=Timeout+in+calling+jwks
+uri
```

- サードパーティーの にフェデレーションして認証する場合 IdPs、プロバイダーはエラーレスポンスを返すことがあります。これは、設定エラーや、次のようなその他の理由が原因である可能性があります。
  - 他のプロバイダーからエラーレスポンスを受け取った場合、認証サーバーは以下のようにエラーをクライアントの redirect_uri にリダイレクトします。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=[IdP name]+Error+-+[status code]+error
 getting token
```

  - Google からエラーレスポンスを受け取った場合、認証サーバーは以下のようにエラーをクライアントの redirect_uri にリダイレクトします。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Google+Error+-+[status code]+[Google-
provided error code]
```

- Amazon Cognito が外部 IdP に接続するときに通信例外が発生すると、認証サーバーは、次のいずれかredirect_uriのメッセージを使用してエラーでクライアントの にリダイレクトします。
  -

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Connection+reset
```

  -

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Read+timed+out
```

トークンエンドポイント

/oauth2/token の OAuth 2.0 トークンエンドポイントは、JSON ウェブトークン (JWT) を発行します。

ユーザープール OAuth 2.0 認証サーバーは、トークンエンドポイントから次のタイプのセッションに JWTs) を発行します。

1. 認可コード付与のリクエストを完了したユーザー。コードの引き換えに成功すると、ID、アクセス、更新の各トークンが返されます。
2. クライアント認証情報の付与を完了した M achine-to-machine (M2M) セッション。クライアントシークレットによる認証が成功すると、アクセストークンが返されます。
3. 以前にサインインして更新トークンを受け取ったことがあるユーザー。更新トークン認証は、新しい ID トークンとアクセストークンを返します。

> **Note**
>
> ホストされた UI またはフェデレーションで認証コード付与を使用してサインインするユーザーは、トークンエンドポイントからトークンを常に更新できます。API オペレーションでサインイン`InitiateAuth`し、記憶されたデバイスがユーザープールでアクティブでない場合にトークンエンドポイントでトークンを更新`AdminInitiateAuth`できるユーザー。記憶されたデバイスがアクティブの場合は、 `InitiateAuth`または `AdminInitiateAuth` API リクエスト`REFRESH_TOKEN_AUTH`の `AuthFlow` の を使用してトークンを更新します。

ユーザープールにドメインを追加すると、トークンエンドポイントが公開されます。HTTP POST リクエストを受け入れます。アプリケーションセキュリティのために、認証コードのサインインイベントで PKCE を使用します。PKCE は、認証コードを渡すユーザーが、認証したユーザーと同じであることを確認します。PKCE の詳細については、「IETF RFC 7636」を参照してください。

ユーザープールアプリクライアントと付与タイプ、クライアントシークレット、許可されたスコープ、クライアント ID の詳細については、「ユーザープールアプリクライアント」を参照してください。M2M 認証、クライアント認証情報の付与、およびアクセストークンスコープによる認証の詳細については、「」を参照してくださいスコープ、M2M 、およびリソースサーバーによる API 認証。

アクセストークンからユーザーに関する情報を取得するには、それを UserInfo エンドポイントまたは GetUser API リクエストに渡します。

POST /oauth2/token

`/oauth2/token` エンドポイントは `HTTPS POST` のみをサポートします。ユーザープールクライアントは、ブラウザ経由ではなく、このエンドポイントに直接リクエストを送信します。

トークンエンドポイントは client_secret_basic と client_secret_post をサポートしています。OpenID Connect 仕様の詳細については、「クライアント認証」を参照してください。OpenID Connect 仕様のトークンエンドポイントの詳細については、「トークンエンドポイント」を参照してください。

ヘッダーのリクエストパラメータ

Authorization

クライアントがシークレットで発行された場合、クライアントはその client_id および client_secret をベーシック client_secret_basic 認証として認証ヘッダーに渡す必要があります。また、client_id と client_secret を client_secret_post 認証としてリクエスト本文に含めることもできます。

認証ヘッダー文字列は Basic Base64Encode(client_id:client_secret) です。次の例は、Base64-encodedバージョンの文字列 を使用したabcdef01234567890、クライアントシークレット djc98u3jiedmi283eu928を持つアプリケーションクライアントの認証ヘッダーですdjc98u3jiedmi283eu928:abcdef01234567890。

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

このパラメータの値を 'application/x-www-form-urlencoded' に設定します。

本文のリクエストパラメータ

grant_type

(必須) リクエストする OIDC 許可のタイプ。

authorization_code、refresh_token、または client_credentials を指定する必要があります。次の条件で、トークンエンドポイントからカスタムスコープのアクセストークンをリクエストできます。

- アプリケーションクライアント設定でリクエストされたスコープを有効にしました。
- クライアントシークレットを使用してアプリケーションクライアントを設定しました。
- アプリケーションクライアントでクライアント認証情報の付与を有効にします。

client_id

(オプション) ユーザープール内のアプリケーションクライアントの ID。ユーザーを認証したのと同じアプリクライアントを指定します。

クライアントがパブリックで、シークレットがない場合、または がclient_secret_post認可client_secretされている場合は、このパラメータを指定する必要があります。

client_secret

(オプション) ユーザーを認証したアプリクライアントのクライアントシークレット。アプリケーションクライアントがクライアントシークレットを持っていて、Authorization ヘッダーを送信しなかった場合は必須です。

scope

(オプション) アプリケーションクライアントに関連付けられている任意のカスタムスコープの組み合わせにすることができます。リクエストするスコープは、アプリケーションクライアントに対してアクティブ化する必要があります。そうでない場合、Amazon Cognito はそれを無視します。クライアントがスコープをリクエストしない場合、認証サーバーはアプリクライアント設定で承認したすべてのカスタムスコープを割り当てます。

grant_type が client_credentials である場合にのみ使用されます。

redirect_uri

(オプション) authorization_codeでの取得に使用されたredirect_uriものと同じである必要があります/oauth2/authorize。

が の場合grant_type、このパラメータを指定する必要がありますauthorization_code。

refresh_token

(オプション) ユーザーのセッションの新しいアクセストークンと ID トークンを生成するには、/oauth2/tokenリクエストの refresh_token パラメータの値を、同じアプリケーションクライアントから以前に発行された更新トークンに設定します。

code

(オプション) 認証コード付与からの認証コード。認証リクエストに grant_typeの が含まれている場合は、このパラメータを指定する必要がありますauthorization_code。

code_verifier

(オプション) PKCE を使用した認証コード付与リクエストcode_challengeで の計算に使用した任意の値。

肯定的なレスポンスを持つリクエストの例

トークン用の認可コードの交換

例 – POST リクエスト

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=authorization_code&
                            client_id=1example23456789&
                            code=AUTHORIZATION_CODE&
                            redirect_uri=com.myclientapp://myclient/redirect
```

例 - レスポンス

```
HTTP/1.1 200 OK

                           Content-Type: application/json

                           {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "refresh_token":"eyJj3example",
                            "token_type":"Bearer",
                            "expires_in":3600
                           }
```

> **Note**
>
> トークンエンドポイントは、grant_type が authorization_code の場合にのみ refresh_token を返します。

アクセストークンのクライアント認証情報の交換: 認証ヘッダーのクライアントシークレット

例 – POST リクエスト

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
```

```
                            Content-Type='application/x-www-form-urlencoded'&

Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=client_credentials&
                            client_id=1example23456789&

scope=resourceServerIdentifier1/scope1 resourceServerIdentifier2/scope2
```

例 - レスポンス

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

アクセストークンのクライアント認証情報の交換: リクエスト本文のクライアントシークレット

例 – POST リクエスト

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&client_id=1example23456789&scope=my_resource_server_identifier%2F
```

例 - レスポンス

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
"access_token": "eyJra12345EXAMPLE",
"expires_in": 3600,
"token_type": "Bearer"
}
```

トークン用の PKCE を使用した認可コード付与の交換

例 - POST リクエスト

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
                                Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                                grant_type=authorization_code&
                                client_id=1example23456789&
                                code=AUTHORIZATION_CODE&
                                code_verifier=CODE_VERIFIER&
                                redirect_uri=com.myclientapp://myclient/redirect
```

例 - レスポンス

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "refresh_token":"eyJj3example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

> **Note**
>
> トークンエンドポイントは、refresh_token が grant_type の場合にのみ authorization_code を返します。

トークン用の更新トークンの交換

例 – POST リクエスト

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=refresh_token&
                            client_id=1example23456789&
                            refresh_token=eyJj3example
```

例 – レスポンス

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "token_type":"Bearer",
                            "expires_in":3600
                                  }
```

> **Note**
>
> トークンエンドポイントは、grant_type が authorization_code の場合にのみ refresh_token を返します。

否定応答の例

例 - エラーレスポンス

```
HTTP/1.1 400 Bad Request
                            Content-Type: application/json;charset=UTF-8

                            {
                            "error":"invalid_request|invalid_client|invalid_grant|
unauthorized_client|unsupported_grant_type"
                            }
```

invalid_request

リクエストに必須パラメータが含まれていない、サポートされていないパラメータ値 (unsupported_grant_type 以外) が含まれている、または正しい形式ではありません。たとえば、grant_type が refresh_token であるが、refresh_token が含まれていない、などです。

invalid_client

クライアントの認証に失敗しました。たとえば、クライアントが認証ヘッダーに client_id と client_secret を含めたが、その client_id と client_secret を持つクライアントが存在しない場合です。

invalid_grant

更新トークンが失効しています。

認可コードが既に消費されているか、存在しません。

アプリクライアントにはリクエストされたスコープ内すべての属性への読み取りアクセス権はありません。例えば、アプリが email スコープをリクエストすると、アプリクライアントは email 属性の読み取りはできますが、email_verified は読み取れません。

unauthorized_client

クライアントがコード付与フローまたはトークンの更新を許可されていません。

unsupported_grant_type

grant_type が authorization_code、refresh_token、または client_credentials 以外の場合に返されます。

UserInfo エンドポイント

`userInfo` エンドポイントは OpenID Connect (OIDC) userInfo エンドポイントです。トークンエンドポイント が発行したアクセストークンをサービスプロバイダーが提示すると、ユーザー属性で応答します。ユーザーのアクセストークンのスコープは、userInfo エンドポイントがレスポンスで返すユーザー属性を定義します。`openid` スコープはアクセストークンクレームのいずれかである必要があります。

Amazon Cognito は、InitiateAuth などのユーザープール API リクエストに応答してアクセストークンを発行します。スコープが含まれていないため、userInfo エンドポイントはこれらのアクセストークンを受け入れません。代わりに、トークンエンドポイントからアクセストークンを提示する必要があります。

OAuth 2.0 サードパーティーの ID プロバイダー (IdP) は、userInfo エンドポイントもホストしています。ユーザーがその IdP で認証すると、Amazon Cognito は認証コードを IdP `token`エンドポイントとサイレントに交換します。ユーザープールは IdP アクセストークンを渡して、IdP `userInfo`エンドポイントからのユーザー情報の取得を許可します。

GET /oauth2/userInfo

アプリは、ブラウザ経由ではなく、このエンドポイントに直接リクエストを送信します。

詳細については、OpenID Connect (OIDC) 仕様の「UserInfo エンドポイント」を参照してください。

トピック

- ヘッダーのリクエストパラメータ
- 例 – リクエスト
- 例 - 肯定的なレスポンス
- ネガティブレスポンスの例

ヘッダーのリクエストパラメータ

`Authorization: Bearer` ***`<access_token>`***

認証ヘッダーフィールドにアクセストークンを渡します。

必須。

例 – リクエスト

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

例 - 肯定的なレスポンス

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
Connection: keep-alive
{
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
    "username": "bob"
}
```

OIDC クレームのリストについては、「スタンダードクレーム」を参照してください。現在、Amazon Cognito は、email_verified と phone_number_verified の値を文字列として返します。

ネガティブレスポンスの例

例 - 不正なリクエスト

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalid_request",
error_description="Bad OAuth2 request at UserInfo Endpoint"
```

invalid_request

リクエストに必須パラメータがないか、サポートされていないパラメータ値が含まれているか、形式が正しくありません。

例 — 不正なトークン

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="Access token is expired, disabled, or deleted, or the user has
 globally signed out."
```

invalid_token

アクセストークンの有効期限が切れているか、取り消されているか、形式が正しくないか、無効です。

エンドポイントの取り消し

/oauth2/revoke エンドポイントは、指定した更新トークンを使用して Amazon Cognito が最初に発行したユーザーのアクセストークンを取り消します。このエンドポイントは、同じ更新トークンから後続のすべてのアクセストークンと ID トークンも取り消します。エンドポイントがトークンを取り消した後、取り消されたトークンを使用して Amazon Cognito トークンが認証する API にアクセスすることはできません。

POST/oauth2/revoke

/oauth2/revoke エンドポイントは HTTPS POST のみをサポートします。ユーザープールクライアントは、システムブラウザ経由ではなくこのエンドポイントに直接リクエストを送信します。

ヘッダーのリクエストパラメータ

Authorization

アプリケーションクライアントにクライアントシークレットがある場合、アプリケーションは基本的な HTTP 認証を介して認証ヘッダーclient_secretで client_idと を渡す必要があります。シークレットはベーシック Base64Encode(client_id:client_secret) です。

Content-Type

常に 'application/x-www-form-urlencoded' にする必要があります。

本文のリクエストパラメータ

token

(必須) クライアントが取り消す更新トークン。このリクエストは、Amazon Cognito がこの更新トークンで発行したすべてのアクセストークンも無効にします。

必須。

client_id

(オプション) 取り消すトークンのアプリケーションクライアント ID。

クライアントがパブリックでありシークレットがない場合は必須です。

取り消しリクエストの例

例 1: クライアントシークレットのないアプリクライアントのトークンを取り消す

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    token=2YotnFZFEjr1zCsicMWpAA&
    client_id=djc98u3jiedmi283eu928
```

例 2: クライアントシークレットを持つアプリクライアントのトークンを取り消す

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    token=2YotnFZFEjr1zCsicMWpAA
```

取り消しエラーレスポンス

正常なレスポンスには空のボディが含まれています。エラーレスポンスは、`error` フィールドを持つ JSON オブジェクトで、`error_description` フィールドがある場合もあります。

エンドポイントエラー

- リクエストにトークンが存在しない場合、またはアプリケーションクライアントに対して機能が無効化されている場合に、HTTP 400 とエラー `invalid_request` が返されます。
- Amazon Cognito が取り消しリクエストで送信したトークンが更新トークンでない場合は、HTTP 400 とエラー `unsupported_token_type` が発生します。
- クライアントの認証情報が有効でない場合、HTTP 401 とエラー `invalid_client` が発生します。
- トークンが取り消されている、またはクライアントが無効なトークンを送信した場合は、HTTP 200 OK が発生します。

saml2/idpresponse エンドポイント

は SAML アサーション`/saml2/idpresponse`を受け取ります。 service-provider-initiated (SP 開始) サインインでは、SAML 2.0 ID プロバイダー (IdP) は SAML レスポンスを使用してユーザーをこのエンドポイントにリダイレクトします。SP 開始のサインインでは、アプリケーションはこのエンドポイントとやり取りしません。アサーションコンシューマーサービス (ACS) URL `saml2/idpresponse`として へのパスを使用して IdP を設定します。セッション開始の詳細については、「」を参照してくださいAmazon Cognito ユーザープールでの SAML セッション開始。

IdP が開始するサインインでは、ユーザーは独自のプロセスを通じて IdP でサインインし、HTTPS 経由でHTTP `POST`リクエストの本文に SAML アサーションを送信できます。`POST` リクエストの本文は、 `SAMLResponse`パラメータと `Relaystate`パラメータである必要があります。詳細については、「IdP が開始する SAML サインインの使用」を参照してください。

POST **/saml2/idpresponse**

IdP が開始するサインインで`/saml2/idpresponse`エンドポイントを使用するには、ユーザーのセッションに関する情報をユーザープールに提供するパラメータを含む POST リクエストを生成します。

- サインインするアプリケーションクライアント。
- 終了するコールバック URL。
- ユーザーのアクセストークンでリクエストする OAuth 2.0 スコープ。
- サインインリクエストを開始した IdP。

IdP が開始するリクエスト本文パラメータ

SAMLResponse

ユーザープール内の有効なアプリケーションクライアントと IdP IdP 設定に関連付けられた IdP からの Base64-encodedされた SAML アサーション。

RelayState

`RelayState` パラメータには、エンドポイントに渡すリクエストパラメータが含まれますoauth2/authorize。これらのパラメータの詳細については、「」を参照してください認可エンドポイント。

response_type

OAuth 2.0 許可タイプ。

client_id

アプリクライアント ID。

redirect_uri

Amazon Cognito がユーザーを認証した後、認証サーバーによってブラウザがリダイレクトされる URL です。

identity_provider

ユーザーをリダイレクトする ID プロバイダーの名前。

idp_identifier

ユーザーをリダイレクトする ID プロバイダーの識別子。

scope

ユーザーが認証サーバーにリクエストする OAuth 2.0 スコープ。

肯定的なレスポンスを持つリクエストの例

例 - POST リクエスト

次のリクエストは、アプリクライアント MySAMLIdPの IdP からのユーザー に対する認証コード付与用です1example23456789。ユーザーは認証コードhttps://www.example.comを使用して にリダイレクトします。認証コードは、OAuth 2.0 スコープ openid、email、および を持つアクセストークンを含むトークンと交換できますphone。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

例 - レスポンス

以下は、前のリクエストに対するレスポンスです。

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 許可

Amazon Cognito ユーザープール OAuth 2.0 認証サーバーは、3 種類の OAuth 2.0 認証付与に対応してトークンを発行します。ユーザープール内の各アプリクライアントに、サポートされる許可タイプを設定できます。クライアント認証情報許可は、暗黙または認証コード許可と同じアプリクライアントとして有効にできません。暗黙的コード付与と認証コード付与のリクエストは、認可エンドポイント で開始され、クライアント認証情報付与のリクエストは トークンエンドポイント で開始されます。

認証コード付与

認証リクエストが成功すると、認可サーバーは code の認証コードをコールバック URL のパラメータに追加します。次に、トークンエンドポイント と、アクセス、ID、更新トークンのコードを交換することができます。認証コード付与をリクエストするには、リクエストで response_type を code に設定します。リクエスト例については、「認証コード付与」を参照してください。

認証コード付与は、最も安全な認証付与の形式です。トークンの内容がユーザーに直接表示されることはありません。代わりに、ユーザーのトークンを取得して安全に保管する責任はアプリにあります。Amazon Cognito では、認証コード付与が 3 種類のトークン (ID、アクセス、更新) すべてを認証サーバーから取得する唯一の方法です。Amazon Cognito ユーザープール API による認証から 3 種類のトークンをすべて取得することもできますが、API は aws.cognito.signin.user.admin 以外のスコープを持つアクセストークンを発行しません。

暗黙の許可

認証リクエストが成功すると、認可サーバーは access_token パラメータのアクセストークンと、id_token パラメータの ID トークンを。コールバック URL に追加します。暗黙的な許可では、トークンエンドポイント に追加の操作は必要ありません。暗黙的な許可をリクエストするには、リクエストで response_type を token に設定します。暗黙的な許可では、ID とアクセストークンのみが生成されます。リクエスト例については、「openid スコープを使用しないトークン付与」を参照してください。

暗黙的な許可は、レガシーの認証付与方法です。認証コード付与とは異なり、ユーザーはトークンを傍受して検査できます。暗黙的な許可によるトークンの配信を防ぐには、認証コード付与のみをサポートするようにアプリクライアントを設定します。

クライアント認証情報

クライアント認証情報は、アクセスの認可のみの machine-to-machine許可です。クライアント認証情報許可を受けるには、認可エンドポイント をバイパスして、トークンエンドポイント に直接リクエストを生成してください。アプリクライアントにはクライアントシークレットが必要で、クライアント認証情報許可のみをサポートする必要があります。リクエストが成功すると、認証サーバーがアクセストークンを返します。

クライアント認証情報許可からのアクセストークンは、OAuth 2.0 スコープを含む認証メカニズムです。通常、トークンには、アクセス保護された API に対する HTTP 操作を許可するカスタムスコープクレームが含まれます。詳細については、「スコープ、M2M 、およびリソースサーバーによる API 認証」を参照してください。

クライアント認証情報の付与は、 AWS 請求書にコストを追加します。詳細については、「Amazon Cognito の料金」を参照してください。

Amazon Cognito ユーザープールでの認証コード付与での PKCE の使用

Amazon Cognito は、認証コード付与でコード交換 (PKCE) 認証の証明キーをサポートしています。PKCE は、パブリッククライアント向けの OAuth 2.0 認証コード付与を拡張したものです。PKCE は、傍受された認証コードが使用されるのを防ぎます。

Amazon Cognito が PKCE を使用する方法

PKCE で認証を開始するには、アプリケーションが一意の文字列値を生成する必要があります。この文字列は、Amazon Cognito が最初の認証許可をリクエストするクライアントとトークンの認証コードを交換するクライアントを比較するために使用するシークレット値であるコード検証子です。

アプリケーションは、コード検証文字列に SHA256 ハッシュを適用し、結果を base64 にエンコードする必要があります。ハッシュされた文字列をリクエスト本文のcode_challengeパラメータ認可エンドポイントとして に渡します。アプリが認証コードをトークンと交換するときは、 へのリクエスト本文にcode_verifierパラメータとしてプレーンテキストのコード検証文字列を含める必要がありますトークンエンドポイント。Amazon Cognito はコード検証子に対して同じ hash-and-encode 操作を実行します。Amazon Cognito は、コード検証によって認証リクエストで受け取ったのと同じコードチャレンジになると判断した場合のみ、ID、アクセス、および更新トークンを返します。

PKCE で認可付与フローを実装するには

1. Amazon Cognito コンソールを開きます。プロンプトが表示されたら、 AWS 認証情報を入力します。
2. [User Pools] (ユーザープール) を選択します。
3. リストから既存のユーザープールを選択するか、ユーザープールを作成します。ユーザープールを作成すると、ウィザード中にアプリケーションクライアントを設定し、ホストされた UI を設定するように求められます。

    a. 新しいユーザープールを作成する場合は、ガイド付きセットアップ中にアプリクライアントを設定し、ホストされた UI を設定します。

    b. 既存のユーザープールを設定する場合は、ドメインとパブリックアプリケーションクライアント を追加します。

4. PKCE のコードチャレンジを作成するために、ランダムな英数字文字列を生成します。通常はユニバーサル一意識別子 (UUID) です。この文字列は、リクエストで に送信するcode_verifierパラメータの値ですトークンエンドポイント。

5. SHA256 アルゴリズムを使用してcode_verifier文字列をハッシュします。ハッシュ操作の結果を base64 にエンコードします。この文字列は、 へのリクエストで送信するcode_challengeパラメータの値です認可エンドポイント。

   次のPython例では、 を生成code_verifierし、 を計算しますcode_challenge。

```
#!/usr/bin/env python3

import random
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use a cryptographically strong random number generator source
rand = random.SystemRandom()

code_verifier = ''.join(rand.choices(ascii_letters + digits, k=128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

   Python スクリプトからの出力例を次に示します。

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. PKCE を使用した認証コード付与リクエストを使用して、ホストされた UI サインインを完了します。URL の例を次に示します。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 認証を収集codeし、トークンエンドポイントを持つトークンと引き換えます。リクエストの例を次に示します。

```
POST /oauth2/token HTTP/1.1
Host: mydomain.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296

redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
code=7378f445-c87f-400c-855e-0297d072ff03&
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. レスポンスを確認します。これには、ID、アクセス、更新トークンが含まれます。Amazon Cognito ユーザープールトークンの使用の詳細については、「」を参照してくださいユーザープールでのトークンの使用。

ホストされた UI とフェデレーションエラーレスポンス

ホスト UI のサインインプロセスまたはフェデレーションログインでエラーが返されることがあります。認証がエラーで終了する原因となる条件は次のとおりです。

- ユーザープールでは実行できない操作を、ユーザーが実行する。
- Lambda トリガーが想定どおりの構文で応答しない。
- ID プロバイダー (IdP) がエラーを返す。
- Amazon Cognito は、ユーザーが提供した属性情報を検証できませんでした。
- IdP は、必須の属性に対応するクレームを送信しませんでした。

Amazon Cognito はエラーを検出すると、以下のいずれかの方法でエラーを通知します。

1. Amazon Cognito は、リクエストパラメータにエラーを含むリダイレクト URL を送信します。
2. Amazon Cognito はホストされた UI にエラーを表示します。

Amazon Cognito がリクエストパラメータに追加するエラーの形式は次のとおりです。

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

ユーザーが操作を実行できないときにエラー情報を送信できるようにする場合は、URL とテキスト、またはページのスクリーンショットをキャプチャするようにリクエストしてください。

> **Note**
>
> Amazon Cognito エラーの説明は固定文字列ではないため、固定のパターンや形式に依存するロジックは使用しないでください。

OIDC とソーシャル ID プロバイダーのエラーメッセージ

ID プロバイダー (IdP) は、エラーを返すことがあります。OIDC または OAuth 2.0 IdP が標準に準拠したエラーを返すと、Amazon Cognito はユーザーをコールバック URL にリダイレクトし、プロバイダーのエラーレスポンスをエラーリクエストパラメータに追加します。Amazon Cognito は、既存のエラー文字列にプロバイダー名と HTTP エラーコードを追加します。

次の URL は、エラーを返した IdP から Amazon Cognito へのリダイレクトの例です。

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request
+The+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

Amazon Cognito はプロバイダーから受け取ったもののみを返すため、ユーザーにはこの情報のサブセットが表示される場合があります。

IdP による初回ログインで問題が発生すると、IdP はすべてのエラーメッセージをユーザーに直接配信します。Amazon Cognito は、ユーザーセッションを検証するリクエストを IdP に生成すると、エラーメッセージをユーザーに中継します。Amazon Cognito は、以下のエンドポイントからの OAuth および OIDC IdP のエラーメッセージを中継します。

`/token`

Amazon Cognito が IdP の認証コードをアクセストークンと交換します。

`/.well-known/openid-configuration`

Amazon Cognito は発行者のエンドポイントへのパスを検出します。

`/.well-known/jwks.json`

ユーザーの JSON ウェブトークン (JWT) を検証するために、Amazon Cognito は IdP がトークンの署名に使用する JSON ウェブキー (JWK) を検出します。

Amazon Cognito は HTTP エラーを返す可能性のある SAML 2.0 プロバイダーへのアウトバウンドセッションを開始しないため、SAML 2.0 IdP とのセッション中のユーザーエラーには、この形式のプロバイダーエラーメッセージは含まれません。

Amazon Cognito ユーザープール API リファレンス

Amazon Cognito ユーザープールは、ウェブおよびモバイルアプリのユーザーによるサインアップとサインインを可能にします。認証されたユーザーのパスワードを変更し、未認証ユーザーがパスワードを忘れた場合のフローを開始することができます。詳細については、「ユーザープール認証フロー」および「ユーザープールでのトークンの使用」を参照してください。

Amazon Cognito ユーザープール API には、ユーザープールとユーザーを表示および変更し、ユーザーの認証と承認を実行するオペレーションが含まれています。Amazon Cognito ユーザープール API に統合される API オペレーションのクラスの説明については、「Amazon Cognito ユーザープール API とユーザープールのエンドポイントの使用」を参照してください。

Amazon Cognito ユーザープール API オペレーションと構文の詳細なリストについては、「Amazon Cognito ユーザープール API リファレンス」を参照してください。Amazon Cognito ユーザープール API リファレンスの各ページは、さまざまな AWS SDK の構文と例を含む参考資料へのリンクです。

Amazon Cognito ID プール (フェデレーティッド ID) API リファレンス

Amazon Cognito ID プールの使用により、ウェブおよびモバイルアプリケーションのユーザーは、権限が制限された一時的な AWS 認証情報を取得して AWS の他のサービスにアクセスすることができます。

完全な ID プール (フェデレーション ID) API リファレンスについては、「Amazon Cognito API リファレンス」を参照してください。

Amazon Cognito Sync API リファレンス

AWS のサービスである Amazon Cognito Sync は、アプリケーション関連のユーザーデータのデバイス間での同期を可能にするクライアントライブラリです。

Amazon Cognito Sync API リファレンスの詳細については、Amazon Cognito Sync API リファレンスを参照してください。

Amazon Cognito のドキュメント履歴

次の表は、Amazon Cognito のドキュメントへの重要な追加項目を示しています。また、お客様からいただいたフィードバックを反映して、ドキュメントの小さな更新を頻繁に行っています。フィードバックを送信するには、Amazon Cognito ドキュメントの任意のページで下部にあるフィードバックリンクを使用してください。

変更	説明	日付
トークン前の Lambda トリガーで複雑なオブジェクトのサポートを追加	ID およびアクセストークンクレームに配列と JSON オブジェクトを追加できるようになりました。	2024 年 5 月 30 日
Verified Permissions と Amazon Cognito に関する情報を更新しました。	Amazon Verified Permissions が Amazon Cognito とより直接統合されるようになりました。	2024 年 5 月 15 日
マルチリージョンの Amazon SES 検証済み ID。	Amazon SES AWS リージョン を使用しない一部の では、Amazon Cognito ユーザープールは 2 つのリモートリージョン間で E メールの負荷を分散します。 Amazon SES	2024 年 5 月 10 日
M2M 認証とコスト管理に関する情報を追加しました。	Amazon Cognito ユーザープールで machine-to-machine (M2M) ユースケースのクライアント認証情報付与を使用する方法について説明します。	2024 年 5 月 9 日
Amazon Cognito が欧州 (スペイン) およびアジアパシフィック (ハイデラバード) で利用可能になりました AWS リージョン。	欧州 (スペイン) およびアジアパシフィック (ハイデラバード) リージョンで Amazon Cognito リソースを作成できるようになりました。	2024 年 4 月 15 日

Amazon Cognito がアジアパシフィック (メルボルン) で利用可能になりました AWS リージョン。	アジアパシフィック (メルボルン) リージョンで Amazon Cognito リソースを作成できるようになりました。	2024 年 4 月 4 日
Flutter for Amazon Cognito ユーザープールに Android アプリの例を追加しました。	Amazon Cognito 用のスターターモバイルアプリは、 の Flutter アプリケーションの例から構築できます GitHub。	2024 年 4 月 4 日
新しい入門用コンテンツ	開始方法、一般的なシナリオ、マルチテナントのベストプラクティス、サインイン後のリソースへのアクセスに関するコンテンツを拡張しました。	2024 年 4 月 1 日
Amazon Cognito が欧州 (チューリッヒ) で利用可能になりました AWS リージョン。	欧州 (チューリッヒ) リージョンで Amazon Cognito リソースを作成できるようになりました。	2024 年 3 月 14 日
Amazon Cognito が中東 (UAE) で利用可能になりました AWS リージョン。	中東 (UAE) リージョンで Amazon Cognito リソースを作成できるようになりました。	2024 年 3 月 8 日
新しい SAML 機能と改善されたコンテンツ。	SAML リクエストの署名、SAML レスポンスの暗号化、IdP 開始 SAML SSO の設定が可能になりました。	2024 年 2 月 1 日
クォータの引き上げが利用可能になりました。	Amazon Cognito リクエストレートクォータの追加容量を購入できるようになりました。	2024 年 1 月 25 日

Amazon Cognito ID プールは、Service Quotas のリクエストレートをサポートします。	Amazon Cognito ID プールの requests-per-second (RPS) クォータをモニタリングし、Service Quotas コンソールで引き上げをリクエストできるようになりました。	2023 年 12 月 19 日
アクセストークンの内容をカスタマイズするための新機能を追加しました。	ユーザープールアクセストークンのクレームとスコープを追加、変更、削除できるようになりました。	2023 年 12 月 12 日
アプリケーションクライアントと OAuth スコープに関するコンテンツが改善されました。	ユーザープールアプリクライアント と スコープ、M2M 、およびリソースサーバーによる API 認証 をわかりやすくするための編集と訂正。従来のコンソールの説明を削除しました。	2023 年 11 月 14 日
デバイスとデバイス認証に関するコンテンツが改善されました。	デバイスキーとデバイス SRP 認証の使用に関する新しいコンテンツ。	2023 年 10 月 18 日
AWS Management Console ガイダンスを更新しました。	ユーザープールコンソールリファレンスを削除し、関連するテーマ内のトピックを並べ替え、Amazon Cognito コンソールのタブベースの整理に関するガイダンスを追加しました。	2023 年 8 月 30 日
LOGIN エンドポイントへの直接アクセスが強調されなくなりました。	ユーザープール ログインエンドポイント の視覚的概要を追加しまし、認可エンドポイント を使って認証を開始することを強調しました。	2023 年 8 月 30 日

Amazon Cognito が、アジアパシフィック (大阪) およびイスラエル (テルアビブ) で利用可能になりました AWS リージョン。	アジアパシフィック (大阪) およびイスラエル (テルアビブ) リージョンで Amazon Cognito リソースを作成できるようになりました。	2023 年 8 月 30 日
Amazon Verified Permissions を使用した Amazon Cognito の認可に関する情報を追加しました。	アプリ内で検証済みアクセス許可 API を呼び出し、中央機関によるアクセス決定を行うことができます。	2023 年 8 月 1 日
ユーザープールの詳細なユーザーアクティビティを Amazon CloudWatch Logs に記録する新機能を追加しました。	E メールおよび SMS メッセージ配信エラーを CloudWatch ロググループにログ記録できるようになりました。	2023 年 8 月 1 日
ID プールゲストユーザーの AWS マネージドポリシーに関する情報を更新しました。	ID プールゲストユーザーのアクセス許可のスコープダウンに、インラインセッションポリシーと AWS マネージドセッションポリシーの両方が含まれるようになりました。	2023 年 5 月 16 日
Amazon Cognito ID プールのコンテンツ改善と新しいコンソール手順。	新しいコンソールのチュートリアルを追加し、新しいコンソールエクスペリエンスを反映しました。また、アイデンティティプールに関するコード統合の詳細を改善しました。	2023 年 5 月 16 日
サービスホームページとユーザープールホームページの追加と改善。	Amazon Cognito とユーザープールの概要ページを更新しました。	2023 年 5 月 16 日

ユーザープールトークンドキュメントの全般的な改善。	トークンの例を更新し、トークンの検証に関する新しい情報を追加しました。	2023 年 2 月 16 日
Amazon Cognito ID プールのデータイベントを にログ記録できるようになりました AWS CloudTrail。	CloudTrail は、データイベントを記録する証跡での Amazon Cognito ID プールのハイボリュームな API オペレーションの選択をサポートします。	2023 年 2 月 15 日
Lambda トリガーの例と説明を更新しました。	Lambda トリガーの例が JavaScript バージョン 3 に更新されました。Lambda トリガーを API アクションに直接相関させることができるようになりました。	2023 年 1 月 31 日
Amazon Cognito ID プールは、認証されていないセッションに AWS 管理ポリシーを適用します。	拡張フローを使用して認証するアイデンティティプールユーザーには、セッションに追加の AWS 管理ポリシーが適用されるようになりました。	2023 年 1 月 31 日
コード例を追加しました。	このガイドには、さまざまなプログラミング言語の Amazon Cognito アプリのサンプルコードが含まれるようになりました。	2023 年 1 月 23 日
Amazon Cognito ユーザープールでの API モデルと認証に関する情報を追加しました。	Amazon Cognito ユーザープールには、リクエスト承認用の複数の API インターフェイスと形式があります。	2022 年 12 月 15 日

Amazon Cognito が欧州 (ミラノ) で利用可能になりました AWS リージョン。	欧州 (ミラノ) リージョンで Amazon Cognito ユーザープールを作成できるようになりました。	2022 年 12 月 6 日
ユーザープールの削除保護に関する情報を追加しました。	で新しいユーザープールを作成すると AWS Management Console、デフォルトで削除から保護されるようになりました。	2022 年 10 月 20 日
ホストされた UI のユーザーガイドと、ホストされた UI の TOTP MFA に関する情報を追加しました。	ユーザーは、Amazon Cognito がホストする UI で TOTP MFA デバイスを登録できるようになりました。デフォルトのホストされた UI をプレビューできるようになりました。	2022 年 9 月 8 日
AWS WAF および Amazon Cognito に関する情報を追加しました。	AWS WAF ウェブ ACL を Amazon Cognito ユーザープールに関連付けることができるようになりました。	2022 年 8 月 3 日
AWS CloudTrail イベントの例をさらに追加しました。	Amazon Cognito は、フェデレーションとホストされた UI リクエストを証跡に記録するようになりました	2022 年 6 月 15 日
2 ステップ属性検証に関する情報を追加しました。	ユーザーがログインする前に新しいメールアドレスまたは電話番号を確認する必要があるかどうかを選択できるようになりました。	2022 年 6 月 9 日

フェデレーションドキュメントを更新しました。新しい IP アドレス伝達機能。	ユーザープールソーシャル を設定するためのチュートリアルを更新しました IdPs。フェデレーションユーザープロファイルと属性マッピングに関する情報を追加しました。高度なセキュリティのために、デバイスのフィンガープリントに関する新しい情報を追加しました。	2022 年 5 月 31 日
ホストされた UI を操作しずにフェデレーティッドユーザーをサインインする	Amazon Cognito がユーザーをフェデレーションサインインにサイレントで誘導するように、アプリケーションをブックマークする方法に関する新しいページを追加しました。	2022 年 5 月 29 日
Amazon Cognito ユーザープールのリージョン内 SMS および E メールメッセージング	SMS メッセージに Amazon Simple Notification Service を使用し、ユーザープール AWS リージョン と同じ の E メールメッセージに Amazon Simple Email Service を使用できるようになりました。	2022 年 3 月 14 日
クォータの更新ページ	リソースとリクエストレートのクォータを追加し、明確化しました。	2022 年 1 月 10 日
新しい Amazon Cognito ユーザープールコンソールエクスペリエンス	更新された Amazon Cognito コンソールでユーザープールを作成および管理するための手順を更新しました。	2021 年 11 月 18 日

RevokeToken API と失効エンドポイント	RevokeToken オペレーションを使用して、ユーザーの更新トークンを取り消すことができます。	2021 年 6 月 10 日
マルチテナントのベストプラクティス	マルチテナントアプリケーションのベストプラクティスを追加しました。	2021 年 3 月 4 日
アクセスコントロールの属性	Amazon Cognito アイデンティティプールは、お客様が AWS リソースへのアクセス権をユーザーに付与する方法として、アクセスコントロール (AFAC) の属性を提供します。認証は、ID プロバイダーからのユーザーの属性 (ID プロバイダーは Amazon Cognito とのフェデレーションにこれらを使用します) に基づいて実行することができます。	2021 年 1 月 15 日
カスタム SMS 送信者 Lambda トリガーとカスタム E メール送信者 Lambda トリガー	カスタム SMS 送信者の Lambda トリガーとカスタム E メール送信者の Lambda トリガーは、サードパーティープロバイダーが Lambda 関数コード内からユーザーに E メールと SMS 通知を送信できるようにします。	2020 年 11 月 30 日
Amazon Cognito トークンの更新	アクセストークン、ID トークン、および更新トークンに更新された有効期限情報が追加されました。	2020 年 10 月 29 日

Amazon Cognito Service Quotas	Service Quotas は、Amazon Cognito カテゴリクォータで使用できます。Service Quotas コンソールを使用して、クォータの使用状況の表示、クォータの引き上げのリクエスト、クォータの使用状況をモニタリングする CloudWatch アラームの作成を行うことができます。この変更の一環として、Amazon Cognito ユーザープールの利用可能な CloudWatchメトリクスセクションが更新され、新しい情報が反映されました。新しいセクション名は、および Service Quotas での CloudWatchクォータと使用状況の追跡です。	2020 年 10 月 29 日
Amazon Cognito クォータの分類	クォータカテゴリは、クォータの使用状況の監視と引き上げのリクエストに役立てることができます。クォータは、一般的なユースケースに基づいてカテゴリに分類されます。	2020 年 8 月 17 日
US AWS GovCLoud でサポートされている Amazon Cognito	Amazon Cognito が AWS GovCloud（米国) リージョンでサポートされるようになりました。	2020 年 5 月 13 日

Amazon Cognito Pinpoint ドキュメントの更新	新しいサービスリンクロールが追加されました。「Amazon Cognito ユーザープールでの Amazon Pinpoint 分析の使用」にある手順が更新されました。	2020 年 5 月 13 日
新しい Amazon Cognito 専用セキュリティの章	セキュリティの章は、組織が AWS サービスの組み込みセキュリティと設定可能なセキュリティの両方に関する詳細情報を取得するのに役立ちます。新しい章では、クラウドとクラウド内のセキュリティに関する情報を提供します。	2020 年 4 月 30 日
Amazon Cognito ID プールが Apple でサインインをサポート	「Apple でサインイン」は、cn-north-1 リージョンを除く、Amazon Cognito が運営されているその他すべてのリージョンで利用できます。	2020 年 4 月 7 日
新しい Facebook API バージョニング	Facebook API にバージョン選択を追加しました。	2020 年 4 月 3 日
ユーザー名の大文字と小文字を区別しない更新	ユーザープールを作成する前に、ユーザー名の大文字と小文字を区別しない設定を有効にすることを推奨事項として追加しました。	2020 年 2 月 11 日

に関する新しい情報 AWS Amplify	AWS Amplify SDKsとライブラリを使用して Amazon Cognito をウェブまたはモバイルアプリと統合する方法についての情報を追加しました。 AWS Amplifyより前の Amazon Cognito SDK の使用に関する情報は削除されました。	2019 年 11 月 22 日
ユーザープールトリガーの新しい属性	Amazon Cognito は、ほとんどのユーザープールトリガーの AWS Lambda 関数に渡されるイベント情報に `clientMetadata` パラメータを含めるようになりました。このパラメータを使用して、追加のデータでカスタム認証ワークフローを強化できます。	2019 年 10 月 4 日
更新された制限	ListUsers API アクションのスロットリング制限が更新されました。	2019 年 6 月 25 日
新しい制限	ユーザープールのソフト制限に、ユーザー数の制限が含まれるようになりました。	2019 年 6 月 17 日

Amazon Cognito ユーザープールの Amazon Amazon SES E メール設定	Amazon Cognito が Amazon SES 設定を使用してユーザーに E メールを送信するように、ユーザープールを設定できます。この設定では、Amazon Cognito が、これ以外の方法では成し得なかった大規模な配信ボリュームで E メールを送信できるようになります。	2019 年 4 月 8 日
タグ付けのサポート	Amazon Cognito リソースのタグ付けに関する情報を追加しました。	2019 年 3 月 26 日
カスタムドメインの証明書を変更する	Amazon Cognito でホストされる UI のホストにカスタムドメインを使用する場合は、必要に応じてこのドメインの SSL 証明書を変更できます。	2018 年 12 月 19 日
新しい制限	各ユーザーが所属できるグループの最大数に新しい制限が追加されました。	2018 年 12 月 14 日
更新された制限	ユーザープールのソフト制限が更新されました。	2018 年 12 月 11 日
E メールアドレスと電話番号を検証するためのドキュメントの更新	ユーザーがアプリにサインアップするときに E メールまたは電話による確認を要求するようにユーザープールを構成する方法に関する情報を追加しました。	2018 年 11 月 20 日

E メールのテストに関するドキュメントの更新	アプリのテスト中に Amazon Cognito からの E メールを開始するためのガイダンスを追加しました。	2018 年 11 月 13 日
Amazon Cognito アドバンストセキュリティ	新しいセキュリティ機能を追加しました。これにより、デベロッパーは、アプリやユーザーを悪意のあるボットから保護し、ユーザーアカウントをセキュリティで保護して認証情報の侵害を防ぎ、サインイン試行の計算されたリスクに基づいてサインインに必要なチャレンジを自動的に調整できます。	2018 年 14 月 6 日
Amazon Cognito でホストされた UI のカスタムドメイン	デベロッパーが Amazon Cognito ユーザープール内のホストされた UI に独自の完全なカスタムドメインを使用することを可能にします。	2018 年 6 月 4 日
Amazon Cognito ユーザープール OIDC ID プロバイダー	Salesforce や Ping Identity などの OpenID Connect (OIDC) ID プロバイダー経由でのユーザープールサインインを追加しました。	2018 年 5 月 17 日
Amazon Cognito Lambda 移行トリガー	Lambda の移行トリガー機能を対象とするページが追加されました。	2018 年 4 月 8 日

Amazon Cognito デベロッパーガイドの更新	トップレベルの「Amazon Cognito とは」と「Amazon Cognito の使用開始方法」を追加しました。また、一般的なシナリオを追加し、ユーザープール TOC を再編成しました。新しい「Amazon Cognito ユーザープールの使用開始方法」セクションを追加しました。	2018 年 4 月 6 日
Amazon Cognito アドバンストセキュリティベータ	デベロッパーがアプリやユーザーを悪意のあるボットから保護し、インターネットの他の場所で侵害された認証情報に対してユーザーアカウントをセキュリティで保護して、サインインの試行の兆候について計算されたリスクに基づいて、サインインに必要なチャレンジを自動的に調整できるようにする、新しいセキュリティ機能が追加されました。	2017 年 11 月 28 日
Amazon Pinpoint の統合	Amazon Cognito ユーザープールアプリの分析を提供し、Amazon Pinpoint キャンペーン向けのユーザーデータを強化するために Amazon Pinpoint を使用する機能を追加しました。	2017 年 9 月 26 日

Amazon Cognito ユーザープールのフェデレーションおよび組み込みアプリケーション UI 機能	ユーザーが Facebook、Google、Login with Amazon、または SAML ID プロバイダーを使用してユーザープールにサインインすることを許可する機能を追加しました。カスタマイズ可能な組み込みアプリ UI およびカスタムクレームを使用した OAuth 2.0 のサポートを追加しました。	2017 年 8 月 10 日
HIPAA および PCI コンプライアンス関連の機能の変更	ユーザーが電話番号または E メールアドレスユーザー名として使用することを許可する機能が追加されました。	2017 年 7 月 6 日
ユーザーグループとロールベースのアクセスコントロール機能	ユーザーグループを作成および管理する管理機能が追加されました。管理者は、グループメンバーシップと管理者が作成したルールに基づいて、IAM ロールをユーザーに割り当てることができます。	2016 年 12 月 15 日
ドキュメントの更新	ユーザープールで AWS Lambda トリガーを使用する方法を示す例を更新しました。	2016 年 11 月 27 日
ドキュメントの更新	iOS コード例を更新しました。	2016 年 11 月 18 日
ドキュメントの更新	ユーザーアカウントの認証の流れに関する情報を追加しました。	2016 年 11 月 9 日

ユーザーアカウントの作成機能	Amazon Cognito コンソールと API を使用してユーザーアカウントを作成するための管理機能を追加しました。	2016 年 10 月 6 日
ユーザーインポート機能	Cognito ユーザープール一括インポート機能を追加しました。この機能を使用して、既存の ID プロバイダーから Amazon Cognito ユーザープールにユーザーを移行します。	2016 年 9 月 1 日
Cognito ユーザープールの一般提供	Cognito ユーザープール機能を追加しました。この機能を使用して、ユーザーディレクトリを作成、管理し、ユーザープールを使用してモバイルアプリケーションまたはウェブアプリケーションにサインアップとサインインを追加します。	2016 年 7 月 28 日
SAML サポート	Security Assertion Markup Language 2.0 (SAML 2.0) を通じたアイデンティティプロバイダーによる認証サポートを追加しました。	2016 年 6 月 23 日
CloudTrail 統合	との統合を追加しました AWS CloudTrail。	2016 年 2 月 18 日
イベントと Lambda の統合	Amazon Cognito の重要なイベントに応じて AWS Lambda 関数を実行できます。	2015 年 4 月 9 日
Amazon Kinesis へのデータストリーム	データストリームにコントロールや洞察を提供します。	2015 年 3 月 4 日

OpenID Connect のサポート	OpenID 接続プロバイダーをサポートします。	2014 年 11 月 23 日
プッシュ同期	サイレントプッシュ同期のサポートを有効にします。	2014 年 11 月 6 日
デベロッパーが認証した ID のサポートを追加	独自の認証およびアイデンティティ管理システムを持つデベロッパーが、Amazon Cognito の ID プロバイダーとして扱われるようになります。	2014 年 9 月 29 日
Amazon Cognito の一般提供		2014 年 7 月 10 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。