リファレンスガイド

AWS SDKsとツール

AWS SDKsとツール: リファレンスガイド

Table of Contents

AWS SDKsリファレンスガイド

多くの SDK とツールは、共有設計仕様または共有ライブラリを通じて、いくつかの共通機能を共有しています。

このガイドには以下に関する情報が含まれています。

- 構成 – 共有 `config`および `credentials` ファイルまたは環境変数を使用して AWS SDKsとツールを設定する方法。
- 認証とアクセス – を使用して開発 AWS するときに、コードまたはツールが で認証する方法を確立します AWS のサービス。
- 設定リファレンス – 認証と設定に使用できるすべての標準設定のリファレンス。
- AWS Common Runtime (CRT) ライブラリ – ほぼすべての SDKs で使用できる共有 AWS 共通ランタイム (CRT) ライブラリの概要。
- AWS SDKsメンテナンスポリシー は、モバイルおよびモノのインターネット (IoT) SDK を含む AWS Software Development Kit (SDK) とツールのメンテナンスポリシーとバージョニング SDKs 、およびそれらの基盤となる依存関係について説明します。 SDKs IoT

この AWS SDKsおよびツールリファレンスガイドは、複数の SDKs およびツールに適用される情報の基礎となることを目的としています。ここに記載されている情報に加えて、使用している SDK またはツール固有のガイドも使用してください。このガイドでの資料の関連セクションを含む SDK とツールは次のとおりです。

次を使用している場合:	このガイドの関連セクションは、以下のとおりです。
• 任意の SDK またはツール	AWS SDKsメンテナンスポリシー
• AWS Cloud Development Kit (AWS CDK) デベロッパーガイド • AWS Serverless Application Model デベロッパーガイド • AWS Toolkit for Eclipse ユーザーガイド • AWS Toolkit for JetBrains ユーザーガイド	構成 認証とアクセス AWS SDKsメンテナンスポリシー

次を使用している場合:	このガイドの関連セクションは、以下のとおりです。
• AWS Toolkit for Visual Studio ユーザーガイド • AWS Toolkit for Visual Studio Code ユーザーガイド	
• AWS Command Line Interface ユーザーガイド • AWS SDK for C++ デベロッパーガイド • AWS SDK for Go デベロッパーガイド • AWS SDK for Java デベロッパーガイド • AWS SDK for JavaScript デベロッパーガイド • AWS SDK for Kotlin • AWS SDK for .NET デベロッパーガイド • AWS SDK for PHP デベロッパーガイド • AWS SDK for Python (Boto3) の開始方法 • AWS SDK for Ruby デベロッパーガイド • AWS SDK for Rust • AWS SDK for Swift • AWS Tools for Windows PowerShell ユーザーガイド	構成 認証とアクセス 設定リファレンス AWS Common Runtime (CRT) ライブラリ AWS SDKsメンテナンスポリシー AWS SDKsとツールのバージョンサポートマトリックス

デベロッパーリソース

でのアプリケーションの開発に役立つツールの概要については AWS、「 で構築するツール AWS」を参照してください。サポートに関する情報は、「AWS ナレッジセンター」 を参照してください。

Amazon Q Developer は、生成 AI を活用した会話型アシスタントで、 AWS アプリケーションの理解、構築、拡張、運用に役立ちます。でのビルドを加速するために AWS、Amazon Q を強化するモデルには、より完全で実用的な、参照される回答を生成する高品質の AWS コンテンツが拡張されて

います。詳細については、「Amazon Q デベロッパーユーザーガイド」の「Amazon Q デベロッパーとは」を参照してください。

ツールキットテレメトリ通知

AWS 統合開発環境 (IDE) ツールキットは、IDE から AWS のサービスへのアクセスを可能にするプラグインと拡張機能です。各 IDE Toolkit の詳細については、前の表の Toolkit ユーザーガイドを参照してください。

AWS IDE Toolkits は、今後の AWS Toolkit リリースに関する決定を通知するために、クライアント側のテレメトリデータを収集して保存する場合があります。収集されたデータは、 AWS Toolkit の使用量を定量化します。

すべての AWS IDE Toolkit で収集されたテレメトリデータの詳細については、`aws-toolkit-common`Github リポジトリの commonDefinitions.json ドキュメントを参照してください。

各 AWS IDE Toolkit によって収集されたテレメトリデータの詳細については、次の AWS Toolkits の Github リポジトリにあるリソースドキュメントを参照してください。

- AWS Toolkit for Visual Studio
- AWS Toolkit for Visual Studio Code
- AWS Toolkit for JetBrains

AWS Toolkit でアクセスできる特定の AWS サービスでは、追加のクライアント側のテレメトリデータが収集される場合があります。個々の AWS サービスによって収集されるデータの種類の詳細については、関心のある特定のサービスのAWS ドキュメントトピックを参照してください。

構成

AWS SDK や AWS Command Line Interface (AWS CLI) AWS などの他の開発者ツールを使用すると、 AWS サービス API を操作できます。ただし、その前に、要求された操作を実行するために必要な情報を SDK またはツールに設定する必要があります。

この情報には以下のアイテムが含まれます。

- API の呼び出し元を識別する認証情報。認証情報は、サーバーへのリクエストを暗号化するために使用されます。 AWS AWS この情報を使用して本人確認を行い、それに関連するアクセス権限ポリシーを取得できます。次に、どのようなアクションを実行できるかを判断できます。
- リクエストの処理方法、リクエストの送信先 (AWS サービスエンドポイント)、 AWS CLI およびレスポンスの解釈または表示方法をまたは SDK に伝えるために使用するその他の設定情報。

各 SDK またはツールは、必要な認証情報と設定情報を供給するために使用できる複数のソースをサポートしています。ソースの中には SDK やツールに独自のものもあるため、その方法の使用方法の詳細については、そのツールまたは SDK のドキュメントを参照する必要があります。

ただし、ほとんどの AWS SDK とツールは、(コード自体以外の) 次の 2 つの主要なソースからの共通設定をサポートしています。

- AWS 共有設定ファイルと認証情報ファイル — AWS SDK またはツールに認証と設定を指定する最も一般的な方法は、`configcredentials`共有ファイルとファイルです。これらのファイルを使用して、ツールやアプリケーションが使用できる設定を保存します。共有 `config` ファイルと `credentials` ファイル内の設定は特定のプロファイルに関連付けられます。複数のプロファイルを使用して、さまざまな設定構成を作成してさまざまなシナリオに適用できます。 AWS ツールを使用してコマンドを呼び出したり、SDK を使用して AWS API を呼び出したりする場合、そのアクションに使用するプロファイル、つまりどの構成設定を使用するかを指定できます。プロファイルの 1 つが `default` プロファイルとして指定され、使用するプロファイルを明示的に指定しない場合に自動的に使用されます。これらのファイルに保存できる設定は、このリファレンスガイドに記載されています。
- 環境変数 – 一部の設定は、オペレーティングシステムの環境変数に保存することもできます。環境変数は一度に 1 セットしか有効にできませんが、プログラムの実行や要件の変化に応じて動的に変更するのは簡単です。

このセクションのその他のトピック

- 共有 config ファイルおよび credentials ファイル
- 共有 config ファイルと 共有 credentials ファイルの場所
- 環境変数のサポート
- JVM システムプロパティーのサポート

共有 `config` ファイルおよび `credentials` ファイル

共有 AWS `config` ファイルと `credentials` ファイルには、一連のプロファイルが含まれています。プロファイルは、（AWS Command Line Interface AWS CLI）、 AWS SDKs、およびその他のツールで使用されるキーと値のペアの一連の設定です。プロファイルを使用するときに SDK / ツールの一部を設定するために、設定値がプロファイルに添付されます。これらのファイルは、値がユーザーのローカル環境にあるすべてのアプリケーション、プロセス、または SDK に影響するという点で「共有」されます。

共有 `config` ファイルと `credentials` ファイルはどちらも ASCII 文字（UTF-8 でエンコードされた）のみを含むプレーンテキストファイルです。これらは一般に INI ファイルと呼ばれる形式をとります。

プロファイル

共有 `config` ファイルと `credentials` ファイル内の設定は特定のプロファイルに関連付けられます。ファイル内で複数のプロファイルを定義して、異なる開発環境に適用する異なる設定設定を作成できます。

`[default]` プロファイルには、特定の名前付きプロファイルが指定されていない場合に SDK またはツールオペレーションで使用される値が含まれます。名前で明示的に参照できる個別のプロファイルを作成することもできます。各プロファイルは、アプリケーションやシナリオに応じて異なる設定や値を使用できます。

> **Note**
>
> `[default]` は単に名前のないプロファイルです。このプロファイルは、ユーザーがプロファイルを指定しない場合に SDK が使用するデフォルトのプロファイルであるため、`default` の名前が付けられています。継承されたデフォルト値を他のプロファイルに使用することはありません。`[default]` プロファイルで何かを設定し、名前付きプロファイルで設定しない場合、名前付きプロファイルを使用する場合、値は設定されません。

名前付きプロファイルを設定する

`[default]` プロファイルと複数の名前付きプロファイルは、同じファイル内に存在できます。次の設定を使用して、コードの実行時に SDK またはツールが使用するプロファイルの設定を選択します。プロファイルはコード内で選択することも、 を操作するときにコマンドごとに選択することもできます AWS CLI。

次のいずれかを設定して、この機能を設定します。

`AWS_PROFILE` - 環境変数

この環境変数を名前付きプロファイルまたは「デフォルト」に設定すると、すべての SDK コードと AWS CLI コマンドはそのプロファイルの設定を使用します。

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_PROFILE="my_default_profile_name";
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_PROFILE "my_default_profile_name"
```

`aws.profile` - JVM システムプロパティ

JVM の SDK for Kotlin と SDK for Java 2.x では、`aws.profile`システムプロパティ を設定できます。SDK は、サービスクライアントを作成するときに、コードで設定が上書きされない限り、名前付きプロファイルの設定を使用します。SDK for Java 1.x は、このシステムプロパティをサポートしていません。

設定ファイルの形式

`config` ファイルは、セクションにまとめられています。セクションは、設定の名前付きコレクションであり、別のセクション定義の行が検出されるまで続きます。

`config` ファイルは、次の形式を使用するプレーンテキストファイルです。

- セクション内のすべてのエントリは、`setting-name=value` の一般的な形式になります。
- 行の先頭にハッシュタグ (#) を付けると、行をコメントアウトできます。

セクションタイプ

セクション定義は、設定のコレクションに名前を付ける行です。セクション定義行の先頭と末尾は角括弧 ([ ]) です。括弧内には、セクションタイプ識別子とセクションのカスタム名があります。英文字、数字、ハイフン (-)、アンダースコア (_) は使用できますが、スペースは使用できません。

セクションタイプ: **default**

セクション定義行の例：`[default]`

`[default]` は、`profile`セクション識別子を必要としない唯一のプロファイルです。

`[default]` プロファイルのある基本 `config` ファイルの例を以下に示します。`region` を設定します。別のセクション定義が見つかるまで、この行に続くすべての設定は、このプロファイルの一部です。

```
[default]
#Full line comment, this text is ignored.
region = us-east-2
```

セクションタイプ: **profile**

セクション定義行の例：`[profile dev]`

`profile` セクション定義行は、さまざまな開発シナリオに適用できる名前付き設定グループです。名前付きプロファイルについての理解を深めるには、前のセクションの「プロファイル」を参照してください。

次の例は、`profile`セクション定義行と という名前のプロファイルを持つ`config`ファイルを示していますfoo。別のセクション定義が見つかるまで、この行に続くすべての設定は、この名前付きプロファイルの一部です。

```
[profile foo]
...settings...
```

次の例の `s3` 設定やサブ設定など、一部の設定には独自のサブ設定グループがネストされています。サブ設定を 1 つまたは複数のスペースでインデントしてグループに関連付けます。

```
[profile test]
region = us-west-2
```

```
s3 =
    max_concurrent_requests=10
    max_queue_size=1000
```

セクションタイプ: **sso-session**

セクション定義行の例：[sso-session *my-sso*]

sso-session セクション定義行には、 を使用して AWS 認証情報を解決するようにプロファイルを設定する際に使用する設定のグループに名前が付けられます AWS IAM Identity Center。シングルサインオン認証の設定の詳細については、「IAM Identity Center 認証」を参照してください。プロファイルは、キーと値のペアによって sso-session セクションにリンクされます。ここで、sso-session はキー、sso-session セクションの名前は値です（sso-session = <name-of-sso-session-section> など）。

次の例では、「my- AWS sso」のトークンを使用して、111122223333」アカウントの SampleRole「」IAM ロールの短期認証情報を取得するプロファイルを設定します。「my-sso」sso-session セクションは、profile セクションの中で sso-session キーを使用して名前で参照されます。

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
```

セクションタイプ: **services**

セクション定義行の例：[services *dev*]

> **Note**
>
> services セクションはサービス固有のエンドポイントのカスタマイズをサポートしており、この機能を含む SDK とツールでのみ使用できます。お使いの SDK でこの機能が使用できるかどうかを確認するには、「サービス固有のエンドポイント」の AWS SDKsとの互換性を参照してください。

services セクション定義行には、 AWS のサービス リクエストのカスタムエンドポイントを設定する設定のグループに名前が付けられます。プロファイルは、キーと値のペアによって services セクションにリンクされます。ここで、services はキー、services セクションの名前は値です (services = <name-of-services-section> など) 。

services セクションはさらに<SERVICE> = 行ごとにサブセクションに分割されます。ここで、<SERVICE>は AWS のサービス 識別子キーです。 AWS のサービス 識別子は、すべてのスペースをアンダースコアに置き換え、すべての文字を小文字に置き換えserviceIdることで、API モデルのに基づいています。services セクションで使用するすべてのサービス識別子キーのリストについては、「サービス固有のエンドポイントの識別子」を参照してください。サービス識別子キーの後には、ネストされた設定 (それぞれが 1 行にあり、2 つのスペースでインデントされている) が続きます。

次の例では、services 定義を使用して、 Amazon DynamoDB サービスに対して行われたリクエストにのみ使用するようにエンドポイントを設定しています。"local-dynamodb" services セクションは、profile セクションの中で services キーを使用して名前で参照されます。 AWS のサービス 識別子キーは ですdynamodb。 Amazon DynamoDB サービスサブセクションは 行目から始まりますdynamodb = 。直後のインデントされた行はすべてそのサブセクションに含まれ、そのサービスに適用されます。

```
[profile dev]
services = local-dynamodb

[services local-dynamodb]
dynamodb =
  endpoint_url = http://localhost:8000
```

カスタムエンドポイントの設定の詳細については、「サービス固有のエンドポイント」を参照してください。

認証情報ファイルの形式

プロファイルセクションが単語 profile で始まらないことを除けば、credentials ファイルのルールは一般的に config ファイルのルールと同じです。角括弧の間には、プロファイル名自体のみを使用してください。次の例は、 という名前のプロファイルセクションを持つ credentials ファイルを示していますfoo。

```
[foo]
...credential settings...
```

ファイルに保存できるのは、「シークレット」または機密と見なされる設定credentialsのみです。aws_access_key_id、aws_secret_access_key、および aws_session_token。これらの設定は共有configファイルに配置することもできますが、これらの機密値は別のcredentialsファイルに保持することをお勧めします。これにより、必要に応じてファイルごとに個別のアクセス許可を与えることができます。

[default] プロファイルのある基本 credentials ファイルの例を以下に示します。これにより aws_access_key_id、、aws_secret_access_key、および aws_session_token グローバル設定が設定されます。

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRI
```

credentials ファイルdefaultで名前付きプロファイルを使用するか、「」を使用するかにかかわらず、ここでの設定は、同じプロファイル名を使用するconfigファイルの設定と組み合わせられます。同じ名前を共有するプロファイルの両方のファイルに認証情報がある場合、認証情報ファイルのキーが優先されます。

共有 config ファイルと 共有 credentials ファイルの場所

共有 AWS config ファイルと credentialsファイルは、 AWS SDKsとツールの設定情報を保持するプレーンテキストファイルです。ファイルは環境内にローカルに存在し、SDK コードまたは環境で実行する AWS CLI コマンドによって自動的に使用されます。例えば、独自のコンピュータで、または Amazon Elastic Compute Cloud インスタンスで開発する場合などです。

SDK またはツールを実行すると、これらのファイルをチェックし、使用可能な構成設定をロードします。ファイルがまだ存在しない場合は、SDK またはツールによって基本的なファイルが自動的に作成されます。

デフォルトでは、ファイルは homeまたは ユーザーフォルダに配置された .aws という名前のフォルダにあります。

オペレーティングシステム	ファイルのデフォルトの場所と名前
Linux および macOS	~/.aws/config ~/.aws/credentials

オペレーティングシステム	ファイルのデフォルトの場所と名前

Windows	`%USERPROFILE%\.aws\config`
	
	`%USERPROFILE%\.aws\credentials`

ホームディレクトリの解像度

~ は、ホームディレクトリの解決にのみ使用されます。

- パスを開始します。
- 直後に /またはプラットフォーム固有の区切り文字が続きます。Windows では、 ~/ と ~\の両方がホームディレクトリに解決されます。

ホームディレクトリを決定するときに、次の変数がチェックされます。

- （全プラットフォーム）HOME 環境変数
- （Windows プラットフォーム）USERPROFILE 環境変数
- (Windows プラットフォーム) HOMEDRIVEとHOMEPATH環境変数の連結 ($HOMEDRIVE $HOMEPATH)
- （SDK またはツールごとのオプション）SDK またはツール固有のホームパス解決関数または変数

可能な場合は、パスの先頭にユーザーのホームディレクトリが指定されている場合（例：~username/）、要求されたユーザー名のホームディレクトリ（例：/home/username/.aws/config）に解決されます。

これらのファイルのデフォルトの場所を変更する

次のいずれかを使用して、SDK またはツールによってこれらのファイルがロードされる場所を上書きできます。

環境変数を使用します。

以下の環境変数を設定して、これらのファイルの場所または名前をデフォルト値からカスタム値に変更できます。

- config ファイルの環境変数：**AWS_CONFIG_FILE**

- `credentials` ファイルの環境変数：**`AWS_SHARED_CREDENTIALS_FILE`**

Linux/macOS

Linux または macOS で次の[エクスポート]コマンドを実行して、別の場所を指定できます。

```
$ export AWS_CONFIG_FILE=/some/file/path/on/the/system/config-file-name
$ export AWS_SHARED_CREDENTIALS_FILE=/some/other/file/path/on/the/system/
credentials-file-name
```

Windows

Windows で次の [setx] コマンドを実行して、別の場所を指定できます。

```
C:\> setx AWS_CONFIG_FILE c:\some\file\path\on\the\system\config-file-name
C:\> setx AWS_SHARED_CREDENTIALS_FILE c:\some\other\file\path\on\the\system
\credentials-file-name
```

環境変数を使用してシステムを設定する方法の詳細については、「」を参照してください環境変数のサポート。

JVM システムプロパティを使用する

JVM で実行されている SDK for Kotlin と SDK for Java 2.x では、次の JVM システムプロパティを設定して、これらのファイルの場所または名前をデフォルトからカスタム値に変更できます。

- `config` ファイル JVM システムプロパティ: **`aws.configFile`**
- `credentials` ファイルの環境変数：**`aws.sharedCredentialsFile`**

JVM システムプロパティを設定する方法については、「」を参照してくださいthe section called “JVM システムプロパティの設定方法”。SDK for Java 1.x は、これらのシステムプロパティをサポートしていません。

環境変数のサポート

環境変数を使用すると、別の方法で設定オプションと認証情報を指定できます。このため、スクリプト処理や、名前付きプロファイルを一時的にデフォルトとして設定する場合に便利です。ほとんど

の SDK でサポートされている環境変数のリストについては、「環境変数の一覧」を参照してください。

オプションの優先順位

- 環境変数を使用して設定を指定した場合、 AWS `config` と `credentials` 共有ファイル内のプロファイルからロードされた値は上書きされます。
- AWS CLI コマンドラインでパラメータを使用して設定を指定した場合、対応する環境変数、または設定ファイルのプロファイルからの値が上書きされます。

環境変数の設定方法

次の例では、デフォルトのユーザーの環境変数を設定する方法を示します。

Linux, macOS, or Unix

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export
 AWS_SESSION_TOKEN=AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
$ export AWS_REGION=us-west-2
```

環境変数を設定すると、シェルセッションの終了時まで、または変数に別の値を設定するまで、使用する値が変更されます。変数をシェルのスタートアップスクリプトで設定することで、変数をこれからのセッションで永続的にすることができます。

Windows Command Prompt

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> setx
 AWS_SESSION_TOKEN AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
C:\> setx AWS_REGION us-west-2
```

`set` を使用して環境変数を設定すると、現在のコマンドプロンプトセッションの終了時まで、または変数を別の値に設定するまで、使用する値が変更されます。`setx` を使用して環境変数を設定すると、現在のコマンドプロンプトセッションおよびコマンド実行後に作成するすべてのコマンドプロンプトセッションで使用する値が変更されます。これは、コマンド実行時にすでに実行されている他のコマンドシェルには影響を及ぼしません。

PowerShell

```
PS C:\> $Env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
PS C:\> $Env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
PS C:
\> $Env:AWS_SESSION_TOKEN="AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R40l
PS C:\> $Env:AWS_REGION="us-west-2"
```

前の例に示すように PowerShell プロンプトで環境変数を設定した場合は、現在のセッションの期間だけ値が保存されます。PowerShell およびコマンドプロンプトセッション間で環境変数を永続的に設定するには、[コントロールパネル] の [システム] アプリケーションを使用して変数を保存します。または、変数を PowerShell プロファイルに追加すると、その変数を今後のすべての PowerShell セッションに設定できます。環境変数の保存やそれをセッション間で永続的に維持する詳細については、「PowerShell documentation」(PowerShell ドキュメント) を参照してください。

サーバーレス環境変数設定

開発にサーバーレスアーキテクチャを使用する場合、環境変数を設定するための他のオプションもあります。コンテナによっては、非クラウド環境と同様に、そのコンテナで実行されるコードに対して異なる方法を使用して環境変数を表示したりアクセスしたりすることができます。

たとえば、 AWS Lambda では環境変数を直接設定できます。詳細については、「AWS Lambda デベロッパーガイド」 の「AWS Lambda 環境変数の使用」 を参照してください。

サーバーレスフレームワークでは、環境設定の下のプロバイダーキーの下の `serverless.yml` ファイルに SDK 環境変数を設定できることがよくあります。この `serverless.yml` ファイルについて詳しくは、「サーバーレスフレームワーク」ドキュメントの 「一般関数設定」 を参照してください。

コンテナ環境変数の設定にどのメカニズムを使用するかにかかわらず、コンテナによって予約されているものもあります。たとえば、Lambdaの 「定義済みランタイム環境変数」 で説明されているものなどです。環境変数の処理方法や制限の有無については、使用しているコンテナの公式ドキュメントを必ず確認してください。

JVM システムプロパティーのサポート

JVM システムプロパティには、JVM 上で実行される SDK (やなど) の設定オプションと認証情報を指定するもう 1 つの方法があります。 AWS SDK for Java AWS SDK for KotlinSDK でサポートされている JVM システムプロパティーの一覧については、「設定リファレンス」を参照してください。

オプションの優先順位

- JVM システムプロパティを使用して設定を指定すると、環境変数にある値、または共有 AWS `config` およびファイルのプロファイルからロードされた値が上書きされます。`credentials`
- 環境変数を使用して設定を指定すると、共有 AWS `config` `credentials` およびファイルのプロファイルからロードされた値よりも優先されます。

JVM システムプロパティの設定方法

JVM システムプロパティーはいくつかの方法で設定できます。

コマンドラインで

`java`スイッチを使用してコマンドを呼び出すときに、コマンドラインで JVM システムプロパティーを設定します。`-D`以下のコマンドは、コード内の値を明示的にオーバーライドしない限り、 AWS リージョン すべてのサービスクライアントに対してをグローバルに設定します。

```
java -Daws.region=us-east-1 -jar <your_application.jar> <other_arguments>
```

複数の JVM システムプロパティーを設定する必要がある場合は、`-D`スイッチを複数回指定してください。

環境変数を使用する。

コマンドラインにアクセスして JVM を呼び出してアプリケーションを実行できない場合は、`JAVA_TOOL_OPTIONS`環境変数を使用してコマンドラインオプションを設定できます。この方法は、Java AWS Lambda ランタイムで関数を実行したり、組み込み JVM でコードを実行したりする場合に役立ちます。

次の例では、コード内の値を明示的にオーバーライドしない限り、 AWS リージョン すべてのサービスクライアントに対してをグローバルに設定します。

Linux, macOS, or Unix

```
$ export JAVA_TOOL_OPTIONS="-Daws.region=us-east-1"
```

環境変数を設定すると使用する値が変更され、その値はシェルセッションが終了するか、または変数に別の値が設定されるまで有効です。変数をシェルのスタートアップスクリプトで設定することで、変数をこれからのセッションで永続的にすることができます。

Windows Command Prompt

```
C:\> setx JAVA_TOOL_OPTIONS -Daws.region=us-east-1
```

`set` を使用して環境変数を設定すると、現在のコマンドプロンプトセッションの終了時まで、または変数を別の値に設定するまで、使用する値が変更されます。`setx` を使用して環境変数を設定すると、現在のコマンドプロンプトセッションおよびコマンド実行後に作成するすべてのコマンドプロンプトセッションで使用する値が変更されます。これは、コマンド実行時にすでに実行されている他のコマンドシェルには影響を及ぼしません。

実行時

次の例に示すように、`System.setProperty`メソッドを使用して、実行時にコード内で JVM システムプロパティーを設定することもできます。

```
System.setProperty("aws.region", "us-east-1");
```

> **Important**
>
> SDK サービスクライアントを初期化する前に JVM システムプロパティーを設定してください。そうしないと、サービスクライアントが他の値を使用する可能性があります。

認証とアクセス

AWS のサービス を使用して開発する際には、AWS によりコードがどのように認証するかを確立する必要があります。環境と利用可能な AWS のアクセスに応じて、AWS リソースへのプログラムによるアクセスはさまざまな方法で設定できます。

ローカル (AWS 内部ではない) で実行されるコードの認証オプション

- IAM Identity Center 認証セキュリティのベストプラクティスとして、AWS Organizations と IAM Identity Center を使用して、すべての AWS アカウント にわたってアクセスを管理することをお勧めします。AWS IAM Identity Center でユーザーを作成するか、Microsoft Active Directory を使用するか、SAML 2.0 ID プロバイダー (IdP) を使用するか、または IdP を AWS アカウント に個別にフェデレーションすることができます。お使いのリージョンが IAM Identity Center をサポートしているかどうかを確認するには、Amazon Web Services 全般のリファレンス の「AWS IAM Identity Center エンドポイントとクォータ」を参照してください。
- IAM Roles Anywhere – IAM Roles Anywhere を使用すると、サーバー、コンテナ、アプリケーションなど、AWS の外部で実行されるワークロードに関する一時的なセキュリティ認証情報を、IAM で取得することがきます。IAM Roles Anywhere を使用するには、ワークロードで X.509 証明書を使用する必要があります。
- ロールの割り当て – IAM ロールを引き受けて、他の方法ではアクセスできない可能性のある AWS リソースに一時的にアクセスできます。
- AWS アクセスキー — あまり便利でなかったり、 AWS リソースのセキュリティリスクを増大させたりする可能性のあるその他のオプション。

AWS環境内で実行されるコードの認証オプション

- Amazon EC2 インスタンスの IAM ロールの使用 — IAM ロールを使用して、Amazon EC2 インスタンスでアプリケーションを安全に実行します。
- IAM Identity Center AWS を使用してプログラムから操作する方法は次のとおりです。
  - コンソールから AWS CLI コマンドを実行する場合に「AWS CloudShell」 を使用します。
  - AWS リソースを備えた統合開発環境 (IDE) を AWS で使用してプログラミングを開始する場合に「AWS Cloud9」 を使用します。
  - ソフトウェア開発チーム向けのクラウドベースのコラボレーションスペースを試すには、「Amazon CodeCatalyst」 のご使用を検討ください。

ウェブベースのアイデンティティープロバイダーによる認証 - モバイルまたはクライアントベースのウェブアプリケーション

AWS へのアクセスを必要とするモバイルアプリケーションまたはクライアントベースのウェブアプリケーションを作成する場合は、ウェブ ID フェデレーションを使用して AWS 一時的なセキュリティ認証情報を動的に要求するようにアプリを構築してください。

ウェブ ID フェデレーションを使用すると、カスタムサインインコードを作成したり独自のユーザー ID を管理したりする必要はありません。その代わりに、アプリのユーザーは、よく知られている外部 ID プロバイダー (IdP) (例: Login with Amazon、Facebook、Google などの OpenID Connect (OIDC) 互換の IdP) を使用してサインインすることができます。認証トークンを受け取ったら、そのトークンを AWS アカウント のリソースを使用するためのアクセス許可を持つ IAM ロールにマッピングし、AWS の一時的セキュリティ認証情報に変換することができます。

SDK またはツールへ設定する方法については、「ウェブアイデンティティまたは OpenIDコネクトとのフェデレーション」を参照してください。

モバイルアプリケーションに対しては、Amazon Cognito の使用をお勧めします。Amazon Cognito は ID ブローカーとして機能し、ユーザーの代わりに多くのフェデレーション作業を行います。詳細については、「IAM ユーザーガイド」の「モバイルアプリに対する Amazon Cognito の使用」を参照してください。

アクセス管理に関する詳細情報

「IAM ユーザーガイド」には、AWS リソースにアクセスするためのアクセス許可を安全に制御するための以下の詳細情報があります。

- IAM ID (ユーザー、ユーザーグループ、ロール) - AWS のアイデンティティ基本を理解します。
- IAM におけるセキュリティのベストプラクティス —「責任分担モデル」 に従って AWS アプリケーションを開発する際に従うべきセキュリティ上の推奨事項。

Amazon Web Services 全般のリファレンス には、以下に関する基本的な基本事項があります。

- AWS 認証情報の理解と取得 — コンソールアクセスとプログラムアクセスの両方に関するアクセスキーオプションと管理プラクティス。

AWS ビルダー ID

すでに所有している、または作成したい AWS アカウント を AWS ビルダー ID で補完します。AWS アカウント は作成する AWS リソースのコンテナとして機能し、それらのリソースにセキュリティ境界を設けるのに対し、AWS ビルダー ID ユーザー個人を表します。AWS ビルダー ID を使用してサインインし、Amazon CodeWhispererやAmazon CodeCatalystなどの開発者ツールやサービスにアクセスできます。

- AWS サインイン ユーザーガイドでの「AWS ビルダー ID によるサインイン」 — AWS ビルダー ID の作成方法と使用方法、Builder ID の機能について学んでください。
- CodeWhisperer と AWS Toolkit -Builder ID による認証」 — 「CodeWhisperer ユーザーガイド」でのBuilder ID – CodeWhispererが AWS ビルダー ID を使用する方法について説明します。
- CodeCatalyst の概念 - Amazon CodeCatalyst ユーザーガイドでのAWS ビルダー ID – CodeCatalyst での AWS ビルダー ID の使用方法について説明します。

IAM Identity Center 認証

AWS IAM Identity Center は、AWS コンピューティング以外のサービスで開発するときに AWS 認証情報を提供する推奨方法です。たとえば、これはローカルの開発環境のようなものです。Amazon Elastic Compute Cloud (Amazon EC2) や などの AWS リソースで開発する場合は AWS Cloud9、代わりにそのサービスから認証情報を取得することをお勧めします。

このチュートリアルでは、IAM Identity Center アクセスを確立し、 AWS アクセスポータルと を使用して SDK またはツール用に設定します AWS CLI。

- AWS アクセスポータルは、IAM Identity Center に手動でサインインするウェブの場所です。URL のフォーマットは `d-xxxxxxxxxx.awsapps.com/start`、または *`your_subdomain`*`.awsapps.com/start` です。 AWS アクセスポータルにサインインすると、そのユーザーに設定された ロール AWS アカウント と ロールを表示できます。この手順では、AWS アクセスポータルを使用して、SDK/ツール認証プロセスに必要な設定値を取得します。
- AWS CLI は、コードによって行われた API コールに IAM Identity Center 認証を使用するように SDK またはツールを設定するために使用されます。この 1 回限りのプロセスでは、共有 AWS `config`ファイルが更新され、コードの実行時に SDK またはツールによって使用されます。

IAM Identity Center を使用してプログラムによるアクセスを設定します

ステップ 1：アクセスを確立し、適切なアクセス許可セットを選択します

IAM Identity Center をまだ有効にしていない場合は、「 ユーザーガイド」の「IAM Identity Center のAWS IAM Identity Center 有効化」を参照してください。

AWS 認証情報にアクセスするには、次のいずれかの方法を選択します。

IAM Identity Center 経由のアクセスを確立していません

1. AWS IAM Identity Center ユーザーガイドのデフォルトの IAM Identity Center ディレクトリを使用してユーザーアクセスを設定する手順に従って、ユーザーを追加し、管理者権限を追加します。
2. アクセス`AdministratorAccess`許可セットは、通常の開発には使用しないでください。代わりに、雇用主がこの目的のためにカスタム`PowerUserAccess`アクセス許可セットを作成していない限り、事前定義されたアクセス許可セットを使用することをお勧めします。

   同じデフォルトの IAM Identity Center ディレクトリでユーザーアクセスを設定する手順を再度実行しますが、今回は次の操作を行います。

   - *`Admin team`* グループを作成する代わりに、*`Dev team`*グループを作成し、その後、これを手順に置き換えます。
   - 既存のユーザーを使用できますが、そのユーザーを新しい*`Dev team`*グループに追加する必要があります。
   - アクセス*`AdministratorAccess`*許可セットを作成する代わりに、アクセス*`PowerUserAccess`*許可セットを作成し、その後、指示でこれを置き換えます。

   完了したら、次のものが必要です。

   - `Dev team` グループ。
   - `Dev team` グループにアタッチされた`PowerUserAccess`アクセス許可セット。
   - ユーザーが `Dev team` グループに追加されました。
3. ポータルを終了し、再度サインインして、 `Administrator`または の AWS アカウント およびオプションを確認します`PowerUserAccess`。ツール/SDK を使用する`PowerUserAccess`ときに を選択します。

雇用主が管理するフェデレーティッド ID プロバイダー (Microsoft Entra や Okta など) AWS を通じてに既にアクセスしている

ID プロバイダーのポータル AWS から にサインインします。Cloud Administrator がユーザー `PowerUserAccess` (開発者) にアクセス許可を付与している場合は、アクセスできる AWS アカウント とアクセス許可セットが表示されます。アクセス許可セットの名前の横に、そのアクセス許可セットを使用してアカウントに手動またはプログラムでアクセスするオプションが表示されます。

カスタム実装では、アクセス許可セット名が異なるなど、エクスペリエンスが異なる場合があります。どのアクセス許可セットを使用すればよいかわからない場合は、IT チームにお問い合わせください。

雇用主が管理する AWS アクセスポータル AWS から に既にアクセスできる

AWS アクセスポータル AWS から にサインインします。Cloud Administrator がユーザー `PowerUserAccess` (開発者) にアクセス許可を付与している場合は、アクセスできる AWS アカウント とアクセス許可セットが表示されます。アクセス許可セットの名前の横に、そのアクセス許可セットを使用してアカウントに手動またはプログラムでアクセスするオプションが表示されます。

雇用主が管理するフェデレーティッドカスタム ID プロバイダー AWS を通じて に既にアクセスできる

サポートについては、IT チームにお問い合わせください。

ステップ 2：IAM Identity Center を使用するように SDK とツールを設定します

1. 開発マシンに最新の AWS CLIをインストールします。

   a. 「AWS Command Line Interface ユーザーガイド」の「AWS CLI の最新バージョンをインストールまたは更新します。」を参照してください。

   b. (オプション) AWS CLI が動作していることを確認するには、コマンドプロンプトを開き、`aws --version` コマンドを実行します。

2. AWS アクセスポータルにサインインします。この URL は、雇用主から提供されたり、「ステップ 1: アクセスを確立する」の後に E メールで取得したりする場合があります。そうでない場合は、https://console.aws.amazon.com/singlesignon/ の ダッシュボードで AWS アクセスポータル URL を見つけます。

   a. AWS アクセスポータルの アカウント タブで、管理する個々のアカウントを選択します。ユーザーのロールが表示されます。アクセスキーを選択して、コマンドライン用の認証情報または適切なアクセス許可セット用のプログラムによるアクセスを取得します。事前定義さ

れた PowerUserAccess 許可セットを使用するか、またはユーザーもしくは雇用主が開発のために最小特権の許可を適用するために作成した許可セットを使用してください。

b. [認証情報の取得] ダイアログボックスで、オペレーティングシステムに応じて、[MacOS と Linux] または [Windows] を選択します。

c. [IAM Identity Center 認証情報] メソッドを選択して、次のステップに必要な SSO Start URL と SSO Region の値を取得します。

3. AWS CLI コマンドプロンプトで、 aws configure sso コマンドを実行します。プロンプトが表示されたら、前のステップで収集した設定値を入力します。この AWS CLI コマンドの詳細については、aws configure sso「ウィザードでプロファイルを設定する」を参照してください。

   - [CLI プロファイル名]には、開始時に#####を入力することをお勧めします。デフォルト以外の（名前付き）プロファイルとそれに関連する環境変数を設定する方法については、「プロファイル」を参照してください。

4. （オプション) AWS CLI コマンドプロンプトで、 aws sts get-caller-identity コマンドを実行してアクティブなセッション ID を確認します。レスポンスには、設定した IAM Identity Center アクセス許可セットが表示されるはずです。

5. AWS SDK を使用している場合は、開発環境で SDK 用のアプリケーションを作成します。

   a. 一部の SDK では、IAM Identity Center 認証を使用する前に、SSO や SSOOIDC などの追加パッケージをアプリケーションに追加する必要があります。詳細については、具体的な SDK を参照してください。

   b. へのアクセスを以前に設定している場合は AWS、共有 AWS credentialsファイルで を確認しますAWS アクセスキー。認証情報プロバイダーチェーン 優先順位により、SDK またはツールが IAM Identity Center の認証情報を使用する前に、静的認証情報をすべて削除する必要があります。

SDK とツールがこの設定を使用して認証情報を使用および更新する方法についての詳細は、「IAM Identity Center 認証を理解する」を参照してください。

設定したセッションの長さによっては、アクセスが最終的に期限切れになり、SDK で認証エラーが発生します。必要に応じてアクセスポータルセッションを再度更新するには、 を使用して aws sso login コマンド AWS CLI を実行します。

IAM Identity Center アクセスポータルのセッション期間とアクセス許可セットのセッション期間の両方を延長できます。これにより、 AWS CLIに手動でサインインし直す必要が出るまでにコードを実

行できる時間が長くなります。詳細については、『AWS IAM Identity Center ユーザーガイド:』の以下のトピックを参照してください。

- IAM Identity Center セッション期間 — ユーザーの AWS アクセスポータルセッションの期間を設定します。
- アクセス許可セットセッション期間 – セッション期間を設定します

SDK およびツール用の、すべての IAM アイデンティティセンターのプロバイダー設定の詳細については、このガイドの「IAM Identity Center 認証情報プロバイダー」を参照してください。

IAM Identity Center 認証を理解する

IAM Identity Center の関連条項

以下の用語は、AWS IAM Identity Center の背後にあるプロセスと設定を理解するのに役立ちます。AWS SDK API のドキュメントでは、これらの認証概念の一部に IAM Identity Center とは異なる名前を使用しています。両方の名前を知っておくと役に立ちます。

次の表は、別名の相互関係を示しています。

IAM アイデンティティセンターの名前	SDK API の名前	説明
アイデンティティセンター	`sso`	AWS シングルサインオンの名前は変更されましたが、`sso` API 名前空間は下位互換性のために元の名前を維持します。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 Identity Center コンソール 管理コンソール		シングルサインオンの設定に使用するコンソール。

IAM アイデンティティセンターの名前	SDK API の名前	説明
AWS アクセスポータル URL		IAM Identity Center のアカウントに一意の URL（`https://xxx.awsapps.com/start` など）。IAM Identity Center のサインイン認証情報を使用してこのポータルにサインインします。
IAM Identity Center アクセスポータルセッション	認証セッション	発信者がベアラーアクセストークンを取得できます。
アクセス許可設定セッション		SDK が内部的に AWS のサービス 呼び出しに使用する IAM セッション。非公式な議論では、このセッションが誤って「ロールセッション」と呼ばれることがあります。
アクセス許可セット認証情報	AWS 認証情報 sigv4 認証情報	SDK がほとんどの AWS のサービス 呼び出し（具体的にはすべての sigv4 AWS のサービス 呼び出し）で実際に使用する認証情報。非公式な議論では、このセッションが誤って「ロール認証情報」と呼ばれることがあります。
IAM Identity Center 認証情報プロバイダー	SSO 認証情報プロバイダー	認証情報の取得方法（機能を提供するクラスやモジュールなど）。

AWS のサービス の SDK 認証情報解決について理解します

IAM Identity Center API は、ベアラートークンの認証情報を sigv4 の認証情報と交換します。Amazon CodeWhisperer や Amazon CodeCatalyst のようないくつかの例外を除いて、AWS のサービス はほとんど sigv4 API です。以下では、AWS IAM Identity Center を通じてアプリケーションコードのほとんどの AWS のサービス 呼び出しをサポートするための認証情報解決プロセスについて説明します。

AWS アクセスポータルセッションを開始する

- まず、認証情報を使用してセッションにサインインします。
  - AWS Command Line Interface (AWS CLI) の `aws sso login` コマンドを使用します。アクティブなセッションがまだない場合は、新しい IAM Identity Center セッションが開始されます。
- 新しいセッションを開始すると、IAM Identity Center から更新トークンとアクセストークンを受け取ります。AWS CLI はまた、SSO キャッシュ JSON ファイルを新しいアクセストークンと更新トークンで更新し、SDK で使用できるようにします。
- すでにアクティブなセッションがある場合、AWS CLI コマンドは既存のセッションを再利用し、既存のセッションの有効期限が切れると期限切れになります。IAM Identity Center セッションの長さを設定する方法については、「AWS IAM Identity Center ユーザーガイド」の「ユーザーの AWS アクセスポータルセッションの期間の設定」を参照してください。
  - 頻繁にサインインする必要性を減らすため、セッションの最大期間が 90 日間に延長されました。

SDK が AWS のサービス 呼び出しの認証情報を取得する方法

SDK を使用すると、サービスごとにクライアントオブジェクトをインスタンス化するときに AWS のサービス へのアクセスができるようになります。共有 AWS `config` ファイルの選択したプロファイルが IAM Identity Center の認証情報解決用に設定されている場合、IAM Identity Center を使用してアプリケーションの認証情報を解決します。

- 認証情報解決プロセスは、ランタイムにクライアントが作成されるときに完了します。

IAM Identity Center のシングルサインオンを使用して sigv4 API の認証情報を取得するために、SDK は IAM Identity Center のアクセストークンを使用して IAM セッションを取得します。この IAM セッションはアクセス許可セットセッションと呼ばれ、IAM ロールを引き受けることで SDK への AWS アクセスが可能となります。

- アクセス許可セットのセッション期間は IAM Identity Center のセッション期間とは独立して設定されます。
  - アクセス許可セットのセッション期間を設定する方法については、「AWS IAM Identity Center ユーザーガイド」の「セッション期間の設定」を参照してください。
- ほとんどの AWS SDK API ドキュメントでは、AWS 認証情報や sigv4 認証情報とも呼ばれていることに注意してください。

アクセス許可セットの認証情報は、IAM Identity Center API の getRoleCredentials への呼び出しから SDK に返されます。SDK のクライアントオブジェクトは、引き受けた IAM ロールを使用して AWS のサービス を呼び出します。たとえば、アカウントのバケットを一覧表示するように Amazon S3 に要求します。クライアントオブジェクトは、アクセス許可セットセッションの有効期限が切れるまで、それらのアクセス許可セット認証情報を使用して操作を続けることができます。

セッションの有効期限と更新

SSO トークンプロバイダー設定 を使用する場合、IAM Identity Center から取得した 1 時間単位のアクセストークンは、更新トークンを使用して自動的に更新されます。

- SDK がアクセストークンを使用しようとしたときにそのアクセストークンの有効期限が切れている場合、SDK は更新トークンを使用して新しいアクセストークンの取得を試みます。IAM Identity Center は、更新トークンを IAM Identity Center のアクセスポータルのセッション期間と比較します。更新トークンの有効期限が切れていない場合、IAM Identity Center は別のアクセストークンで応答します。
- このアクセストークンは、既存のクライアントのアクセス許可セットセッションを更新したり、新しいクライアントの認証情報を解決したりするために使用できます。

ただし、IAM Identity Center アクセスポータルセッションの有効期限が切れると、新しいアクセストークンは付与されません。そのため、アクセス許可セットの有効期間は更新できません。既存のクライアントのキャッシュされたアクセス許可セットセッションの長さがタイムアウトになると、有効期限が切れます (アクセスも失われます) 。

IAM Identity Center セッションの有効期限が切れるとすぐに、新しいクライアントを作成するコードは認証に失敗します。これは、アクセス許可セットの認証情報がキャッシュされないためです。有効なアクセストークンが得られるまで、コードで新しいクライアントを作成したり、認証情報解決プロセスを完了したりすることはできません。

まとめると、SDK が新しいアクセス許可セット認証情報を必要とする場合、SDK はまず有効な既存の認証情報を確認し、それらを使用します。これは、認証情報が新しいクライアントのものか、認証情報の有効期限が切れた既存のクライアントのものかに関係なく適用されます。認証情報が見つからない、または有効でない場合、SDK は IAM Identity Center API を呼び出して新しい認証情報を取得します。API を呼び出すには、アクセストークンが必要です。アクセストークンの有効期限が切れている場合、SDK は更新トークンを使用して、IAM Identity Center サービスから新しいアクセストークンを取得しようとします。このトークンは、IAM Identity Center アクセスポータルセッションの有効期限が切れていない場合に付与されます。

IAM Roles Anywhere

IAM Roles Anywhere を使用すると、サーバー、コンテナ、アプリケーションなど、AWS の外部で実行されるワークロードに関する一時的なセキュリティ認証情報を IAM で取得することができます。IAM Roles Anywhere を使用するには、ワークロードで X.509 証明書を使用する必要があります。IAM Roles Anywhere を認証情報プロバイダーとして設定するのに必要な証明書とプライベートキーは、クラウド管理者が提供する必要があります。

ステップ 1：IAM Roles Anywhere を設定します

IAM Roles Anywhere は、AWS の外部で実行されるワークロードまたはプロセスの一時的な認証情報を取得する方法を提供します。認証機関との間でトラストアンカーが確立され、関連する IAM ロールの一時的な認証情報を取得できます。このロールは、コードが IAM Roles Anywhere で認証されるときにワークロードが持つアクセス許可を設定します。

トラストアンカー、IAM ロール、IAM Roles Anywhere プロファイルを設定する手順については、「IAM Roles Anywhere ユーザーガイド」の「AWS Identity and Access Management Roles Anywhere でトラストアンカーとプロファイルの作成」を参照してください。

> **Note**
>
> 「IAM Roles Anywhere ユーザーガイド」のプロファイルは、IAM Roles Anywhere サービス内の独特の概念を指しています。共有 AWS `config` ファイル内のプロファイルとは関係ありません。

ステップ 2：IAM Roles Anywhere の使用

IAM Roles Anywhere から一時的なセキュリティ認証情報を取得するには、IAM Roles Anywhere にある認証情報ヘルパーツールを使用してください。この認証情報ツールは IAM Roles Anywhere の署名プロセスを実装します。

認証情報ヘルパーツールをダウンロードする手順については、「IAM Roles Anywhere ユーザーガイド」の「AWS Identity and Access Management Roles Anywhere からの一時的なセキュリティ認証情報の取得」を参照してください。

IAM Roles Anywhere の一時的なセキュリティ認証情報を AWS SDK と AWS CLI で使用するには、共有 AWS `config` ファイルに `credential_process` 設定を設定できます。SDK と AWS CLI は、認証に `credential_process` を使用するプロセス認証情報プロバイダーをサポートします。`credential_process` を設定する一般的な構造を以下に示します。

```
credential_process = [path to helper tool] [command] [--parameter1 value] [--
parameter2 value] [...]
```

ヘルパーツールの `credential-process` コマンドは、`credential_process` 設定と互換性のある標準 JSON 形式で一時的な認証情報を返します。コマンド名にはハイフンが含まれていますが、設定名にはアンダースコアが含まれていることに注意してください。コマンドには以下のパラメータが必要となります。

- `private-key` – リクエストに署名したプライベートキーへのパス。
- `certificate` – 証明書へのパス。
- `role-arn` – 一時的な認証情報を取得するロールの ARN。
- `profile-arn` – 指定されたロールのマッピングを行うプロファイルの ARN。
- `trust-anchor-arn` – 認証に使用するトラストアンカーの ARN。

クラウド管理者は、証明書とプライベートキーを提供する必要があります。3 つの ARN 値はすべて AWS Management Console からコピーできます。次の例は、ヘルパーツールから一時的な認証情報を取得するように設定した共有 `config` ファイルを示しています。

```
[profile dev]
credential_process = ./aws_signing_helper credential-process --certificate /
path/to/certificate --private-key /path/to/private-key --trust-anchor-
arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID --profile-
```

```
arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID --role-
arn arn:aws:iam::account:role/ROLE_ID
```

オプションのパラメータとその他のヘルパーツールの詳細については、GitHub の「IAM Roles Anywhere 認証情報ヘルパー」を参照してください。

SDK 設定自体とプロセス認証情報プロバイダーの詳細については、このガイドの「プロセス認証情報プロバイダー」を参照してください。

ロールの割り当て

ロールでは、他の方法ではアクセスできない AWS リソースへのアクセスに、一時的なセキュリティ認証情報のセットを使用する必要があるとします。これらの一時的な認証情報は、アクセスキーID、シークレットアクセスキー、およびセキュリティトークンで構成されています。AWS Security Token Service (AWS STS) API リクエストの詳細については、「AWS Security Token Service API リファレンス」の「アクション」を参照してください。

ロールを引き受けるように SDK またはツールを設定するには、まず引き受けるのための特定のロールを作成または特定する必要があります。IAM ロールは、ロール (Amazon リソースネーム (「ARN」)で一意に識別されます。ロールは別のエンティティとの信頼関係を確立します。ロールを使用する信頼できるエンティティは AWS のサービス 、別の AWS アカウント 、ウェブ ID プロバイダーまたはOIDC、または SAML フェデレーションである可能性があります。ロールの作成と使用の詳細については、「IAM ユーザーガイド」の「IAM ロールを使用する」を参照してください。

IAM ロールが特定されると、そのロールから信頼されている場合は、そのロールによって付与された権限を使用するように SDK またはツールを設定できます。これを実行するには、IAM ロールの継承 または ウェブアイデンティティまたは OpenIDコネクトとのフェデレーション のいずれかを使用します。

IAM ロールの継承

ロールを引き受けると、 AWS STS は一時的なセキュリティ認証情報のセットを返します。これらの認証情報は、別のプロファイル、またはコードが実行されているインスタンスまたはコンテナから取得されます。ロールを引き受ける他の例としては、Amazon EC2 から複数の AWS アカウント を管理したり、AWS アカウント にわたって AWS CodeCommit を使用したり、 AWS CodeBuild から別のアカウントにアクセスしたりすることが挙げられます。

ステップ 1: IAM ロールを設定する

ロールを引き受けるように SDK またはツールを設定するには、まず引き受けるのための特定のロールを作成または特定する必要があります。IAM ロールはロール「ARN」 を使用して一意に識別されます。ロールは別のエンティティとの信頼関係を確立します。通常はアカウント内またはクロスアカウントアクセス用です。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

ステップ 2: SDK またはツールを設定する

`credential_source` または `source_profile` から認証情報を取得するように SDK またはツールを設定します。

`credential_source` を使用してAmazon ECS コンテナ、Amazon EC2 インスタンス、または環境変数から認証情報を取得します。

`source_profile` を使用して別のプロファイルから認証情報を取得します。 `source_profile` はまた、ロールチェイニングもサポートしています。ロールチェイニングとは、引き受けたロールを使って別のロールを引き受けるプロファイルの階層構造です。

これをプロファイルで指定すると、SDK またはツールは自動的に対応する AWS STS`AssumeRole` API コールを行います。ロールを引き受けることで一時的な認証情報を取得、使用するには、AWS `config` 共有ファイルに次の設定値を指定します。これらの設定の詳細については、「ロール認証情報プロバイダーを引き受けます」を参照してください。

- `role_arn` - ステップ 1 で作成された IAM ロール
- `source_profile` または `credential_source` のいずれかを設定します
- (オプション) `duration_seconds`
- (オプション) `external_id`
- (オプション) `mfa_serial`
- (オプション) `role_session_name`

次の例は、 `config` 共有ファイル内の 2 つの引き受けロールオプションの設定を示しています。

```
role_arn = arn:aws:iam::123456789012:role/my-role-name
source_profile = profile-name-with-user-that-can-assume-role
```

```
role_arn = arn:aws:iam::123456789012:role/my-role-name
```

```
credential_source = Ec2InstanceMetadata
```

すべての引き受けロールの認証情報プロバイダーの設定の詳細については、このガイドの「ロール認証情報プロバイダーを引き受けます」を参照してください。

ウェブアイデンティティまたは OpenIDコネクトとのフェデレーション

AWS へのアクセスを必要とするモバイルアプリケーションまたはクライアントベースのウェブアプリケーションを作成すると、 AWS STS はパブリックID プロバイダー (IdP) を通じて認証された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返します。パブリック ID プロバイダーの例としては、Login with Amazon、Facebook、Google、または OpenID Connect (OIDC) に対応している任意の ID プロバイダーがあります。この方法では、ユーザーは自分の AWS や IAM ID を必要としません。

Amazon Elastic Kubernetes Service を使用している場合、この機能により、コンテナごとに異なる IAM ロールを指定できます。Kubernetes には、この認証情報プロバイダーが一時的な認証情報を取得するために使用する OIDC トークンをコンテナに配布する機能があります。この Amazon EKS の設定の詳細については、「Amazon EKS ユーザーガイド」の「サービスアカウントの IAM ロール」を参照してください。ただし、より単純なオプションとして、SDK がサポートしている場合は、代わりに Amazon EKS Pod Identities を利用することをお勧めします。

ステップ 1: ID プロバイダーと IAM ロールを設定する

外部 IdP サービスとのフェデレーションを設定するには、IAM の ID プロバイダーを作成し、外部 IdP とその設定について AWS に通知します。これにより、AWS アカウント と外部 IdP の間の「信頼」が確立されます。認証のためにウェブ ID トークンを使用するように SDK を設定する前に、まず ID プロバイダー (IdP) と、それにアクセスするための IAM ロールを設定する必要があります。これらを設定するには、「IAM ユーザーガイド」の「ウェブ OpenID Connect フェデレーション用のロールの作成 (コンソール)」 を参照してください。

ステップ 2: SDK またはツールを設定する

認証に使用したウェブ ID トークンを使用するように SDK またはツールを AWS STS から設定します。

これをプロファイルで指定すると、SDK またはツールは自動的に対応する AWS STS`AssumeRoleWithWebIdentity` API コールを行います。ウェブ ID フェデレーションを使用して一時的な認証情報を取得、使用するには、AWS`config` 共有プロファイルで以下の設定値を指定します。これらの設定の詳細については、「ロール認証情報プロバイダーを引き受けます」を参照してください。

- `role_arn` - ステップ 1 で作成された IAM ロール
- `web_identity_token_file`-外部 IdP から
- (オプション) `duration_seconds`
- (オプション) `role_session_name`

ウェブ IDを使用してロールを引き受ける `config` 共有ファイル設定の例を次に示します。

```
[profile web-identity]
role_arn=arn:aws:iam::123456789012:role/my-role-name
web_identity_token_file=/path/to/a/token
```

> **Note**
>
> モバイルアプリケーションに対しては、Amazon Cognito の使用をお勧めします。Amazon Cognito は ID ブローカーとして機能し、ユーザーの代わりに多くのフェデレーション作業を行います。ただし、Amazon Cognito ID プロバイダーは、他の ID プロバイダーのように SDK やツールのコアライブラリには含まれていません。Amazon Cognito API にアクセスするには、SDK またはツールのビルドまたはライブラリに Amazon Cognito サービスクライアントを含めてください。AWS SDK での使用方法については、「Amazon Cognito 開発者ガイド」の「コード例」 を参照してください。

すべての引き受けロールの認証情報プロバイダーの設定の詳細については、このガイドの「ロール認証情報プロバイダーを引き受けます」を参照してください。

AWS アクセスキー

短期の認証情報を使用します

セッション期間の長いオプションを使用するには、IAM Identity Center 認証 を使用するように SDK またはツールを設定することをお勧めします。

ただし、SDK またはツールの一時認証情報を直接設定する方法については、「短期の認証情報を使用した認証」を参照してください。

長期認証情報の使用

> **Warning**
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

全体のアクセスを管理する AWS アカウント

セキュリティのベストプラクティスとして、IAM Identity Center AWS Organizations で を使用して、すべての へのアクセスを管理することをお勧めします AWS アカウント。詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティベストプラクティス」を参照してください。

IAM Identity Center でユーザーを作成する、Microsoft Active Directory を使用する、SAML 2.0 ID プロバイダー (IdP) を使用する、または IdP を に個別にフェデレーションすることができます AWS アカウント。これらのアプローチのいずれかを使用して、ユーザーにシングルサインオンのエクスペリエンスを提供できます。また、多要素認証 (MFA) を強制し、 AWS アカウント アクセスに一時的な認証情報を使用することもできます。これは IAM ユーザーとは異なります。IAM ユーザーは、共有できる長期的な認証情報であり、 AWS リソースに対するセキュリティリスクが高まる可能性があります。

サンドボックス環境専用の IAM ユーザーを作成する

を初めて使用する場合は AWS、テスト IAM ユーザーを作成し、それを使用してチュートリアルを実行し、 が提供する AWS ものを調べることができます。学習中はこの種の資格情報を使用しても問題ありませんが、サンドボックス環境以外では使用しないことをお勧めします。

以下のユースケースでは、 で IAM ユーザーの使用を開始するのが理にかなっている場合があります AWS。

- AWS SDK またはツールの使用を開始し、 AWS のサービス サンドボックス環境で を探索します。
- 学習の一環として、人間によるサインインプロセスをサポートしない、スケジュールされたスクリプト、ジョブ、その他の自動プロセスを実行する。

これらのユースケース以外で IAM ユーザーを使用している場合は、 AWS アカウント できるだけ早く IAM Identity Center に移行するか、ID プロバイダーを にフェデレーションしてください。詳細については、「AWSでの ID フェデレーション」を参照してください。

IAM ユーザーのアクセスキーを保護する

IAM ユーザーのアクセスキーは定期的に更新する必要があります。「IAM ユーザーガイド」の「アクセスキーの更新」のガイダンスに従ってください。IAM ユーザーのアクセスキーを誤って共有したと思われる場合は、アクセスキーを更新してください。

IAM ユーザーアクセスキーは、ローカルマシンの共有 AWS `credentials`ファイルに保存する必要があります。IAM ユーザーのアクセスキーをコードに保存しないでください。IAM ユーザーのアクセスキーを含む設定ファイルは、いずれのソースコード管理ソフトウェアにも含めないでください。オープンソースプロジェクトの git-secrets などの外部ツールを使用すると、機密情報を誤って Git リポジトリにコミットすることを防ぐことができます。詳細については、「IAM ユーザーガイド」の「IAM アイデンティティ (ユーザー、ユーザーグループ、ロール)」を参照してください。

はじめに IAM ユーザーを設定するには、「 長期認証情報を使用した認証」を参照してください。

短期の認証情報を使用した認証

セッション期間の長いオプションで IAM Identity Center 認証 を使用するように SDK またはツールを設定することをお勧めします。ただし、AWS アクセスポータルにある一時的な認証情報をコピーして使用することもできます。有効期限が切れたら、新しい認証情報をコピーする必要があります。一時的な認証情報は、プロファイルで使用することも、システムプロパティや環境変数の値として使用することもできます。

AWS アクセスポータルから取得した短期認証情報を使用して認証情報ファイルを設定します

1. 認証情報の共有ファイルの作成。
2. 認証情報ファイルに、作業用の一時認証情報を貼り付けるまで、次のプレースホルダーテキストを貼り付けます。

```
[default]
aws_access_key_id=<value from AWS access portal>
aws_secret_access_key=<value from AWS access portal>
aws_session_token=<value from AWS access portal>
```

3. ファイルを保存します。これで、`~/.aws/credentials` ファイルはローカルの開発システムに存在しているはずです。このファイルには、特定の名前付きプロファイルが指定されていない場合に SDK またはツールが使用する [デフォルト] プロファイルが含まれています。
4. AWS アクセスポータルにサインインします。
5. 手動での認証情報更新のために、次の手順に従って、AWS アクセスポータルから IAM ロール認証情報をコピーします。
   a. リンク先の手順のステップ 4 で、開発ニーズに合ったアクセスを許可する IAM ロールの名前を選択します。通常、このロールには PowerUserAccess や Developer などの名前が付いています。
   b. リンク先の手順のステップ 7 で、[AWS 認証情報ファイルにプロファイルを手動で追加] オプションを選択し、内容をコピーします。
6. コピーした認証情報をローカル `credentials` ファイルに貼り付けます。`default` プロファイルを使用する場合、生成されたプロファイル名は必要ありません。ファイルは以下のようになります。

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONG
```

7. `credentials` ファイルを保存します。

SDK は、サービスクライアントを作成するときに、これらの一時的な認証情報にアクセスしてリクエストごとに使用します。ステップ 5a で選択した IAM ロールの設定により、一時的な認証情報の有効期間が決まります。最大期間は 12 時間です。

一時的な認証情報の有効期限が切れたら、ステップ 4 ~ 7 を繰り返します。

長期認証情報を使用した認証

> **Warning**
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

IAM ユーザーを使用してコードを実行する場合、開発環境の SDK またはツールは、共有 AWS `credentials`ファイルで長期的な IAM ユーザー認証情報を使用して認証します。「IAM トピックのセキュリティのベストプラクティス」 を確認し、できるだけ早く IAM Identity Center またはその他の一時的な認証情報に移行してください。

認証情報に関する重要な警告とガイダンス

認証情報に関する警告

- お使いのアカウントのルート認証情報を使用して AWS リソースにアクセスしないでください。これらの認証情報は無制限のアカウントアクセスを提供し、取り消すのが困難です。
- アプリケーションファイルにリテラルアクセスキーや認証情報を配置しないでください。これを行うと、パブリックリポジトリにプロジェクトをアップロードするなど、誤って認証情報が公開されるリスクが発生します。
- プロジェクト領域に認証情報を含むファイルを含めないでください。
- 共有 AWS `credentials`ファイルに保存されている認証情報はすべてプレーンテキストで保存されることに注意してください。

認証情報を安全に管理するための追加のガイダンス

AWS 認証情報を安全に管理する方法の一般的な説明については、「」の AWS 「アクセスキーを管理するためのベストプラクティス」を参照してくださいAWS 全般のリファレンス。そこでの説明に加えて、以下の点を考慮してください。

- Amazon Elastic Container Service (Amazon ECS) タスクで、タスク用の IAM ロールを使用します。
- Amazon EC2 インスタンスで実行中のアプリケーションに対して、IAM ロールを使用します。

前提条件: AWS アカウントを作成する

IAM ユーザーを使用して AWS サービスにアクセスするには、 AWS アカウントと AWS 認証情報が必要です。

1. アカウントを作成する。

   AWS アカウントを作成するには、「 AWS Account Management リファレンスガイド」の「開始方法: 初めての AWS ユーザーですか？」を参照してください。

2. 管理者ユーザーを作成します。

   マネジメントコンソールとサービスへのアクセスには、root ユーザーアカウント (作成した初期アカウント) を使用しないでください。代わりに、「IAM ユーザーガイド」の「管理ユーザーを作成する」で説明されているように、管理ユーザーアカウントを作成します。

   管理ユーザーアカウントを作成してログインの詳細を記録したら、ルートユーザーアカウントから確実にサインアウトし、管理者アカウントを使用して再度サインインします。

これらのアカウントはいずれも、 での開発や でのアプリケーションの実行 AWS には適していません AWS。そのためには、これらのタスクに適したユーザー、アクセス許可セットまたはサービスロールを作成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「最小特権アクセス許可を適用する」を参照してください。

ステップ 2: IAM ユーザーを作成する

- 「IAM ユーザーガイド」の「IAM ユーザーの作成 (コンソール)」の手順に従って IAM ユーザーを作成します。IAM ユーザーを作成する場合：
  - へのユーザーアクセスを提供する AWS Management Console を選択することをお勧めします。これにより、診断ログの確認 AWS CloudTrail や Amazon Simple Storage Service へのファイルのアップロードなど、ビジュアル環境で実行しているコード AWS のサービス に関連する を表示できます。これは、コードをデバッグする場合に役立ちます。
  - アクセス許可の設定 - アクセス許可オプション で、このユーザーにアクセス許可を割り当てる方法については、ポリシーを直接アタッチする を選択します。
    - ほとんどの「開始方法」 SDK チュートリアルでは、Amazon S3 サービスを例として使用しています。アプリケーションに Amazon S3 へのフルアクセスを提供するには、このユーザーにアタッチする `AmazonS3FullAccess` ポリシーを選択します。
  - アクセス許可の境界またはタグの設定に関する、その手順のオプションステップは無視できます。

ステップ 2: アクセスキーを取得する

1. IAM コンソールのナビゲーションペインで [ユーザー] を選択し、以前に作成したユーザーの **`User name`** を選択します。
2. ユーザーのページで、[セキュリティ認証情報] ページを選択します。次に、[アクセスキー] で [アクセスキーの作成] を選択します。

3. [アクセスキーを作成ステップ1]で、[コマンドラインインターフェイス (CLI)]または[ローカルコード]を選択します。どちらのオプションも、 AWS CLI と SDKsの両方で使用するのと同じタイプのキーを生成します。
4. [アクセスキーの作成ステップ 2] で、オプションのタグを入力して [次へ] を選択します。
5. [アクセスキーの作成ステップ 3] で、[.csv ファイルをダウンロード] を選択し、IAM ユーザーのアクセスキーとシークレットアクセスキーを含む `.csv` ファイルを保存します。この情報は後で必要になります。

> **Warning**
> 適切なセキュリティ対策を講じてこれらの認証情報を安全に保管してください。

6. [完了] を選択します。

ステップ 3: **credentials** 共有ファイルを更新する

1. 共有 AWS `credentials` ファイルを作成するか、開きます。このファイルは、`~/.aws/credentials`LinuxおよびmacOSシステム、および`%USERPROFILE%\.aws\credentials`Windows上にあります。詳細については、「認証情報ファイルの場所」 を参照してください。
2. 共有 `credentials` ファイルに次のテキストを追加します。ID 値の例とキー値の例を、先にダウンロードした `.csv` ファイルの値に置き換えます。

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

3. ファイルを保存します。

認証情報を保存する最も一般的な方法は `credentials` 共有ファイルです。これらは環境変数として設定することもできます。AWS アクセスキーの環境変数名を参照してください。これは始めるための方法ですが、IAM Identity Center やその他の一時的な認証情報にできるだけ早く移行することをお勧めします。長期認証情報の使用から移行した後は、必ず `credentials` 共有ファイルからこれらの認証情報を削除してください。

Amazon EC2 インスタンスの IAM ロールの使用

この例では、Amazon EC2 インスタンスにデプロイされたアプリケーションで使用するために Amazon S3 アクセスを持つ AWS Identity and Access Management ロールの設定について説明します。 Amazon EC2

Amazon Elastic Compute Cloud インスタンスの場合、IAM ロールを指定し、そのロールへのアクセスを Amazon EC2 インスタンスに許可します。詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 の IAM ロール」またはAmazon EC2 ユーザーガイド」の「Amazon EC2 の IAM ロールAmazon EC2」を参照してください。 Amazon EC2

IAM ロールを作成する

Amazon S3 に読み取り専用アクセスを付与する IAM ロールを作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [ロール]、[ロールを作成] の順に選択します。
3. [信頼されたエンティティタイプ] で [信頼されたエンティティを選択] するため、[AWS のサービス] を選択します。
4. [Use case] (ユースケース) で [Amazon EC2] を選択し、[Next] (次へ) を選択します。
5. [権限の追加] では、ポリシーリストから [Amazon S3 読み取り専用アクセス] のチェックボックスを選択し、[次へ] を選択します。
6. ロールの名前を入力し、[ロールの作成] を選択します。この名前は Amazon EC2 インスタンスを起動するときに必要になるため、忘れないようにしてください。

Amazon EC2 インスタンスを起動して IAM ロールを指定します

IAM ロールで Amazon EC2 インスタンスを起動するには、Amazon EC2 コンソールを使用します。

Amazon EC2 ユーザーガイド」または「Amazon Amazon EC2」の指示に従ってインスタンスを起動します。

[Review Instance Launch (インスタンス作成の確認)] ページを開いたら、[Edit instance details (インスタンスの詳細の編集)] を選択します。[IAM role] (IAM ロール) で、前に作成した IAM ロールを選択します。指示にしたがって手順を完了します。

> **Note**
>
> そのインスタンスに接続するには、セキュリティグループとキーペアを作成するか、または既存のものを使用する必要があります。

この IAM と Amazon EC2 のセットアップで、Amazon EC2 インスタンスにアプリケーションをデプロイすることができます。これにより、Amazon S3 サービスへの読み取りアクセスが付与されます。

EC2 インスタンスへの接続

EC2 インスタンスに接続することで、サンプルアプリケーションをそのインスタンスに転送して、アプリケーションを実行できるようにします。また、インスタンスの起動に使用したキーペアのプライベート部分を含むファイル、すなわち PEM ファイルも必要です。

これを行うには、Amazon EC2 ユーザーガイド」またはAmazon EC2 ユーザーガイド」の接続手順に従います。接続する際は、開発マシンからインスタンスにファイルを転送できるように接続してください。

AWS ツールキットを使用している場合は、多くの場合、 ツールキットを使用してインスタンスに接続することもできます。詳細については、 ご利用されている Toolkit の特定のユーザーガイドをご参照ください。

EC2 インスタンスでのサンプルアプリケーションの実行

1. ローカルドライブからインスタンスにアプリケーションファイルをコピーします。

   インスタンスにファイルを転送する方法については、Amazon EC2 ユーザーガイド」またはAmazon EC2 ユーザーガイド」を参照してください。
2. アプリケーションを起動し、開発マシンと同じ実行結果が得られることを確認します。
3. （オプション）アプリケーションで、IAM ロールによって提供されている認証情報が使用されていることを確認します。

   a. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
   b. インスタンスを選択し、[Actions] (アクション)、[Instance Settings] (インスタンスの設定)、[Attach/Replace IAM Role] (IAM ロールの添付/置換) を使用して IAM ロールをデタッチします。

c. アプリケーションを再度実行して、認可エラーが返されることを確認します。

設定リファレンス

SDKs、 の言語固有の APIs を提供します AWS のサービス。認証、再試行動作など、API コールを正常に行うために必要な面倒な作業の一部はこれらによって処理されます。そのために、SDK にはリクエストに使用する認証情報の取得、各サービスで使用する設定の管理、グローバル設定に使用する値の取得といった柔軟な戦略があります。

設定の詳細については、以下のセクションを参照してください。

- AWS SDKs標準化された認証情報プロバイダー – 複数の SDK で標準化された共通の認証情報プロバイダー。
- AWS SDKs標準化された機能 – 複数の SDK で標準化された共通機能。

サービスクライアントの作成

にプログラムでアクセスするために AWS のサービス、SDKs は各 にクライアントクラス/オブジェクトを使用します AWS のサービス。たとえば、アプリケーションが Amazon EC2 にアクセスする必要がある場合、アプリケーションはそのサービスとインターフェイスをとる Amazon EC2 クライアントオブジェクトを作成します。次に、サービスクライアントを使用して、その AWS のサービスに対してリクエストを実行します。ほとんどの SDKs では、サービスクライアントオブジェクトはイミュータブルであるため、リクエストを行う各サービスと、異なる設定を使用して同じサービスにリクエストを行う新しいクライアントを作成する必要があります。

設定の優先順位

グローバル設定は、ほとんどの SDK でサポートされ、 AWS のサービス全体に幅広く影響する機能、認証情報プロバイダー、およびその他の機能を設定します。すべての SDK には、グローバル設定の値を見つけるための一連の場所 (またはソース) があります。設定検索の優先順位は次のとおりです。

1. コードまたはサービスクライアント自体に設定されている明示的な設定は、他の設定よりも優先されます。
   - 一部の設定はオペレーションごとに設定でき、呼び出すオペレーションごとに必要に応じて変更できます。 AWS CLI または の場合 AWS Tools for PowerShell、これらはコマンドラインに入力したオペレーションごとのパラメータの形式になります。SDK の場合、明示的な割り当て

は、 AWS のサービス クライアントまたは設定オブジェクトをインスタンス化するとき、または個々の API を呼び出すときに設定したパラメータの形式をとることができます。

2. Java/Kotlin のみ: 設定の JVM システムプロパティがチェックされます。設定されている場合は、その値を使用してクライアントが設定されます。
3. 環境変数が確認されます。設定されている場合は、その値を使用してクライアントが設定されます。
4. SDK は、共有`credentials`ファイルで 設定を確認します。設定されている場合、クライアントはそれを使用します。
5. 設定の共有`config`ファイル。設定が存在する場合、SDK はその設定を使用します。
   - `AWS_PROFILE` 環境変数または `aws.profile` JVM システムプロパティを使用して、SDK がロードするプロファイルを指定できます。
6. SDK ソースコード自体によって提供されるデフォルト値が最後に使用されます。

> **Note**
>
> SDK やツールによっては、チェックの順序が異なる場合があります。また、SDK やツールの中には、他の方法でパラメータを保存したり取得したりできるものもあります。例えば、は SDK ストア という追加のソース AWS SDK for .NET をサポートしています。SDK またはツールにのみ存在するプロバイダーについて詳しくは、使用している SDK またはツールの特定のガイドを参照してください。

順序によって、どのメソッドが優先され、他のメソッドをオーバーライドするかが決まります。たとえば、共有 `config` ファイルにプロファイルを設定した場合、そのプロファイルは SDK またはツールが最初に他の場所を確認した後にのみ検出され、使用されます。つまり、`credentials` ファイルに設定を入力すると、`config` ファイルにある設定の代わりにその設定が使用されます。環境変数に設定と値を設定すると、`credentials` と `config` ファイルの両方の設定がオーバーライドされます。最後に、個々のオペレーション (AWS CLI コマンドラインパラメータまたは API パラメータ) またはコード内の設定は、その 1 つのコマンドの他のすべての値をオーバーライドします。

`Config` ファイル設定リスト

次の表に示す設定は、共有 AWS `config`ファイルで割り当てることができます。これらはグローバルで、すべての AWS のサービスに影響します。SDKsとツールは、一意の設定と環境変数もサポー

トしている場合があります。個々の SDK またはツールでのみサポートされている設定と環境変数を確認するには、その特定の SDK またはツールガイドを参照してください。

設定名	詳細
api_versions	一般設定
aws_access_key_id	AWS アクセスキー
aws_secret_access_key	AWS アクセスキー
aws_session_token	AWS アクセスキー
ca_bundle	一般設定
credential_process	プロセス認証情報プロバイダー
credential_source	ロール認証情報プロバイダーを引き受けます
defaults_mode	スマート設定デフォルト
disable_request_compression	リクエスト圧縮
duration_seconds	ロール認証情報プロバイダーを引き受けます
ec2_metadata_service_endpoint	IMDS 認証情報プロバイダー
ec2_metadata_servi	IMDS 認証情報プロバイダー

設定名	詳細	
ce_endpoint_mode		
ec2_metadata_v1_disabled	IMDS 認証情報プロバイダー	
endpoint_discovery_enabled	エンドポイント検出	
endpoint_url	サービス固有のエンドポイント	
external_id	ロール認証情報プロバイダーを引き受けます	
ignore_configured_endpoint_urls	サービス固有のエンドポイント	
max_attempts	再試行動作	
metadata_service_num_attempts	Amazon EC2 インスタンスメタデータ	
metadata_service_timeout	Amazon EC2 インスタンスメタデータ	
mfa_serial	ロール認証情報プロバイダーを引き受けます	
output	一般設定	
parameter_validation	一般設定	
region	AWS リージョン	

設定名	詳細
request_min_compression_size_bytes	リクエスト圧縮
retry_mode	再試行動作
role_arn	ロール認証情報プロバイダーを引き受けます
role_session_name	ロール認証情報プロバイダーを引き受けます
s3_disable_multiregion_access_points	Amazon S3 マルチリージョンアクセスポイント
s3_use_arn_region	Amazon S3 アクセスポイント
sdk_ua_app_id	アプリケーション ID
source_profile	ロール認証情報プロバイダーを引き受けます
sso_account_id	IAM Identity Center 認証情報プロバイダー
sso_region	IAM Identity Center 認証情報プロバイダー
sso_registration_scopes	IAM Identity Center 認証情報プロバイダー
sso_role_name	IAM Identity Center 認証情報プロバイダー
sso_start_url	IAM Identity Center 認証情報プロバイダー
sts_regional_endpoints	AWS STS リージョン化されたエンドポイント

設定名	詳細
use_dualstack_endpoint	デュアルスタックと FIPS エンドポイント
use_fips_endpoint	デュアルスタックと FIPS エンドポイント
web_identity_token_file	ロール認証情報プロバイダーを引き受けます

Credentials ファイル設定リスト

次の表に示す設定は、共有 AWS credentialsファイルで割り当てることができます。これらはグローバルで、すべての AWS のサービスに影響します。SDKsとツールは、一意の設定と環境変数もサポートしている場合があります。個々の SDK またはツールでのみサポートされている設定と環境変数を確認するには、その特定の SDK またはツールガイドを参照してください。

設定名	詳細
aws_access_key_id	AWS アクセスキー
aws_secret_access_key	AWS アクセスキー
aws_session_token	AWS アクセスキー

環境変数の一覧

ほとんどの SDK でサポートされる環境変数は、以下の表に示されています。これらはグローバルで、すべての AWS のサービスに影響します。SDKsとツールは、一意の設定と環境変数もサポートしている場合があります。個々の SDK またはツールでのみサポートされている設定と環境変数を確認するには、その特定の SDK またはツールガイドを参照してください。

設定名	詳細
AWS_ACCESS_KEY_ID	AWS アクセスキー
AWS_CA_BUNDLE	一般設定
AWS_CONFIG_FILE	共有 config ファイルと credentials ファイルの場所
AWS_CONTAINER_AUTHORIZATION_TOKEN	コンテナ認証情報プロバイダー
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE	コンテナ認証情報プロバイダー
AWS_CONTAINER_CREDENTIALS_FULL_URI	コンテナ認証情報プロバイダー
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	コンテナ認証情報プロバイダー
AWS_DEFAULTS_MODE	スマート設定デフォルト
AWS_DISABLE_REQUEST_COMPRESSION	リクエスト圧縮

設定名	詳細	
AWS_EC2_METADATA_DISABLED	IMDS 認証情報プロバイダー	
AWS_EC2_METADATA_SERVICE_ENDPOINT	IMDS 認証情報プロバイダー	
AWS_EC2_METADATA_SERVICE_ENDPOINT_MODE	IMDS 認証情報プロバイダー	
AWS_EC2_METADATA_V1_DISABLED	IMDS 認証情報プロバイダー	
AWS_ENABLE_ENDPOINT_DISCOVERY	エンドポイント検出	
AWS_ENDPOINT_URL	サービス固有のエンドポイント	
AWS_ENDPOINT_URL_<SERVICE>	サービス固有のエンドポイント	
AWS_IAM_ROLE_ARN	ロール認証情報プロバイダーを引き受けます	
AWS_IAM_ROLE_SESSION_NAME	ロール認証情報プロバイダーを引き受けます	

設定名	詳細
AWS_IGNORE_CONFIGURED_ENDPOINT_URLS	サービス固有のエンドポイント
AWS_MAX_ATTEMPTS	再試行動作
AWS_METADATA_SERVICE_NUM_ATTEMPTS	Amazon EC2 インスタンスメタデータ
AWS_METADATA_SERVICE_TIMEOUT	Amazon EC2 インスタンスメタデータ
AWS_PROFILE	共有 config および credentials ファイル
AWS_REGION	AWS リージョン
AWS_REQUEST_MIN_COMPRESSION_SIZE_BYTES	リクエスト圧縮
AWS_RETRY_MODE	再試行動作
AWS_S3_DISABLE_MULTIREGION_ACCESS_POINTS	Amazon S3 マルチリージョンアクセスポイント
AWS_S3_USE_ARN_REGION	Amazon S3 アクセスポイント

設定名	詳細
AWS_SDK_UA_APP_ID	アプリケーション ID
AWS_SECRET_ACCESS_KEY	AWS アクセスキー
AWS_SESSION_TOKEN	AWS アクセスキー
AWS_SHARED_CREDENTIALS_FILE	共有 config ファイルと credentials ファイルの場所
AWS_STS_REGIONAL_ENDPOINTS	AWS STS リージョン化されたエンドポイント
AWS_USE_DUALSTACK_ENDPOINT	デュアルスタックと FIPS エンドポイント
AWS_USE_FIPS_ENDPOINT	デュアルスタックと FIPS エンドポイント
AWS_WEB_IDENTITY_TOKEN_FILE	ロール認証情報プロバイダーを引き受けます

JVM システムプロパティリスト

AWS SDK for Java および AWS SDK for Kotlin (JVM をターゲットとする) には、次の JVM システムプロパティを使用できます。JVM システムプロパティを設定する方法については、the section called “JVM システムプロパティの設定方法”「」を参照してください。

設定名	詳細	
aws.accessKeyId	AWS アクセスキー	
aws.configFile	共有 config ファイルと credentials ファイルの場所	
aws.defaultsMode	スマート設定デフォルト	
aws.disableEc2MetadataV1	IMDS 認証情報プロバイダー	
aws.disableRequestCompression	リクエスト圧縮	
aws.ec2MetadataServiceEndpoint	IMDS 認証情報プロバイダー	
aws.ec2MetadataServiceEndpointMode	IMDS 認証情報プロバイダー	
aws.endpointDiscoveryEnabled	エンドポイント検出	
aws.endpointUrl	サービス固有のエンドポイント	
aws.endpointUrl<ServiceName>	サービス固有のエンドポイント	

設定名	詳細	
aws.ignoreConfiguredEndpointUrls	サービス固有のエンドポイント	
aws.maxAttempts	再試行動作	
aws.profile	共有 config および credentials ファイル	
aws.region	AWS リージョン	
aws.requestMinCompressionSizeBytes	リクエスト圧縮	
aws.retryMode	再試行動作	
aws.roleArn	ロール認証情報プロバイダーを引き受けます	
aws.roleSessionName	ロール認証情報プロバイダーを引き受けます	
aws.s3DisableMultiRegionAccessPoints	Amazon S3 マルチリージョンアクセスポイント	
aws.s3UseArnRegion	Amazon S3 アクセスポイント	
aws.secretAccessKey	AWS アクセスキー	
aws.sessionToken	AWS アクセスキー	

設定名	詳細
aws.sharedCredentialsFile	共有 config ファイルと credentials ファイルの場所
aws.useDualstackEndpoint	デュアルスタックと FIPS エンドポイント
aws.useFipsEndpoint	デュアルスタックと FIPS エンドポイント
aws.userAgentAppId	アプリケーション ID
aws.webIdentityTokenFile	ロール認証情報プロバイダーを引き受けます

AWS SDKs標準化された認証情報プロバイダー

多くの認証情報プロバイダーは、一貫したデフォルト値に標準化されており、多くの SDK で同じように動作します。この一貫性により、複数の SDK 間のコーディングの生産性と明確性が向上します。すべての設定はコード内で上書きすることができます。詳細については、お使いの特定の SDK API を参照してください。

> **Important**
>
> すべての SDK がすべてのプロバイダーをサポートしているわけではなく、プロバイダー内のあらゆる側面をサポートしているわけでもありません。

トピック

- 認証情報プロバイダーチェーン
- AWS アクセスキー
- ロール認証情報プロバイダーを引き受けます

- コンテナ認証情報プロバイダー
- IAM Identity Center 認証情報プロバイダー
- IMDS 認証情報プロバイダー
- プロセス認証情報プロバイダー

認証情報プロバイダーチェーン

すべての SDK には、 AWS のサービスへのリクエストに使用する有効な認証情報を確認するための一連の場所 (またはソース) があります。有効な認証情報が見つかると、検索は停止されます。この体系的な検索は、デフォルトの認証情報プロバイダーチェーンと呼ばれます。

SDK によって使用される個別のチェーンは異なりますが、ほとんどの場合、次のようなソースが含まれます。

認証情報プロバイダー	説明
AWS アクセスキー	AWS IAM ユーザーの アクセスキー (、`AWS_ACCESS_KEY_ID` 、 など`AWS_SECRET_ACCESS_KEY`) 。
ウェブアイデンティティまたはOpenIDコネクトとのフェデレーション - ロール認証情報プロバイダーを引き受けます	よく知られている外部 ID プロバイダー (IdP) (例: Login with Amazon、Facebook、Google などの OpenID Connect (OIDC) 互換の IdP) を使用してサインインします。 AWS Security Token Service () のウェブ ID トークンを使用して IAM ロールのアクセス許可を引き受けます AWS STS。
IAM Identity Center 認証情報プロバイダー	から認証情報を取得します AWS IAM Identity Center。
ロール認証情報プロバイダーを引き受けます	IAM ロールのアクセス許可を引き受けることで、他のリソースにアクセスできます。 (ロールの一時認証情報を取得して使用します) 。
コンテナ認証情報プロバイダー	Amazon Elastic Container Service (Amazon ECS) および Amazon Elastic Kubernetes Service (Amazon EKS) の認証情報。コンテナ認証情報プロバイダーは、お客様のコ

認証情報プロバイダー	説明
	ンテナ化されたアプリケーションの認証情報を取得します。
プロセス認証情報プロバイダー	カスタム認証情報プロバイダー。認証情報を外部ソースまたはプロセス (IAM Roles Anywhere) から取得します。
IMDS 認証情報プロバイダー	Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイルの認証情報。IAM ロールを各 EC2 インスタンスに関連付けます。そのロールの一時認証情報は、インスタンスで実行中のコードで使用できるようになります。認証情報は、Amazon EC2 メタデータサービスを通じて配信されます。

チェーン内の各ステップには、設定値を割り当てる方法が複数あります。コードで指定されている設定値が常に優先されます。ただし、環境変数 と 共有 `config` ファイルおよび `credentials` ファイル もあります。詳細については、「設定の優先順位」を参照してください。

AWS アクセスキー

> **Warning**
>
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

AWS IAM ユーザーの アクセスキーを AWS 認証情報として使用できます。 AWS SDK は、これらの AWS 認証情報を自動的に使用して への API リクエストに署名します。これにより AWS、ワークロードが AWS リソースとデータに安全かつ便利にアクセスできるようになります。認証情報が一時的なもので、有効期限が切れると無効になるように、常に `aws_session_token` を使用することをおすすめします。長期的な認証情報の使用はお勧めしません。

> **Note**
>
> AWS がこれらの一時的な認証情報を更新できない場合、ワークロードに影響が及ばないように認証情報の有効性 AWS を拡張できます。

共有 AWS `credentials`ファイルは、アプリケーションソースディレクトリの外部で安全であり、共有`config`ファイルの SDK 固有の設定とは別のため、認証情報を保存するための推奨場所です。

AWS 認証情報とアクセスキーの使用の詳細については、「IAM ユーザーガイド」のAWS 「セキュリティ認証情報」および「IAM ユーザーのアクセスキーの管理」を参照してください。 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

この機能を設定するには、以下のように使用します。

`aws_access_key_id` - 共有 AWS **`config`**ファイル設定, **`aws_access_key_id`** - 共有 AWS **`credentials`**ファイル設定（推奨メソッド）, **`AWS_ACCESS_KEY_ID`** - 環境変数, **`aws.accessKeyId`** - JVM システムプロパティ: Java/Kotlin のみ

ユーザーを認証するための認証情報の一部として使用される AWS アクセスキーを指定します。

`aws_secret_access_key` - 共有 AWS **`config`**ファイル設定, **`aws_secret_access_key`** - 共有 AWS **`credentials`**ファイル設定（推奨メソッド）, **`AWS_SECRET_ACCESS_KEY`** - 環境変数, **`aws.secretAccessKey`** - JVM システムプロパティ: Java/Kotlin のみ

ユーザーを認証するための認証情報の一部として使用される AWS シークレットキーを指定します。

`aws_session_token` - 共有 AWS **`config`**ファイル設定, **`aws_session_token`** - 共有 AWS **`credentials`**ファイル設定（推奨メソッド）, **`AWS_SESSION_TOKEN`** - 環境変数, **`aws.sessionToken`** - JVM システムプロパティ: Java/Kotlin のみ

ユーザーを認証するための認証情報の一部として使用される AWS セッショントークンを指定します。この値は、ロールを引き受けるリクエストが正常に終了すると返される一時的な認証情報の一部として受け取ります。セッショントークンは、一時的なセキュリティ認証情報を手動で指定する場合にのみ必要です。ただし、長期の認証情報ではなく、一時的なセキュリティ認証情報を常に使用することをお勧めします。セキュリティの推奨事項については、「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。

これらの値を取得する方法については、「短期の認証情報を使用した認証」を参照してください。

config または credentials ファイルに必要な値を設定する例を以下に示します。

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export
 AWS_SESSION_TOKEN=AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
setx
 AWS_SESSION_TOKEN AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	はい	共有 config ファイルはサポートされていません。
SDK for Go V2 (1.x)	はい	

SDK	サポート	注意または詳細情報
SDK for Go 1.x (V1)	はい	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	はい	
SDK for Java 1.x	はい	
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	はい	
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	環境変数はサポートされていません。
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	はい	

SDK	サポート	注意または詳細情報
のツール PowerShell	はい	環境変数はサポートされていません。

ロール認証情報プロバイダーを引き受けます

ロールでは、他の方法ではアクセスできない AWS リソースへのアクセスに、一時的なセキュリティ認証情報のセットを使用する必要があるとします。これらの一時的な認証情報は、アクセスキーID、シークレットアクセスキー、およびセキュリティトークンで構成されています。

ロールを引き受けるように SDK またはツールを設定するには、まず引き受けるのための特定のロールを作成または特定する必要があります。IAM ロールは、ロール (Amazon リソースネーム (「ARN」)で一意に識別されます。ロールは別のエンティティとの信頼関係を確立します。ロールを使用する信頼されたエンティティは AWS のサービス、、別の AWS アカウント、ウェブ ID プロバイダーまたは OIDC、または SAML フェデレーションです。

IAM ロールが特定されると、そのロールから信頼されている場合は、そのロールによって付与された権限を使用するように SDK またはツールを設定できます。これを行うには、次のコマンドを使用します。

これらの設定の使用を開始するためのガイダンスについては、このガイドの「ロールの割り当て」を参照してください。

ロール認証情報プロバイダーを引き受けます

この機能を設定するには、以下のように使用します。

`credential_source` - 共有 AWS **`config`**ファイル設定

Amazon EC2 インスタンスまたは Amazon Elastic Container Service のコンテナ内で使用され、`role_arn` パラメータで指定したロールを引き受けるために使用する認証情報を SDK またはツールが検索できる場所を指定します。

デフォルト値： なし

有効値:

- Environment – SDK またはツールが環境変数 AWS_ACCESS_KEY_ID と AWS_SECRET_ACCESS_KEY からソース認証情報を取得することを指定します。
- Ec2InstanceMetadata – SDK またはツールが EC2 インスタンスプロファイルにアタッチされた IAM ロールを使用してソース認証情報を取得することを指定します。
- EcsContainer – SDK またはツールが ECS コンテナにアタッチされた IAM ロールを使用してソース認証情報を取得することを指定します。

credential_source と source_profile の両方を同じプロファイルで指定することはできません。

認証情報を Amazon EC2 から取得する必要があることを示すために、config ファイルにこれを設定する例を以下に示します。

```
credential_source = Ec2InstanceMetadata
role_arn = arn:aws:iam::123456789012:role/my-role-name
```

duration_seconds - 共有 AWS **config**ファイル設定

ロールセッションの最大期間を秒単位で指定します。

この設定は、プロファイルがロールの継承を指定している場合にのみ適用されます。

デフォルト値： 3600 秒 (1 時間) です

有効な値： この値は 900 秒 (15 分) からロールの最大セッション期間設定 (上限は 43200、すなわち12時間) までの範囲を指定できます。詳細については、「IAM ユーザーガイド」の「ロールの最大セッション期間設定を表示する」 を参照してください。

config ファイルにこれを設定する例を以下に示します。

```
duration_seconds = 43200
```

external_id - 共有 AWS **config**ファイル設定

お客様のアカウントでサードパーティーがロールを引き受けるために使用される独自の識別子を指定します。

この設定は、プロファイルが役割を引き受けるように指定していて、その役割の信頼ポリシーで ExternalId の値が必要な場合にのみ適用されます。この値は、プロファイルがロールを指定するときに AssumeRole 操作に渡される ExternalId パラメータにマップされます。

デフォルト値： NONE。

有効な値：「IAM ユーザーガイド」の AWS 「 リソースへのアクセスを第三者に付与するときに外部 ID を使用する方法」を参照してください。

config ファイルにこれを設定する例を以下に示します。

```
external_id = unique_value_assigned_by_3rd_party
```

mfa_serial - 共有 AWS **config**ファイル設定

ロールを引き受けるときに使用する必要がある多要素認証 (MFA) デバイスの ID もしくはシリアル番号を指定します。

ロールの信頼ポリシーに MFA 認証を必要とする条件が含まれているロールを引き受ける場合に必要です。

デフォルト値： なし。

有効な値： 値には、ハードウェアデバイスのシリアルナンバー (GAHT12345678など) または仮想 MFA デバイス (など) の Amazon リソースネーム (ARN) のいずれか指定できます。MFA の詳細については、「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。

config ファイルにこれを設定する例を以下に示します。

```
mfa_serial = arn:aws:iam::123456789012:mfa/my-user-name
```

role_arn - 共有 AWS **config**ファイル設定, **AWS_IAM_ROLE_ARN** - 環境変数, **aws.roleArn** - JVM システムプロパティ: Java/Kotlin のみ

このプロファイルを使用してリクエストされた操作の実行に使用する IAM ロールの Amazon リソースネーム (ARN) を指定します。

デフォルト値： なし。

有効な値： 値は、以下の arn:aws:iam::*account-id*:role/*role-name* 形式の IAM ロールの ARN である必要があります。

さらに、以下の設定のいずれかを指定する必要があります。

- `source_profile` — そのプロファイルでその役割を引き受ける権限を持つ認証情報を検索するために使用する別のプロファイルを識別する。
- `credential_source` — 現在の環境変数で識別される認証情報、または Amazon EC2 インスタンスプロファイルに添付されている認証情報、または Amazon ECS コンテナインスタンスを使用する。
- `web_identity_token_file` — モバイルまたはウェブアプリケーションで認証されたユーザーにパブリックOpenID Connect (OIDC) 互換の ID プロバイダーを使用する。

`role_session_name` - 共有 AWS **`config`**ファイル設定, **`AWS_IAM_ROLE_SESSION_NAME`** - 環境変数, **`aws.roleSessionName`** - JVM システムプロパティ: Java/Kotlin のみ

ロールセッションにアタッチする名前を指定します。この名前は、このセッションに関連付けられたエントリの AWS CloudTrail ログに表示されます。

デフォルト値：オプションパラメータ。この値を指定しない場合、セッション名は自動的に生成されます。

有効な値： AWS CLI または AWS API がユーザーに代わって `AssumeRole` オペレーション (または オペレーションなどの `AssumeRoleWithWebIdentity` オペレーション) を呼び出すときに `RoleSessionName`パラメータに提供されます。値は、クエリできる引き受けたロールユーザーの Amazon リソースネーム (ARN) の一部になり、このプロファイルによって呼び出されるオペレーションの CloudTrail ログエントリの一部として表示されます。

`arn:aws:sts::`*`123456789012`*`:assumed-role/`*`my-role-name`*`/`***`my-role_session_name`***.

`config` ファイルにこれを設定する例を以下に示します。

```
role_session_name = my-role-session-name
```

`source_profile` - 共有 AWS **`config`**ファイル設定

元のプロファイル内の `role_arn` 設定で指定されたロールを継承するために使用される認証情報の別のプロファイルを指定します。共有 AWS `config` ファイルと `credentials` ファイルでプロファイルがどのように使用されるかについては、「」を参照してください共有 `config` および `credentials` ファイル。

ロール引き受けプロファイルでもあるプロファイルを指定すると、認証情報が完全に解決されるように、各ロールが順番に引き継がれます。SDK が認証情報を含むプロファイルに遭遇すると、このチェーンは停止します。ロールチェーンは、 AWS CLI または AWS API ロールセッションを

最大 1 時間に制限し、引き上げることはできません。詳細については、「IAM ユーザーガイド」の「ロールの主な用語と概念」を参照してください。

デフォルト値： NONE。

有効な値： `config` および `credentials` ファイルで定義されているプロファイルの名前で構成されるテキスト文字列。現在のプロファイルの `role_arn` に対する値も指定する必要があります。

`credential_source` と `source_profile` の両方を同じプロファイルで指定することはできません。

設定ファイルでこれを設定する例:

```
[profile A]
source_profile = B
role_arn =  arn:aws:iam::123456789012:role/RoleA

[profile B]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGS
```

前の例では、A プロファイルは B プロファイル内の認証情報を使用しています。SDK またはツールが A プロファイルを使用するように指定すると、SDK またはツールはリンクされた B のプロファイルの認証情報を自動的に検索し、それらを使用して、指定された IAM ロールの一時的な認証情報をリクエストします。SDK またはツールは、バックグラウンドで sts:AssumeRole オペレーションを使用してこれを実現します。これらの一時的な認証情報は、 AWS リソースにアクセスするためにコードによって使用されます。指定されたロールには、 コマンドや API メソッドなど、リクエストされたコードの実行を許可する AWS のサービス IAM アクセス許可ポリシーがアタッチされている必要があります。

`web_identity_token_file` - 共有 AWS **`config`**ファイル設定, **`AWS_WEB_IDENTITY_TOKEN_FILE`** - 環境変数, **`aws.webIdentityTokenFile`** - JVM システムプロパティ: Java/Kotlin のみ

対応する OAuth 2.0 プロバイダー または、OpenID Connect ID 識別情報プロバイダー からのアクセストークンを含むファイルへのパスを指定します。

この設定により、「Google」 、「Facebook」 、「Amazon」 などの多数のウェブ ID フェデレーションプロバイダーを使用して認証を行うことができます。SDK または開発者ツールはこ

のファイルの内容をロードし、ユーザーに代わって `AssumeRoleWithWebIdentity` オペレーションを呼び出すときに `WebIdentityToken` 引数として渡します。

デフォルト値： NONE。

有効な値： この値はパスとファイル名でなければなりません。ファイルには、認識情報プロバイダーから提供される、OAuth 2.0 アクセストークンまたは OpenID Connect ID トークンが含まれていなければなりません。相対パスは、プロセスの作業ディレクトリを基準にした相対パスとして扱われます。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin のみでサポートされています。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	部分的	`credential_source` はサポートされていません。`duration_seconds` はサポートされていません。`mfa_serial` はサポートされていません。
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	はい	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	部分的	`mfa_serial` はサポートされていません。`AWS_ROLE_ARN` の代わりに を使用します`AWS_IAM_ROLE_ARN` 。`AWS_ROLE_SESSION_NAME` の代わりに を使用します`AWS_IAM_ROLE_SESSION_NAME` 。

SDK	サポート	注意または詳細情報
SDK for Java 1.x	部分的	`mfa_serial` はサポートされていません。JVM システムプロパティはサポートされていません。
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	部分的	`credential_source` はサポートされていません。
SDK for Kotlin	はい	`AWS_ROLE_ARN` の代わりに を使用します`AWS_IAM_ROLE_ARN` 。`AWS_ROLE_SESSION_NAME` の代わりに を使用します`AWS_IAM_ROLE_SESSION_NAME` 。
SDK for .NET 3.x	はい	
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	はい	
のツール PowerShell	はい	

コンテナ認証情報プロバイダー

コンテナ認証情報プロバイダーは、お客様のコンテナ化されたアプリケーションの認証情報を取得します。この認証情報プロバイダーは、Amazon Elastic Container Service (Amazon ECS) および Amazon Elastic Kubernetes Service (Amazon EKS) をご利用のお客様に役立ちます。SDK は GET リクエストを通じて指定された HTTP エンドポイントから認証情報をロードします。

Amazon ECS を利用する場合は、認証情報の分離、認可、監査可能性を改善するために、タスク IAM ロールを使用することをお勧めします。Amazon ECS を設定すると、SDK とツールが認証情報を取得するために使用する `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 環境変数が設定されます。この機能用に Amazon ECS を設定するには、「Amazon Elastic Container Service デベロッパーガイド」の「タスク IAM ロール」を参照してください。

Amazon EKS を利用する場合は、認証情報の分離、最小特権、監査可能性、独立したオペレーション、再利用性、およびスケーラビリティを改善するために、Amazon EKS Pod Identity を利用することをお勧めします。ポッドと IAM ロールの両方は、アプリケーション用に認証情報を管理するために Kubernetes サービスアカウントに関連付けられています。Amazon EKS Pod Identity の詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS Pod Identities」を参照してください。Amazon EKS を設定すると、SDK とツールが認証情報を取得するために使用する `AWS_CONTAINER_CREDENTIALS_FULL_URI` および `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` 環境変数が設定されます。セットアップの詳細については、Amazon EKS ユーザーガイドの「Amazon EKS Pod Identity Agent のセットアップ」または AWS ブログウェブサイトの「Amazon EKS Pod Identity は Amazon EKS クラスター上のアプリケーションの IAM アクセス許可を簡素化します」を参照してください。

この機能を設定するには、以下のように使用します。

`AWS_CONTAINER_CREDENTIALS_FULL_URI` - 環境変数

SDK が認証情報をリクエストするときに使用するフル HTTP URL エンドポイントを指定します。これにはスキームとホストの両方が含まれます。

デフォルト値： なし。

有効な値：有効な URI。

注意：この設定は *`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`* の代替であり、*`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`* が設定されていない場合にのみ使用されます。

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_CONTAINER_CREDENTIALS_FULL_URI=http://localhost/get-credentials
```

または

```
export AWS_CONTAINER_CREDENTIALS_FULL_URI=http://localhost:8080/get-credentials
```

AWS_CONTAINER_CREDENTIALS_RELATIVE_URI - 環境変数

SDK が認証情報をリクエストするときに使用する相対 HTTP URL エンドポイントを指定します。値は、デフォルトの Amazon ECS ホスト名 `169.254.170.2` に付加されます。

デフォルト値: [なし]。

有効な値：有効な相対 URI。

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=/get-credentials?a=1
```

AWS_CONTAINER_AUTHORIZATION_TOKEN - 環境変数

認可トークンをプレーンテキストで指定します。この変数が設定されている場合、SDK は HTTP リクエストの認証ヘッダーに環境変数の値を設定します。

デフォルト値： なし。

有効な値： 文字列。

注意：この設定は *AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE* の代替であり、*AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE* が設定されていない場合にのみ使用されます。

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_CONTAINER_CREDENTIALS_FULL_URI=http://localhost/get-credential
export AWS_CONTAINER_AUTHORIZATION_TOKEN=Basic abcd
```

AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE - 環境変数

プレーンテキストの認可トークンを含むファイルへの絶対ファイルパスを指定します。

デフォルト値: [なし]。

有効な値： 文字列。

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_CONTAINER_CREDENTIALS_FULL_URI=http://localhost/get-credential
export AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE=/path/to/token
```

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	はい	
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	はい	
SDK for Java 2.x	はい	
SDK for Java 1.x	部分的	Amazon EKS Pod Identity および `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` はサポートされていません。
SDK for JavaScript 3.x	はい	

SDK	サポート	注意または詳細情報
SDK for JavaScript 2.x	はい	
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	はい	
のツール PowerShell	はい	

IAM Identity Center 認証情報プロバイダー

この認証メカニズムは AWS IAM Identity Center 、コードへのシングルサインオン (SSO) アクセスを取得するために使用されます。 AWS のサービス

> **Note**
>
> AWS SDK API ドキュメントでは、IAM ID センターの認証情報プロバイダーは SSO 認証情報プロバイダーと呼ばれています。

IAM Identity Center を有効にしたら、共有ファイル内の設定用のプロファイルを定義します。 AWS `config`このプロファイルは IAM Identity Center アクセスポータルへの接続に使用されます。ユーザーが IAM Identity Center で正常に認証されると、ポータルはそのユーザーに関連付けられた IAM ロールの短期認証情報を返します。SDK AWS のサービス が設定から一時的な認証情報を取得してリクエストに使用する方法については、を参照してください[IAM Identity Center 認証を理解する](#)。

`config` ファイルを使用して IAM Identity Center を設定するには 2 つの方法があります。

- SSO トークンプロバイダーの設定 (推奨) — セッション期間の延長。
- 更新不可のレガシー構成 — 固定 8 時間のセッションを使用します。

どちらの構成でも、セッショ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。

カスタムセッション期間を設定するには、SSO トークンプロバイダー設定を使用する必要があります。

次の 2 つのガイドには、IAM Identity Center に関する追加情報が含まれています。

- [AWS IAM Identity Center ユーザーガイド](#)
- [AWS IAM Identity Center ポータル API リファレンス](#)

前提条件

最初に IAM Identity Center を有効にしておく必要があります。IAM アイデンティティセンターの認証を有効にする方法の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[開始方法](#)」を参照してください。

または、「[IAM Identity Center 認証](#)」このガイドの指示に従ってください。これらの手順は、IAM アイデンティティセンターの有効化から、以下の必要な共有 `config` ファイル設定の完了まで、包括的なガイダンスとなります。

SSO トークンプロバイダー設定

> **Note**
>
> AWS CLI [を使用してこの構成を作成する方法については、の「`aws configure sso`ウィザードによるプロファイルの設定](#)」を参照してください AWS CLI。

SSO トークンプロバイダー設定を使用すると、 AWS SDK またはツールは延長されたセッション期間までセッションを自動的に更新します。セッション期間と最大時間について詳しくは、ユーザーガイドの「AWS アクセスポータルと IAM Identity Center 統合アプリケーションのセッション期間の設定」を参照してください。AWS IAM Identity Center

`sso-sessionconfig`ファイルのセクションは、SSO アクセストークンを取得するための設定変数をグループ化するために使用され、それを使用して認証情報を取得できます。 AWS `config` ファイル内のセクションのフォーマットの詳細については、設定ファイルの形式を参照してください。

`sso-session` セクションを定義してプロファイルに関連付けます。`sso_region` と `sso_start_url` は `sso-session` セクション内に設定する必要があります。通常は`sso_account_id`、SDK `sso_role_name` `profile` AWS が認証情報をリクエストできるようにセクションで設定する必要があります。

> **Note**
>
> SDK とツールがこの構成を使用して認証情報を使用および更新する方法の詳細については、「IAM Identity Center 認証を理解する」を参照してください。

次の例では、IAM Identity Center 認証情報をリクエストするように SDK を設定します。トークンの自動更新もサポートしています。

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

`sso-session` 構成は複数のプロファイルで再利用できます。

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
```

```
[profile prod]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole2

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

sso_account_id と sso_role_name は SSO トークン設定のすべてのシナリオで必須というわけではありません。 AWS のサービス アプリケーションがベアラ認証をサポートする認証のみを使用している場合は、 AWS 従来の認証情報は必要ありません。ベアラー認証は、ベアラートークンと呼ばれるセキュリティトークンを使用する HTTP 認証スキームです。このシナリオでは、sso_account_id と sso_role_name は必須ではありません。ベアラートークン認証をサポートしているかどうかは、個別のガイドを参照してください。 AWS のサービス

登録スコープは sso-session の一部として設定されます。スコープは、ユーザーのアカウントに対するアプリケーションのアクセスを制限する OAuth 2.0 のメカニズムです。アプリケーションは 1 つ以上のスコープをリクエストでき、アプリケーションに発行されたアクセストークンは付与されたスコープに限定されます。これらのスコープは、登録された OIDC クライアントがリクエストできるアクセス許可と、クライアントが取得するアクセストークンを定義します。サポートされているアクセススコープのオプションについては、「AWS IAM Identity Center ユーザーガイド」の「アクセススコープ」 を参照してください。次の例では、アカウントとロールを一覧表示するアクセスを許可するように sso_registration_scopes を設定しています。

```
[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

認証トークンは、セッション名に基づいたファイル名を使用して、~/.aws/sso/cache ディレクトリの下のディスクにキャッシュされます。

更新不可のレガシー設定

トークンの自動更新は、更新不可のレガシー設定ではサポートされていません。代わりに、SSO トークンプロバイダー設定 の使用をお勧めします。

更新不可のレガシー設定を使用するには、プロファイル内で次の設定を指定する必要があります。

- sso_start_url
- sso_region
- sso_account_id
- sso_role_name

プロファイルのユーザーポータルは、 sso_start_url および sso_region 設定を使用して指定します。アクセス許可は sso_account_id および sso_role_name 設定で指定します。

次の例では、 config ファイルに 4 つの必要となる値を設定します。

```
[profile my-sso-profile]
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_region = us-west-2
sso_account_id = 111122223333
sso_role_name = SSOReadOnlyRole
```

認証トークンは、sso_start_url に基づいたファイル名を使用して、~/.aws/sso/cache ディレクトリの下のディスクにキャッシュされます。

IAM Identity Center 認証情報プロバイダーの設定

この機能は以下を使用して設定します。

sso_start_url-共有ファイル設定 AWS **config**

組織の IAM Identity Center アクセスポータルを指す URL。IAM Identity Center アクセスポータルの詳細については、AWS IAM Identity Center ユーザーガイドの「 AWS アクセスポータルの使用」を参照してください。

この値を確認するには、 IAM Identity Center コンソール を開き、 [ダッシュボード]を表示して、AWS [アクセスポータル URL]を検索します。

sso_region- AWS **config** 共有ファイル設定

IAM ID センターポータルホスト、つまり IAM ID センターを有効にする前に選択したリージョンを格納するもの。 AWS リージョン AWS これはデフォルトのリージョンとは独立しており、異なる場合もあります。

AWS リージョン とそのコードの詳細なリストについては、の「リージョナルエンドポイント」を参照してください。Amazon Web Services 全般のリファレンスこの値を確認するには、IAM Identity Centerコンソールを開いて[ダッシュボード]を表示し、[リージョン]を探します。

`sso_account_id`- AWS **`config`** 共有ファイル設定

AWS アカウント AWS Organizations 認証に使用するためにサービスを通じて追加されたの数値 ID。

使用可能なアカウントのリストを確認するには、IAM Identity Center コンソールに移動してAWS アカウントページを開きます。AWS IAM Identity Center ポータル API リファレンスの ListAccountsAPI メソッドを使用して、使用可能なアカウントのリストを確認することもできます。たとえば、list-accounts AWS CLI メソッドを呼び出すことができます。

`sso_role_name`-共有ファイル設定 AWS **`config`**

ユーザーのアクセス許可を定義するIAM ロールとしてプロビジョニングされたアクセス許可セットの名前。 AWS アカウント ロールは指定された by に存在する必要がありますsso_account_id。ロールの Amazon リソースネーム (ARN) ではなく、ロール名を使用してください。

アクセス許可セットには IAM ポリシーとカスタムアクセス許可ポリシーがアタッチされており、ユーザーに割り当てられた AWS アカウントに付与されるアクセスレベルを定義します。

使用可能な権限セットのリストを確認するには AWS アカウント、IAM Identity Center AWS アカウントコンソールに移動してページを開きます。 AWS アカウント 表に表示されている正しい権限セット名を選択してください。AWS IAM Identity Center ポータル API リファレンスの ListAccountRolesAPI メソッドを使用して、使用可能な権限セットのリストを確認することもできます。たとえば、 AWS CLI メソッドを呼び出すことができますlist-account-roles。

`sso_registration_scopes`- AWS **`config`** 共有ファイル設定

sso-session に許可するスコープのカンマ区切りのリストです。スコープは、IAM Identity Center ベアラートークンで承認されたエンドポイントへのアクセスを許可します。IAM Identity Center サービスから更新トークンを取得するには、sso:account:access の最低限のスコープを付与する必要があります。サポートされているアクセススコープの文字列については、「AWS IAM Identity Center ユーザーガイド」の「アクセススコープ」を参照してください。この設定は、更新できない従来の設定には適用されません。レガシー構成を使用して発行されたトークンは、暗黙的にsso:account:access スコープに制限されます。

AWS SDK との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	Yes	設定値は`credentials` ファイルでもサポートされています。
SDK for Java 1.x	No	
3.x JavaScript 用の SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	部分的	更新不可のレガシー設定のみ。
以下のためのツール PowerShell	Yes	

IMDS 認証情報プロバイダー

インスタンスメタデータサービス (IMDS) は、インスタンスに関するデータで、実行中のインスタンスを設定または管理するために使用します。利用可能なデータの詳細については、Amazon EC2 ユーザーガイド」の「インスタンスメタデータとユーザーデータ」または「Amazon EC2 ユーザーガイド」の「インスタンスメタデータとユーザーデータ」を参照してください。 Amazon EC2 Amazon EC2 では、インスタンスにさまざまな情報を提供できるローカルエンドポイントをインスタンスで利用可能です。インスタンスにロールがアタッチされている場合は、そのロールに有効な認証情報のセットが使用できます。SDK はそのエンドポイントを使用して、デフォルトの認証情報プロバイダーチェーンの一部として認証情報を解決できます。インスタンスメタデータサービスバージョン 2 (IMDSv2) は、IMDS のより安全なバージョンであり、デフォルトで使用されます。再試行できない条件 (HTTP エラーコード 403、404、405) が原因で失敗した場合は、IMDSv1 がフォールバックとして使用されます。

この機能を設定するには、以下のように使用します。

`AWS_EC2_METADATA_DISABLED` - 環境変数

認証情報の取得に Amazon EC2 インスタンスメタデータサービス (IMDS) を使用するかどうか。

デフォルト値: `false`。

有効値:

- **`true`** – 認証情報を取得するために IMDS を使用しません。
- **`false`** – 認証情報を取得するために IMDS を使用します。

`ec2_metadata_v1_disabled` - 共有 AWS **`config`**ファイル設定, **`AWS_EC2_METADATA_V1_DISABLED`** - 環境変数, **`aws.disableEc2MetadataV1`** - JVM システムプロパティ: Java/Kotlin のみ

IMDSv2 が失敗した場合に、Instance Metadata Service Version 1 (IMDSv1) をフォールバックとして使用するかどうか。

> **Note**
>
> 新しい SDK は IMDSv1 をサポートしていないため、この設定はサポートされていません。詳細については、表 AWS SDKsとの互換性 を参照してください。

デフォルト値: `false`。

有効値:

- **true** – IMDSv1 をフォールバックとして使用しません。
- **false** – IMDSv1 をフォールバックとして使用します。

ec2_metadata_service_endpoint - 共有 AWS **config**ファイル設定, **AWS_EC2_METADATA_SERVICE_ENDPOINT** - 環境変数, **aws.ec2MetadataServiceEndpoint** - JVM システムプロパティ: Java/Kotlin のみ

IMDS のエンドポイント。

デフォルト値： `ec2_metadata_service_endpoint_mode` が `IPv4` に等しい場合、デフォルトエンドポイントは `http://169.254.169.254` です。`ec2_metadata_service_endpoint_mode` が `IPv6` に等しい場合、デフォルトのエンドポイントは `http://[fd00:ec2::254]` です。

有効な値：有効な URI。

ec2_metadata_service_endpoint_mode - 共有 AWS **config**ファイル設定, **AWS_EC2_METADATA_SERVICE_ENDPOINT_MODE** - 環境変数, **aws.ec2MetadataServiceEndpointMode** - JVM システムプロパティ: Java/Kotlin のみ

IMDS のエンドポイントモード。

デフォルト値：`IPv4`。

有効な値： `IPv4`、`IPv6`。

> **Note**
>
> IMDS 認証情報プロバイダーは 認証情報プロバイダーチェーン の一部です。ただし、IMDS 認証情報プロバイダーは、この一連で他のいくつかのプロバイダーの後にのみチェックされます。そのため、プログラムでこのプロバイダーの認証情報を使用する場合は、設定から他の有効な認証情報プロバイダーを削除するか、別のプロファイルを使用する必要があります。あるいは、認証情報プロバイダチェーンに頼ってどのプロバイダが有効な認証情報を返すかを自動的に検出する代わりに、コード内で IMDS 認証情報プロバイダの使用を指定してください。サービスクライアントを作成するときに、認証情報ソースを直接指定できます。

IMDS 認証情報のセキュリティ

デフォルトでは、 AWS SDK に有効な認証情報が設定されていない場合、SDK は Amazon EC2 インスタンスメタデータサービス (IMDS) を使用して AWS ロールの認証情報を取得しようとします。この動作は、`AWS_EC2_METADATA_DISABLED` 環境変数を `true` に設定することで無効にできます。これにより、不必要なネットワークアクティビティが防止され、Amazon EC2 インスタンスメタデータサービスが偽装される可能性がある信頼できないネットワークのセキュリティが強化されます。

> **Note**
>
> AWS 有効な認証情報で設定された SDK クライアントは、これらの設定に関係なく、IMDS を使用して認証情報を取得しません。

Amazon EC2 IMDS 認証情報の使用の無効化

この環境変数の設定方法は、使用中のオペレーティングシステムと、変更を持続的にしたいかどうかによって異なります。

Linux および macOS

Linux または macOS を使用しているお客様は、次のコマンドを使用して、この環境変数を設定できます。

```
$ export AWS_EC2_METADATA_DISABLED=true
```

この設定を複数のシェルセッションやシステムの再起動後も維持したい場合は、`.bash_profile`、`.zsh_profile`、`.profile` などの上記のコマンドをシェルプロファイルファイルに追加できます。

Windows

Windows を使用しているお客様は、次のコマンドを使用して、この環境変数を設定できます。

```
$ set AWS_EC2_METADATA_DISABLED=true
```

この設定を複数のシェルセッションやシステムの再起動後も維持したい場合は、代わりに以下のコマンドを使用できます。

```
$ setx AWS_EC2_METADATA_DISABLED=true
```

> **Note**
>
> setx コマンドは現在のシェルセッションに値を適用しないため、変更を有効にするにはシェルをリロードするか再度開く必要があります。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK for C++	あり	
SDK for Go V2 (1.x)	あり	
SDK for Go 1.x (V1)	あり	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	あり	
SDK for Java 1.x	部分的	JVM システムプロパティ: `com.amazonaws.sdk.disableEc2MetadataV1` の代わりに `aws.ec2MetadataServiceEndpoint` を使用し`aws.disab`

SDK	サポート	注意または詳細情報
		`leEc2MetadataV1` 、`aws.ec2MetadataServiceEndpointMode` サポートされていません。
SDK for JavaScript 3.x	あり	
SDK for JavaScript 2.x	あり	
SDK for Kotlin	あり	IMDSv1 フォールバックを使用しません。
SDK for .NET 3.x	あり	
SDK for PHP 3.x	あり	
SDK for Python (Boto3)	あり	
SDK for Ruby 3.x	あり	
SDK for Rust	あり	IMDSv1 フォールバックを使用しません。
のツール PowerShell	あり	を使用して、コードで IMDSv1 フォールバックを明示的に無効にできます`[Amazon.Util.EC2InstanceMetadata]::EC2MetadataV1Disabled = $true` 。

プロセス認証情報プロバイダー

SDK には、カスタムユースケースに合わせて認証情報プロバイダーチェーンを拡張する方法の機能があります。

IAM Roles Anywhere は、外部で実行されるワークロードまたはプロセスの一時的な認証情報を取得する方法を提供します。 AWSこの用途に合わせて `credential_process` を設定するには、 IAM Roles Anywhere を参照してください。

> **Warning**
>
> 以下では、外部プロセスから認証情報を取得する方法について説明します。これは危険な場合があるため、注意して進めてください。可能であれば、他の認証情報プロバイダーの利用をお勧めします。このオプションを使用する場合は、使用しているオペレーティングシステムのセキュリティ上のベストプラクティスに従って、`config` ファイルができるだけロックされていることを確認する必要があります。カスタム認証情報ツールが機密情報を `StdErr` に書き込まないよう確認してください。SDK および AWS CLI がそのような情報をキャプチャしてログに記録し、不正ユーザーに情報を公開する可能性があるためです。

この機能を設定するには、以下のように使用します。

`credential_process`- AWS **`config`** 共有ファイル設定

使用する認証情報を生成あるいは取得するために ユーザーに代わって SDK またはツールが実行する外部のコマンドを指定します。この設定では、SDK が呼び出すプログラム/コマンドの名前を指定します。SDK がプロセスを呼び出すと、プロセスが JSON データを `stdout` に書き込むのを待ちます。カスタムプロバイダーは、特定の形式で情報を返す必要があります。この情報には、 SDK またはツールがユーザーを認証するために使用できる認証情報が含まれています。

> **Note**
>
> プロセス認証情報プロバイダーは 認証情報プロバイダーチェーン の一部です。ただし、プロセス認証情報プロバイダーは、このシリーズの他のいくつかのプロバイダーの後にのみチェックされます。そのため、プログラムでこのプロバイダーの認証情報を使用する場合は、設定から他の有効な認証情報プロバイダーを削除するか、別のプロファイルを使用する必要があります。あるいは、認証情報プロバイダーチェーンに頼ってどのプロバイダーが有効な認証情報を返すかを自動的に検出する代わりに、プロセス認証情報プロバイダーの使用をコードで指定してください。サービスクライアントを作成するときに、認証情報ソースを直接指定できます。

認証情報プログラムへのパスの指定

設定の値は、SDK または開発ツールがユーザーに代わって実行するプログラムへのパスを含む文字列です。

- パスとファイル名には、A ~ Z、a ~ z、0 ~ 9、ハイフン (-)、アンダースコア (_)、ピリオド (.)、フォワードスラッシュ (/)、バックスラッシュ (\)、スペースのみを使用できます。
- パスまたはファイル名にスペースが含まれている場合は、完全なパスとファイル名を二重引用符 (" ") で囲みます。
- パラメータ名またはパラメータ値にスペースが含まれている場合は、その要素を二重引用符 (" ") で囲みます。囲むのは、名前または値のみであり、そのペアではありません。
- 文字列に環境変数を含めないでください。例えば、`$HOME` または `%USERPROFILE%` を含めることはできません。
- ホームフォルダを ~ として指定しないでください。* フルパスまたはベースファイル名を指定する必要があります。ベースファイル名がある場合、システムは `PATH` 環境変数で指定されたフォルダー内でプログラムを検索しようとします。

  次の例は、Linux/macOS上の `config` 共有ファイルに credential_process を設定する方法を示しています。

  ```
  credential_process = "/path/to/credentials.sh" parameterWithoutSpaces "parameter with
   spaces"
  ```

  次の例は、Windows 上の `config` 共有ファイルに credential_process を設定する方法を示しています。

  ```
  credential_process = "C:\Path\To\credentials.cmd" parameterWithoutSpaces "parameter
   with spaces"
  ```

認証情報プログラムからの有効な出力

SDK はプロファイルで指定されたようにコマンドを実行し、次に標準出力からデータを読み取ります。スクリプトであるかバイナリープログラムであるかに関わらず、指定するコマンドは、以下の構文と一致する JSON 出力を `STDOUT` に生成する必要があります。

```
{
    "Version": 1,
```

```
    "AccessKeyId": "an AWS access key",
    "SecretAccessKey": "your AWS secret access key",
    "SessionToken": "the AWS session token for temporary credentials",
    "Expiration": "RFC3339 timestamp for when the credentials expire"
}
```

> **Note**
>
> 本文書の執筆時点では、`Version` キーは 1 に設定する必要があります。構造が進化するため、時間の経過と共に増えていく可能性があります。

`Expiration` キーは、RFC3339 形式のタイムスタンプです。`Expiration` キーがツールの出力にない場合、SDK はこの認証情報が更新されない長期の認証情報であると判断します。それ以外の認証情報は一時的な認証情報と見なされ、有効期限が切れる前に `credential_process` を再実行して自動的に更新されます。

> **Note**
>
> SDK は、外部プロセスの認証情報をロールを引き受けるような認証情報としてキャッシュしません。キャッシュが必要な場合は、外部プロセス内で実装する必要があります。

外部プロセスはゼロ以外のリターンコードを返して、認証情報の取得時にエラーが発生したことを示すことができます。

AWS SDK との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	Yes	
SDK for C++	Yes	

SDK	サポート	注意または詳細情報
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	
3.x 用 JavaScript SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
用ツール PowerShell	Yes	

AWS SDKs標準化された機能

多くの機能は、一貫したデフォルトに標準化されており、多くの SDK で同じように動作します。この一貫性により、複数の SDK 間のコーディングの生産性と明確性が向上します。すべての設定はコード内で上書きできます。詳細については、特定の SDK API を参照してください。

> **Important**
>
> すべての SDK がすべての機能をサポートしているわけではなく、機能内のすべての側面をサポートしているわけでもありません。

トピック

アプリケーション ID

単一の を複数のカスタマーアプリケーションで使用して、 を呼び出す AWS アカウント ことができます AWS のサービス。アプリケーション ID を使用すると、顧客は を使用して一連の呼び出しを行ったソースアプリケーションを特定できます AWS アカウント。 AWS SDKs と サービスは、顧客通信に表示するために 以外にこの値を使用または解釈しません。例えば、この値を運用 E メールやに含める AWS Health Dashboard ことで、通知に関連付けられているアプリケーションを一意に識別できます。

この機能を設定するには、以下のように使用します。

`sdk_ua_app_id` - 共有 AWS **`config`**ファイル設定, **`AWS_SDK_UA_APP_ID`** - 環境変数, **`aws.userAgentAppId`** - JVM システムプロパティ: Java/Kotlin のみ

この設定は、特定の 内のどのアプリケーションが を呼び AWS アカウント 出すかを識別するためにアプリケーションに割り当てる一意の文字列です AWS。

デフォルト値: None

有効な値： 最大長が 50 の文字列。文字、数字、および次の特殊文字を使用できます: `!`、`$%`、`&*`、`+`、`、`-.`、`,`、`^_`、`|`、`~`。

`config` ファイルにこの値を設定する例を以下に示します。

```
[default]
sdk_ua_app_id=ABCDEF
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_SDK_UA_APP_ID=ABCDEF
export AWS_SDK_UA_APP_ID="ABC DEF"
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_SDK_UA_APP_ID ABCDEF
setx AWS_SDK_UA_APP_ID="ABC DEF"
```

使用するシェルに特別な意味を持つ記号を含める場合は、必要に応じて値をエスケープします。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin のみでサポートされています。

SDK	サポート	注意または詳細情報
AWS CLI v2	いいえ	
SDK for C++	はい	共有 `config` ファイルはサポートされていません。
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	いいえ	
SDK for Java 2.x	部分的	共有`config`ファイル設定はサポートされていません。環境変数はサポートされていません。
SDK for Java 1.x	いいえ	
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	いいえ	
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	環境変数はサポートされていません。

SDK	サポート	注意または詳細情報
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	はい	
のツール PowerShell	いいえ	

Amazon EC2 インスタンスメタデータ

Amazon EC2 では、インスタンスメタデータサービス (IMDS) と呼ばれるサービスがインスタンスで使用できます。このサービスの詳細については、Amazon EC2 ユーザーガイド」の「インスタンスメタデータとユーザーデータ」または「Amazon EC2 ユーザーガイド」の「インスタンスメタデータとユーザーデータ」を参照してください。 Amazon EC2 IAM ロールで設定された Amazon EC2 インスタンスで認証情報の取得を試行すると、インスタンスメタデータサービスへの接続が調整可能になります。

この機能を設定するには、以下のように使用します。

`metadata_service_num_attempts` - 共有 AWS **`config`**ファイル設定,
`AWS_METADATA_SERVICE_NUM_ATTEMPTS` - 環境変数

この設定は、インスタンスメタデータサービスからデータの取得を試行するとき、停止するまでに試行する総回数を指定します。

デフォルト値： 1

有効な値：1 以上の数値。

metadata_service_timeout - 共有 AWS **config**ファイル設定, **AWS_METADATA_SERVICE_TIMEOUT** - 環境変数

インスタンスメタデータサービスからデータの取得を試行するときにタイムアウトするまでの秒数を指定。

デフォルト値： 1

有効な値：1 以上の数値。

config ファイルに次の値を設定する例を以下に示します。

```
[default]
metadata_service_num_attempts=10
metadata_service_timeout=10
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_METADATA_SERVICE_NUM_ATTEMPTS=10
export AWS_METADATA_SERVICE_TIMEOUT=10
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_METADATA_SERVICE_NUM_ATTEMPTS 10
setx AWS_METADATA_SERVICE_TIMEOUT 10
```

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	

SDK	サポート	注意または詳細情報
SDK for C++	なし	
SDK for Go V2 (1.x)	なし	
SDK for Go 1.x (V1)	なし	
SDK for Java 2.x	なし	
SDK for Java 1.x	部分的	`AWS_METADATA_SERVICE_TIMEOUT` のみサポートされています。
SDK for JavaScript 3.x	なし	
SDK for JavaScript 2.x	なし	
SDK for Kotlin	なし	
SDK for .NET 3.x	なし	
SDK for PHP 3.x	あり	
SDK for Python (Boto3)	あり	

SDK	サポート	注意または詳細情報
SDK for Ruby 3.x	なし	
SDK for Rust	なし	
のツール PowerShell	なし	

Amazon S3 アクセスポイント

Amazon S3 サービスでは、Amazon S3 バケットを操作する代替方法としてアクセスポイントが使用できます。アクセスポイントには、バケットに直接ではなく、一意のポリシーと設定を適用できます。 AWS SDKs では、バケット名を明示的に指定する代わりに、バケットフィールドのアクセスポイント Amazon リソースネーム (ARNsを API オペレーションに使用できます。アクセスポイント ARN と `GetObject` を使用してバケットからオブジェクトを取得したり、アクセスポイント ARN と `PutObject` を使用してバケットにオブジェクトを追加したりするなど、特定の操作に使用されます。

Amazon S3 Access Points と ARN の詳細については、「Amazon S3 ユーザーガイド」の「アクセスポイントの使用」を参照してください。

この機能を設定するには、以下のように使用します。

`s3_use_arn_region` - 共有 AWS **`config`**ファイル設定, **`AWS_S3_USE_ARN_REGION`** - 環境変数, **`aws.s3UseArnRegion`** - JVM システムプロパティ: Java/Kotlin のみ, コード内で値を直接設定するには、使用している SDK に直接問い合わせてください。

> この設定は、SDK がアクセスポイント ARN を使用してリクエストのリージョンエンドポイント AWS リージョン を構築するかどうかを制御します。SDK AWS リージョン は、ARN がクライアントの設定と同じ AWS パーティションによって処理されていることを検証 AWS リージョンし、ほとんどの場合失敗するパーティション間の呼び出しを防ぎます。複数定義した場合、コードで設定されたものが優先され、次に環境変数設定が続きます。
>
> デフォルト値: `false`

有効値:

- **`true`** – SDK は、クライアントが設定した ではなく、エンドポイントを構築する AWS リージョン ときに ARN の を使用します AWS リージョン。例外: クライアントの AWS リージョンが FIPS に設定されている場合は AWS リージョン、ARN の と一致する必要があります AWS リージョン。そうしないと、エラーが発生します。
- **`false`** – SDK は、エンドポイントを構築するときに、クライアントで設定された の代わりに ARN の AWS リージョン を使用します。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	はい	
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	はい	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	はい	
SDK for Java 1.x	はい	JVM システムプロパティはサポートされていません。

SDK	サポート	注意または詳細情報
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	はい	
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	標準の優先順位には従いません。共有 `config` ファイルの値が環境変数よりも優先されます。
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	いいえ	
のツール PowerShell	はい	標準の優先順位には従いません。共有 `config` ファイルの値が環境変数よりも優先されます。

Amazon S3 マルチリージョンアクセスポイント

Amazon S3 マルチリージョンアクセスポイントを使用すると、アプリケーションが複数の AWS リージョンにあるAmazon S3 バケットからのリクエストを実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで

使用されるのと同じアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。

マルチリージョンアクセスポイントの詳細については、「Amazon S3 ユーザーガイド」の「Amazon S3 のマルチリージョンアクセスポイント」を参照してください。

マルチリージョンアクセスポイントの Amazon リソースネーム (ARN) の機能の詳細については、「Amazon S3 ユーザーガイド」の「マルチリージョンアクセスポイントを使用したリクエスト」を参照してください。

マルチリージョンアクセスポイント作成の詳細については、「Amazon S3 ユーザーガイド」の「マルチリージョンアクセスポイントの管理」を参照してください。

SigV4A アルゴリズムは、グローバルリージョンリクエストの署名に使用される署名実装です。このアルゴリズムは、AWS Common Runtime (CRT) ライブラリ への依存関係を通じて SDK によって取得されます。

この機能を設定するには、以下のように使用します。

`s3_disable_multiregion_access_points`- AWS **`config`** 共有ファイル設定, **`AWS_S3_DISABLE_MULTIREGION_ACCESS_POINTS`** - 環境変数, **`aws.s3DisableMultiRegionAccessPoints`**-JVM システムプロパティ:Java/Kotlin のみ, コード内で値を直接設定するには、使用している SDK を直接調べてください。

この設定は、SDK がクロスリージョンリクエストを試みる可能性があるかどうかを制御します。複数定義した場合、コードで設定されたものが優先され、次に環境変数設定が続きます。

デフォルト値: `false`

有効値:

- **`true`** – クロスリージョンリクエストの使用を停止します。
- **`false`** – マルチリージョンアクセスポイントを使用したクロスリージョンリクエストを有効にします。

AWS SDK との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	Yes	
SDK for Java 1.x	No	
3.x 用 JavaScript SDK	Yes	
2.x JavaScript 用の SDK	No	
SDK for Kotlin	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
用ツール PowerShell	Yes	

AWS リージョン

AWS リージョン AWS のサービスを扱う際に理解しておくべき重要な概念です。

を使用すると AWS リージョン、 AWS のサービス 特定の地理的領域に物理的に存在する情報にアクセスできます。これは、ユーザーがアクセスする場所の近くでのデータとアプリケーションの実行を

維持するために有効です。リージョンでは耐障害性や安定性が提供され、レイテンシーを低減することもできます。これにより、リージョンの障害の影響を受けずに利用できる冗長リソースを作成できます。

AWS のサービス ほとんどのリクエストは特定の地域に関するものです。あるリージョンで作成したリソースは、 AWS のサービスで提供されるレプリケーション機能を明示的に使用しないかぎり、他のリージョンに存在することはありません。たとえば、Amazon S3 と Amazon EC2 はクロスリージョンのレプリケーションをサポートしています。IAM などの一部のサービスには、リージョンリソースがありません。

AWS 全般のリファレンス には、以下の情報が含まれています。

- リージョンとエンドポイントの関係を理解し、既存のリージョンエンドポイントのリストを表示するには、「AWS サービスエンドポイント」 を参照してください。
- サポートされている各リージョンとエンドポイントの最新リストを確認するには AWS のサービス、「サービスエンドポイントとクォータ」を参照してください。

サービスクライアントの作成

SDK はプログラムからアクセスするために AWS のサービス、それぞれにクライアントクラス/オブジェクトを使用します。 AWS のサービスたとえば、アプリケーションが Amazon EC2 にアクセスする必要がある場合、アプリケーションはそのサービスとやり取りする Amazon EC2 クライアントオブジェクトを作成します。

クライアントにリージョンが明示的に指定されていない場合、クライアントはデフォルトで以下の `region` 設定で設定されたリージョンを使用します。ただし、クライアントのアクティブリージョンは個々のクライアントオブジェクトに明示的に設定できます。この方法でのリージョンの設定は、特定のサービスクライアントのグローバル設定よりも優先されます。代替リージョンは、クライアントのインスタンス化時に SDK に固有に指定されます (特定の SDK ガイドまたは SDK のコードベースを確認してください)。

この機能を設定するには、以下のように使用します。

`region` AWS **`config`**-共有ファイル設定, **`AWS_REGION`** - 環境変数, **`aws.region`**-JVM システムプロパティ:Java/Kotlin のみ

AWS リージョン リクエストに使用するデフォルトを指定します。 AWS このリージョンは、使用するリージョンが指定されていない SDK サービスリクエストに使用されます。

デフォルト値： NONE。この値は明示的に指定する必要があります。

有効値:

- 「AWS 全般リファレンス」の「AWS サービスエンドポイント」に記載されているように、選択したサービスで使用できるどのリージョンコードでも指定できます。たとえば、 AWS リージョン この値によってエンドポイントは米国東部 (バージニア北部) `us-east-1` に設定されます。
- `aws-global()` や Amazon Simple Storage Service AWS Security Token Service (Amazon S3 AWS STS) などのリージョナルエンドポイントに加えて、個別のグローバルエンドポイントをサポートするサービスのグローバルエンドポイントを指定します。

`config` ファイルにこの値を設定する例を以下に示します。

```
[default]
region = us-west-2
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_REGION=us-west-2
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_REGION us-west-2
```

ほとんどの SDK には、アプリケーションコード内からデフォルトリージョンを設定するための「設定」オブジェクトがあります。詳細については、特定の AWS SDK 開発者ガイドを参照してください。

AWS SDK との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	Yes	AWS CLI v2 では、in `AWS_REGION` の値の前に in の任意の値を使用します `AWS_DEFAULT_REGION` (両方の変数がチェックされます)。
AWS CLI v1	Yes	AWS CLI v1 では、`AWS_DEFAULT_REGION` この目的のために名付けられた環境変数を使用します。
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	
3.x JavaScript 用の SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	この SDK は、この目的のために `AWS_DEFAULT_REGION` と名付けられた環境変数を使用します。
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	

SDK	サポート	注意または詳細情報
以下のためのツール PowerShell	Yes	

AWS STS 地域化されたエンドポイント

デフォルトでは、 AWS Security Token Service (AWS STS) はグローバルサービスとして使用でき、AWS STS すべてのリクエストはの 1 `https://sts.amazonaws.com` つのエンドポイントに送信されます。グローバルリクエストは米国東部 (バージニア北部) リージョンにマップされます。AWS AWS STS グローバルエンドポイントの代わりにリージョナルエンドポイントを使用することを推奨します。 AWS STS エンドポイントの詳細については、API リファレンスの「エンドポイント」AWS Security Token Service を参照してください。

この機能を設定するには、以下のように使用します。

`sts_regional_endpoints`-共有ファイル設定 AWS **`config`**, **`AWS_STS_REGIONAL_ENDPOINTS`** - 環境変数

この設定では、SDK またはツールが AWS Security Token Service (AWS STS) AWS のサービスとの通信に使用するエンドポイントを決定する方法を指定します。

デフォルト値: `legacy`

> **Note**
>
> 2022 年 7 月以降にリリースされるすべての SDK メジャーバージョンは、デフォルトで `regional` に設定されます。新しい SDK メジャーバージョンでは、この設定が削除され、`regional` 動作が使用する可能性があります。この変更による将来的な影響を減らすため、可能な場合はアプリケーションで `regional` の使用を開始することをお勧めします。

有効な値： (推奨値：*`regional`*)

- **`legacy`**—`,,,,,sts.amazonaws.com,,,,ap-northeast-1,,,,ap-south-1,ap-southeast-1,ap-southeast-2,aws-global,ca-central-1,eu-central-1,eu-`

north-1, eu-west-1 eu-west-2 eu-west-3 sa-east-1 us-east-1 us-east-2us-west-1, AWS STS AWS およびの各リージョンのグローバルエンドポイントを使用しますus-west-2。他のすべてのリージョンでは、それぞれのリージョンエンドポイントが自動的に使用されます。

- **regional**— SDK またはツールは常に、 AWS STS 現在設定されているリージョンのエンドポイントを使用します。たとえば、クライアントが使用するように設定されている場合us-west-2、 AWS STS sts.us-west-2.amazonaws.comsts.amazonaws.comへの呼び出しはすべてグローバルエンドポイントではなくリージョナルエンドポイントに対して行われます。この設定が有効なときにグローバルエンドポイントにリクエストを送信するには、リージョンを aws-global に設定します。

config ファイルに次の値を設定する例を以下に示します。

```
[default]
sts_regional_endpoints = regional
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_STS_REGIONAL_ENDPOINTS=regional
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_STS_REGIONAL_ENDPOINTS regional
```

AWS SDK との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	部分的	デフォルト値は regional です。

SDK	サポート	注意または詳細情報
SDK for C++	部分的	環境変数と `config` ファイル設定はサポートされていません。SDK は `regional` 設定で実行します。
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	
3.x JavaScript 用の SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	No	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
以下のためのツール PowerShell	Yes	

デュアルスタックと FIPS エンドポイント

この機能を設定するには、以下のように使用します。

use_dualstack_endpoint- AWS **config** 共有ファイル設定, **AWS_USE_DUALSTACK_ENDPOINT** - 環境変数, **aws.useDualstackEndpoint**-JVM システムプロパティ:Java/Kotlin のみ

SDK がデュアルスタックのエンドポイントにリクエストを送信するかどうかをオンまたはオフにします。IPv4 と IPv6 の両方のトラフィックをサポートするデュアルスタックエンドポイントの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 デュアルスタックエンドポイントの使用」を参照してください。デュアルスタックのエンドポイントは、一部のリージョンでは一部のサービスで利用できます。

デフォルト値: `false`

有効値:

- **true** – SDK またはツールは、デュアルスタックエンドポイントを使用してネットワークリクエストを行おうと試みます。サービスや AWS リージョンにデュアルスタックエンドポイントが存在しない場合、リクエストは失敗します。
- **false** – SDK またはツールは、ネットワークリクエストを行うためにデュアルスタックエンドポイントを使用しません。

use_fips_endpoint AWS **config**-共有ファイル設定, **AWS_USE_FIPS_ENDPOINT** - 環境変数, **aws.useFipsEndpoint**-JVM システムプロパティ:Java/Kotlin のみ

SDK またはツールが FIPS 準拠のエンドポイントにリクエストを送信するかどうかをオンまたはオフにします。連邦情報処理標準 (FIPS) は、データとその暗号化に関する米国政府のセキュリティ要件をまとめたものです。政府機関、パートナー、および連邦政府との取引を希望する者は、FIPS ガイドラインを遵守する必要があります。 AWS 標準のエンドポイントとは異なり、FIPS エンドポイントは FIPS 140-2 に準拠する TLS ソフトウェアライブラリを使用します。この設定が有効になっていて、そのサービスに FIPS エンドポイントが存在しない場合、呼び出しは失敗する可能性があります。 AWS リージョン AWS サービス固有のエンドポイント`--endpoint-url` AWS Command Line Interface この設定をオーバーライドするオプションもあります。

FIPS エンドポイントを指定する他の方法について詳しくは AWS リージョン、「サービス別の FIPS エンドポイント」を参照してください。Amazon Elastic Compute Cloud サービスのエンドポイントの詳細については、「Amazon EC2 API リファレンス」の「デュアルスタック (IPv4 と IPv6) エンドポイント」を参照してください。

デフォルト値: `false`

有効値:

- **true** – SDK またはツールが FIPS 準拠のエンドポイントにリクエストを送信します。
- **false** – SDK またはツールが FIPS 準拠のエンドポイントにリクエストを送信しません。

SDK との互換性 AWS

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	Yes	
SDK for Java 1.x	No	
3.x JavaScript 用の SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	

SDK	サポート	注意または詳細情報
SDK for Rust	Yes	
以下のためのツール PowerShell	Yes	

エンドポイント検出

SDKsは、エンドポイント検出を使用してサービスエンドポイント (さまざまなリソースにアクセスするための URLs) にアクセスしますが、 が必要に応じて URLsを変更 AWS するための柔軟性は維持されます。これにより、コードは新しいエンドポイントを自動的に検出できます。一部のサービスには固定エンドポイントはありません。代わりに、最初にエンドポイントを取得するようにリクエストすることで、ランタイムに利用可能なエンドポイントを取得します。使用可能なエンドポイントを取得したら、コードはそのエンドポイントを使用して他の操作にアクセスします。たとえば、Amazon Timestream の場合、SDK は利用可能なエンドポイントを取得する `DescribeEndpoints` リクエストを行い、それらのエンドポイントを使用して `CreateDatabase` や `CreateTable` などの特定の操作を実行します。

この機能を設定するには、以下のように使用します。

`endpoint_discovery_enabled` - 共有 AWS **`config`**ファイル設定, **`AWS_ENABLE_ENDPOINT_DISCOVERY`** - 環境変数, **`aws.endpointDiscoveryEnabled`** - JVM システムプロパティ: Java/Kotlin のみ, コード内で値を直接設定するには、使用している SDK に直接問い合わせてください。

DynamoDB のエンドポイント検出を有効または無効にします。

Timestream ではエンドポイント検出が必要で、Amazon DynamoDB ではオプションです。この設定は、サービスがエンドポイント検出を必要とするかどうか`false`に応じて、デフォルトで `true`または のいずれかになります。Timestream リクエストのデフォルトは `true`、Amazon DynamoDB リクエストのデフォルトは ですfalse。

有効値:

- **`true`** – エンドポイント検出がオプションであるサービスの場合、SDK はエンドポイントを自動的に検出しようとする必要があります。

- **`false`** – エンドポイント検出がオプションであるサービスの場合、SDK はエンドポイントを自動的に検出しようとする必要がありません。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin のみでサポートされています。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	はい	
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	はい	共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	はい	SDK for Java 2.x は、環境変数名`AWS_ENDPOINT_DISCOVERY_ENABLED` に を使用します。
SDK for Java 1.x	部分的	JVM システムプロパティはサポートされていません。
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	はい	

SDK	サポート	注意または詳細情報
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	部分的	Timestream でのみサポートされます。
のツール PowerShell	はい	

一般設定

SDK は SDK の全体的な動作を設定する一般設定の一部をサポートします。

この機能を設定するには、以下のように使用します。

`api_versions`- AWS **`config`** 共有ファイル設定

AWS 一部のサービスでは、下位互換性をサポートするために複数の API バージョンを管理しています。デフォルトでは、SDK と AWS CLI オペレーションは最新の API バージョンを使用します。リクエストに特定の API バージョンを使用することを要求するには、プロファイルに `api_versions` 設定を含めてください。

デフォルト値： なし。（SDK では最新の API バージョンが使用されます。）

有効な値:これはネストされた設定で、その後にそれぞれ 1 AWS つのサービスと使用する API バージョンを示すインデントされた行が 1 つ以上続きます。どの API バージョンが利用可能かについては、 AWS サービスのドキュメントを参照してください。

この例では、2 AWS つのサービスの特定の API `config` バージョンをファイルに設定しています。これらの API バージョンは、この設定を含むプロファイルで実行するコマンドにのみ使用されます。その他のサービスのコマンドは、そのサービスの API の最新バージョンを使用します。

```
api_versions =
    ec2 = 2015-03-01
    cloudfront = 2015-09-017
```

`ca_bundle`- AWS **`config`** 共有ファイル設定, **`AWS_CA_BUNDLE`** - 環境変数

SSL/TLS 接続を確立するときに使用するカスタム証明書バンドル (拡張子 `.pem` のファイル) へのパスを指定します。

デフォルト値： なし

有効な値： フルパスまたはベースファイル名を指定します。ベースファイル名がある場合、システムは PATH 環境変数で指定されたフォルダー内でプログラムを検索しようとします。

`config` ファイルにこの値を設定する例を以下に示します。

```
[default]
ca_bundle = dev/apps/ca-certs/cabundle-2019mar05.pem
```

Linux/macOS のコマンドラインによる環境変数の設定の例を以下に示します。

```
export AWS_CA_BUNDLE=/dev/apps/ca-certs/cabundle-2019mar05.pem
```

Windows のコマンドラインによる環境変数の設定の例を以下に示します。

```
setx AWS_CA_BUNDLE C:\dev\apps\ca-certs\cabundle-2019mar05.pem
```

`output`- AWS **`config`** 共有ファイル設定

AWS CLI およびその他の AWS SDK やツールでの結果のフォーマット方法を指定します。

デフォルト値: `json`

有効値:

- `json` - 出力は JSON 文字列としてフォーマットされます。
- **`yaml`** - 出力は YAML 文字列としてフォーマットされます。
- https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-output-format.html#yaml-stream-output - 出力はストリームされ、YAML 文字列としてフォーマットされます。ストリーミングにより、大きなデータタイプの処理を高速化できます。
- `text` - 出力は、複数行のタブ区切りの文字列値としてフォーマットされます。これは、`grep`、`sed`、または `awk` などのテキストプロセッサに出力を渡すのに役立ちます。
- `table` - 出力は、テーブルとしてフォーマットされ、文字の「+|-」を使用してセルの境界を形成します。通常、情報は他の形式よりも読みやすい「わかりやすい」形式で表示されますが、プログラムとしては役立ちません。

`parameter_validation`-共有ファイル設定 AWS **`config`**

AWS サービスエンドポイントに送信する前に、SDK またはツールがコマンドラインパラメータの検証を試行するかどうかを指定します。

デフォルト値: `true`

有効値:

- **`true`**-デフォルト。SDK またはツールは、コマンドラインパラメータのクライアント側検証を実行します。これにより、SDK またはツールはパラメーターが有効であることを確認し、エラーを検出できます。SDK またはツールは、 AWS サービスエンドポイントにリクエストを送信する前に、有効でないリクエストを拒否できます。
- **`false`**— SDK またはツールは、 AWS コマンドラインパラメータをサービスエンドポイントに送信する前に検証しません。 AWS サービスエンドポイントは、すべてのリクエストを検証し、有効でないリクエストを拒否する責任があります。

SDK との互換性 AWS

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	部分的	`api_versions` はサポートされていません。
SDK for C++	Yes	
SDK for Go V2 (1.x)	部分的	`api_versions` および `parameter_validation` はサポートされていません。
SDK for Go 1.x (V1)	部分的	`api_versions` および `parameter_validation` はサポートされていません。共有 `config` ファイル設定を使用するには、設定ファイルからの読み込みを有効にする必要があります。「セッション」を参照してください。
SDK for Java 2.x	No	
SDK for Java 1.x	No	
3.x JavaScript 用の SDK	Yes	
2.x JavaScript 用の SDK	Yes	
SDK for Kotlin	No	
SDK for .NET 3.x	No	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	No	

SDK	サポート	注意または詳細情報
以下のためのツール PowerShell	No	

IMDS クライアント

SDK は、セッション指向リクエストを使用してインスタンスメタデータサービスのバージョン 2 (IMDSv2) クライアントを実装します。IMDSv2 の詳細については、Amazon EC2 ユーザーガイド」のIMDSv22 を使用する」または「Amazon EC2 ユーザーガイド」のIMDSv22 を使用する」を参照してください。 Amazon EC2 IMDS クライアントは、SDK コードベースにあるクライアント設定オブジェクトを使用して設定できます。

この機能を設定するには、以下のように使用します。

`retries` - クライアント設定オブジェクトメンバー

リクエストが失敗した場合の追加再試行の回数。

デフォルト値： 3

有効な値： 0 より大きい数値。

`port` - クライアント設定オブジェクトメンバー

エンドポイントのポート。

デフォルト値： 80

有効な値： 数値。

`token_ttl` - クライアント設定オブジェクトメンバー

トークンの TTL。

デフォルト値：21,600 秒 (6 時間、割り当てられた最大時間)。

有効な値： 数値。

endpoint - クライアント設定オブジェクトメンバー

IMDS のエンドポイント。

デフォルト値： `endpoint_mode` が `IPv4` に等しい場合、デフォルトエンドポイントは `http://169.254.169.254` です。`endpoint_mode` が `IPv6` に等しい場合、デフォルトのエンドポイントは `http://[fd00:ec2::254]` です。

有効な値： 有効な URI。

大半の SDK では以下のオペレーションがサポートされています。詳細については、特定の SDK コードベースを参照してください。

endpoint_mode - クライアント設定オブジェクトメンバー

IMDS のエンドポイントモード。

デフォルト値: `IPv4`

有効な値:`IPv4`、`IPv6`|

http_open_timeout - クライアント設定オブジェクトメンバー (名前は異なる場合があります)

接続が開くのを待つ秒数。

デフォルト値： 1 秒。

有効な値： 0 より大きい数値。

http_read_timeout - クライアント設定オブジェクトメンバー (名前は異なる場合があります)

1 つのデータチャンクが読み取られるまでの秒数。

デフォルト値： 1 秒。

有効な値： 0 より大きい数値。

http_debug_output - クライアント設定オブジェクトメンバー (名前は異なる場合があります)

デバッグ用の出力ストリームを設定します。

デフォルト値： なし。

有効な値： STDOUT のような有効な I/O ストリーム。

backoff - クライアント設定オブジェクトメンバー (名前は異なる場合があります)

リトライ間またはお客様が用意したバックオフ関数を呼び出すまでの間にスリープする秒数。これは、デフォルトのエクスポネンシャルバックオフ戦略を使用するよう置き換えます。

デフォルト値： サービスによって異なります。

有効な値： SDK によって異なります。数値でも、カスタム関数の呼び出しでもかまいません。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK for C++	なし	IMDSv2 は内部でのみ使用されます。IMDS 認証情報プロバイダー を参照してください。
SDK for Go V2 (1.x)	あり	
SDK for Go 1.x (V1)	あり	
SDK for Java 2.x	あり	
SDK for Java 1.x	あり	
SDK for JavaScript 3.x	あり	

SDK	サポート	注意または詳細情報
SDK for JavaScript 2.x	あり	
SDK for Kotlin	あり	
SDK for .NET 3.x	あり	
SDK for PHP 3.x	あり	
SDK for Python (Boto3)	あり	
SDK for Ruby 3.x	あり	
SDK for Rust	あり	
のツール PowerShell	あり	

再試行動作

再試行動作には、SDK が AWS のサービスへのリクエストによる障害からの回復を試みるかに関する設定が含まれます。

この機能を設定するには、以下のように使用します。

`max_attempts` - 共有 AWS **`config`**ファイル設定, **`AWS_MAX_ATTEMPTS`** - 環境変数, **`aws.maxAttempts`** - JVM システムプロパティ: Java/Kotlin のみ

1 回のリクエストで行う最大試行回数を指定します。

デフォルト値：この値が指定されていない場合、デフォルトは `retry_mode` の設定の値によって異なります。

- `retry_mode` が `legacy` の場合 – SDK 固有のデフォルト値を使用します（`max_attempts` デフォルトについては、特定の SDK ガイドまたは SDK のコードベースを確認してください）。
- `retry_mode` が `standard` の場合 – 3 回試行します。
- `retry_mode` が `adaptive` の場合 – 3 回試行します。

有効な値： 0 より大きい数値。

`retry_mode` - 共有 AWS **`config`**ファイル設定, **`AWS_RETRY_MODE`** - 環境変数, **`aws.retryMode`** - JVM システムプロパティ: Java/Kotlin のみ

SDK または開発者ツールが再試行を試みる方法を指定します。

デフォルト値： `legacy` はデフォルトの再試行方法です。

有効値:

- `legacy` – ご使用の SDK に固有（特定の SDK ガイドまたは SDK のコードベースを確認してください）。
- `standard` – AWS SDKs 全体の再試行ルールの標準セット。このモードには、再試行される標準エラーセットと再試行クォータのサポートが含まれます。`max_attempts` が明示的に設定されていない限り、このモードでのデフォルトの最大試行回数は 3 回です。
- `adaptive` – 標準モードの機能を含みながら、クライアント側の自動スロットリングを含む実験的な再試行モード。このモードは実験段階であるため、将来的に動作が変更される可能性があります。

`standard` 再試行モードと`adaptive`再試行モードの選択

の使用が により適していることが確実でない限り、`standard`再試行モードを使用することをお勧めします`adaptive`。

> **Note**
>
> `adaptive` モードは、バックエンドサービスがリクエストをスロットリングするスコープに基づいてクライアントをプールしていることを前提としています。これを行わないと、両方のリソースに同じクライアントを使用している場合、1 つのリソースのスロットリングによって、無関係なリソースのリクエストが遅延する可能性があります。

規格	アダプティブ
アプリケーションのユースケース: すべて。	アプリケーションのユースケース： 1. レイテンシーの影響を受けません。 2. クライアントは単一のリソースにのみアクセスします。または、アクセスされるサービスリソースによってクライアントを個別にプールするロジックを提供します。
SDK が停止中に再試行するのを防ぐため、回路ブレークをサポートします。	SDK が停止中に再試行するのを防ぐため、回路ブレークをサポートします。
障害発生時にジッターされたエクスポネンシャルバックオフを使用します。	動的バックオフ期間を使用して、レイテンシーが増加する可能性と引き換えに、失敗したリクエストの数を最小限に抑えるように試みます。
最初のリクエストの試行を遅延することはなく、再試行のみを行います。	最初のリクエスト試行をスロットリングまたは遅延させることができます。

`adaptive` モードを使用する場合、アプリケーションはスロットリングされる可能性のある各リソースを中心に設計されたクライアントを構築する必要があります。この場合、リソースは、各 について考えるよりも細かく調整されます AWS のサービス。 は、リクエストのスロットリングに使用する追加のディメンションを持つ AWS のサービス ことができます。Amazon DynamoDB サービスを例として使用しましょう。DynamoDB は、 AWS リージョン とアクセスされるテーブルを使用してリクエストを調整します。つまり、コードがアクセスしているテーブルの 1 つが、他のテーブルよりもスロットリングされている可能性があります。コードが同じクライアントを使用してすべてのテーブルにアクセスし、それらのテーブルの 1 つへのリクエストがスロットリングされた場合、アダプティブ再試行モードではすべてのテーブルのリクエストレートが低下します。コードは、Region-and-table ペアごとに 1 つのクライアントを持つように設計する必要があります。`adaptive` モードの使用時に予期しないレイテンシーが発生した場合は、使用しているサービスの特定の AWS ドキュメントガイドを参照してください。

再試行モードの実装の詳細

以下は、`standard` と `adaptive` 再試行モードの両方の大まかな擬似コードです。

```
MakeSDKRequest() {
```

```
  attempts = 0
  loop {
    GetSendToken()
    response = SendHTTPRequest()
    RequestBookkeeping(response)
    if not Retryable(response)
      return response
    attempts += 1
    if attempts >= MAX_ATTEMPTS:
      return response
    if not HasRetryQuota(response)
      return response
    delay = ExponentialBackoff(attempts)
    sleep(delay)
  }
}
```

擬似コードで使用されるコンポーネントの詳細は次のとおりです。

`GetSendToken`:

トークンバケットは adaptive リトライモードでのみ使用されます。トークンバケットでは、リクエストを開始するためにトークンを用意しておく必要があるため、リクエストレートが最大になります。SDK クライアントは、リクエストを迅速に失敗させるか、トークンが使用可能になるまでブロックするように設定できます。

クライアント側のレート制限は、最初は、トークンの許容量を上限とする任意のレートでリクエストを送信できるようにするアルゴリズムです。ただし、スロットリングされたレスポンスが検出されると、クライアント rate-of-request はそれに応じて制限されます。また、応答の受信が正常に終了すると、それに応じてトークンの許容量が増加します。

アダプティブレート制限を使用すると、 SDKs は の容量をより適切に対応するために、リクエストの送信速度を遅くすることができます AWS のサービス。

`SendHTTPRequest`:

ほとんどの AWS SDKs は、接続プールを使用する HTTP ライブラリを使用して、HTTP リクエストを行うときに既存の接続を再利用できるようにします。通常、スロットリングエラーが原因でリクエストを再試行すると、接続は再利用されます。一時的なエラーが原因で再試行しても、リクエストは再利用されません。

`RequestBookkeeping`:

リクエストが正常に終了したら、再試行クォータを更新する必要があります。adaptive 再試行モードの場合のみ、maxsendrate 状態変数は受信した応答の種類に基づいて更新されます。

Retryable:

このステップでは、以下に基づいて応答を再試行できるかどうかを判断します。

- HTTP ステータスコード 。
- サービスから返されたエラーコード。
- 接続エラーとは、SDK が受信したエラーの中で、サービスからの HTTP 応答が受信されないすべてのエラーを指します。

一時的なエラー (HTTP ステータスコード 400、408、500、502、503、504) とスロットリングエラー (HTTP ステータスコード 400、403、429、502、503、509) はすべて再試行される可能性があります。SDK の再試行動作は、エラーコードまたはサービスからのその他のデータと組み合わせて決定されます。

MAX_ATTEMPTS:

config ファイル設定または環境変数によって指定されます。

HasRetryQuota

このステップでは、トークンを再試行クォータバケットで使用できるようにすることで、再試行リクエストをスロットルします。リトライクォータバケットは、正常に終了する可能性が低い再試行を防ぐためのメカニズムです。これらのクォータは SDK に依存し、多くの場合クライアントに依存し、場合によってはサービスエンドポイントにも依存します。利用可能な再試行クォータトークンは、さまざまな理由でリクエストが失敗すると削除され、成功すると補充されます。トークンがなくなると、再試行ループは終了します。

ExponentialBackoff

再試行可能なエラーの場合、再試行遅延は台形型エクスポネンシャルバックオフを使用して計算されます。SDK はジッター付きの切り捨て二進エクスポネンシャルバックオフを使用します。次のアルゴリズムは、i リクエストに対する応答の休止時間 (秒単位) がどのように定義されているかを示しています。

```
seconds_to_sleep_i = min(b*r^i, MAX_BACKOFF)
```

前述のアルゴリズムでは、以下の値が適用されます。

```
b = random number within the range of: 0 <= b <= 1

r = 2
```

ほとんどの SDK では `MAX_BACKOFF = 20 seconds` です。確認のため、特定の SDK ガイドまたはソースコードを参照してください。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK for C++	はい	
SDK for Go V2 (1.x)	はい	
SDK for Go 1.x (V1)	いいえ	
SDK for Java 2.x	はい	
SDK for Java 1.x	はい	JVM システムプロパティ: `com.amazonaws.sdk.maxAttempts` の代わりに を使用し`aws.maxAttempts` 、`com.amazonaws.sdk.retryMode` の代わりに を使用します`aws.retryMode` 。
SDK for JavaScript 3.x	はい	

SDK	サポート	注意または詳細情報
SDK for JavaScript 2.x	いいえ	最大再試行回数、ジッターを伴うエクスポネンシャルバックオフ、再試行バックオフのカスタムメソッドのオプションをサポートします。
SDK for Kotlin	はい	
SDK for .NET 3.x	はい	
SDK for PHP 3.x	はい	
SDK for Python (Boto3)	はい	
SDK for Ruby 3.x	はい	
SDK for Rust	はい	
のツール PowerShell	はい	

リクエスト圧縮

AWS SDKsとツールは、圧縮されたペイロードの受信をサポートする にリクエストを送信するときに AWS のサービス 、ペイロードを自動的に圧縮できます。サービスに送信する前にクライアントでペイロードを圧縮すると、サービスにデータを送信するために必要なリクエストの総数と帯域幅が減り、ペイロードサイズに対するサービスの制限を理由として失敗するリクエストも減る可能性があります。圧縮では、SDK またはツールは、サービスと SDK の両方によってサポートされるエンコーディングアルゴリズムを選択します。ただし、可能なエンコーディングの現在のリストは gzip のみで構成されていますが、将来的には拡張される可能性があります。

リクエスト圧縮は、アプリケーションが Amazon CloudWatchを使用している場合に特に役立ちます。 CloudWatch は、ログ、メトリクス、イベントの形式でモニタリングおよび運用データを収集するモニタリングおよびオブザーバビリティサービスです。圧縮をサポートするサービスオペレーションの一例は、 CloudWatchの PutMetricDataAPI メソッドです。

この機能を設定するには、以下のように使用します。

`disable_request_compression` - 共有 AWS **`config`**ファイル設定, **`AWS_DISABLE_REQUEST_COMPRESSION`** - 環境変数, **`aws.disableRequestCompression`** - JVM システムプロパティ: Java/Kotlin のみ

オンまたはオフにして、SDK またはツールがリクエストを送信する前にペイロードを圧縮するかどうかを決定します。

デフォルト値: `false`

有効値:

- **`true`** – リクエスト圧縮をオフにします。
- **`false`** – 可能な場合はリクエスト圧縮を使用します。

`request_min_compression_size_bytes` - 共有 AWS **`config`**ファイル設定, **`AWS_REQUEST_MIN_COMPRESSION_SIZE_BYTES`** - 環境変数, **`aws.requestMinCompressionSizeBytes`** - JVM システムプロパティ: Java/Kotlin のみ

SDK またはツールが圧縮する必要があるリクエスト本文の最小サイズ (バイト) を設定します。小さなペイロードは圧縮すると長くなる可能性があるため、圧縮を実行することが有意義である下限が存在します。この値は包括的であり、この値以上のリクエストサイズは圧縮されます。

デフォルト値: 10,240 バイト

有効な値: 0 ~ 10,485,760 バイトの整数値。

AWS SDKs との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK for C++	あり	
SDK for Go V2 (1.x)	あり	
SDK for Go 1.x (V1)	なし	
SDK for Java 2.x	あり	
SDK for Java 1.x	なし	
SDK for JavaScript 3.x	あり	
SDK for JavaScript 2.x	なし	
SDK for Kotlin	あり	
SDK for .NET 3.x	あり	
SDK for PHP 3.x	あり	
SDK for Python (Boto3)	あり	

SDK	サポート	注意または詳細情報
SDK for Ruby 3.x	あり	
SDK for Rust	あり	
のツール PowerShell	あり	

サービス固有のエンドポイント

サービス固有のエンドポイント設定により、API リクエストに任意のエンドポイントを使用するオプションが得られ、この選択は持続します。これらの設定により、ローカルエンドポイント、VPC エンドポイント、およびサードパーティのローカル AWS 開発環境を柔軟にサポートできます。テスト環境と本番環境には異なるエンドポイントを使用できます。エンドポイント URL は個別の AWS のサービスに指定できます。

この機能を設定するには、以下のように使用します。

`endpoint_url` - 共有 AWS **`config`**ファイル設定, **`AWS_ENDPOINT_URL`** - 環境変数, **`aws.endpointUrl`** - JVM システムプロパティ: Java/Kotlin のみ

プロファイル内で直接指定するか、環境変数として指定した場合、この設定はすべてのサービスリクエストに使用されるエンドポイントを指定します。このエンドポイントは、設定されているサービス固有のエンドポイントによって上書きされます。

共有ファイルの `services`セクション AWS `config`内でこの設定を使用して、特定のサービスのカスタムエンドポイントを設定することもできます。`services` 内のサブセクションで使用するすべてのサービス識別子キーのリストについては、「サービス固有のエンドポイントの識別子」を参照してください。

デフォルト値: none

有効な値： エンドポイントのスキームとホストを含む URL。URL は、必要に応じて 1 つ以上のパスセグメントを含むパスコンポーネントを含めることができます。

AWS_ENDPOINT_URL_<SERVICE> 環境変数, **aws.endpointUrl<ServiceName>** - JVM システムプロパティ: Java/Kotlin のみ

AWS_ENDPOINT_URL_<SERVICE>は識別子<SERVICE>であり AWS のサービス 、特定のサービスのカスタムエンドポイントを設定します。サービス固有の環境変数のリストについては、「サービス固有のエンドポイントの識別子」を参照してください。

このサービス固有のエンドポイントは、AWS_ENDPOINT_URL に設定されているグローバルエンドポイントよりも優先されます。

デフォルト値: none

有効な値： エンドポイントのスキームとホストを含む URL。URL は、必要に応じて 1 つ以上のパスセグメントを含むパスコンポーネントを含めることができます。

ignore_configured_endpoint_urls - 共有 AWS **config**ファイル設定, **AWS_IGNORE_CONFIGURED_ENDPOINT_URLS** - 環境変数, **aws.ignoreConfiguredEndpointUrls** - JVM システムプロパティ: Java/Kotlin のみ

この設定は、すべてのカスタムエンドポイント設定を無視するために使用されます。

コードまたはサービスクライアント自体に設定されている明示的なエンドポイントは、この設定に関係なく使用されることに注意してください。例えば、 `--endpoint-url` コマンドに AWS CLI コマンドラインパラメータを含めたり、エンドポイント URL をクライアントコンストラクタに渡したりすると、常に有効になります。

デフォルト値: `false`

有効値:

- **true** — SDK またはツールは、エンドポイント URL を設定するための `config` 共有ファイルや環境変数からカスタム設定オプションを読み取ることはありません。
- **false** — SDK またはツールは、`config` 共有ファイルまたは環境変数からユーザーが提供したエンドポイントをすべて使用します。

環境変数を使用したエンドポイントの設定

すべてのサービスのリクエストをカスタムエンドポイント URL にルーティングするには、AWS_ENDPOINT_URL グローバル環境変数を設定します。

```
export AWS_ENDPOINT_URL=http://localhost:4567
```

特定の のリクエストをカスタムエンドポイント URL AWS のサービス にルーティングするには、 AWS_ENDPOINT_URL_<SERVICE>環境変数を使用します。 の Amazon DynamoDB は serviceIdですDynamoDB。このサービスのエンドポイント URL 環境変数は AWS_ENDPOINT_URL_DYNAMODB です。このエンドポイントは、このサービスのために AWS_ENDPOINT_URL に設定されているグローバルエンドポイントよりも優先されます。

```
export AWS_ENDPOINT_URL_DYNAMODB=http://localhost:5678
```

別の例として、 には serviceIdの AWS Elastic Beanstalk がありますElastic Beanstalk。AWS のサービス 識別子は、すべてのスペースをアンダースコアに置き換え、すべての文字を大文字にserviceIdすることで、API モデルの に基づいています。このサービスにエンドポイントを設定するための、対応する環境変数は AWS_ENDPOINT_URL_ELASTIC_BEANSTALK です。サービス固有の環境変数のリストについては、「サービス固有のエンドポイントの識別子」を参照してください。

```
export AWS_ENDPOINT_URL_ELASTIC_BEANSTALK=http://localhost:5567
```

config 共有ファイルを使用してエンドポイントを設定します

config 共有ファイルでは、 endpoint_url がさまざまな場所でさまざまな機能に使用されます。

- profile 内で endpoint_url を直接指定すると、そのエンドポイントがグローバルエンドポイントになります。
- services セクション内のサービス ID キーの下に endpoint_url をネストすると、そのエンドポイントはそのサービスに対して行われたリクエストにのみ適用されます。共有 config ファイル内の services セクションの定義について詳しくは、「設定ファイルの形式」を参照してください。

次の例では、services 定義を使用して Amazon S3 に使用されることとなるサービス固有のエンドポイント URL と、他のすべてのサービスに使用されることとなるカスタムグローバルエンドポイントを設定します。

```
[profile dev-s3-specific-and-global]
endpoint_url = http://localhost:1234
services = s3-specific

[services s3-specific]
s3 =
```

```
  endpoint_url = https://play.min.io:9000
```

1 つのプロファイルで複数のサービスのエンドポイントを設定できます。この例では、Amazon S3 と AWS Elastic Beanstalk のサービス固有のエンドポイント URLs を同じプロファイルに設定する方法を示します。 には `serviceId`の AWS Elastic Beanstalk がありますElastic Beanstalk。Amazon S3 AWS のサービス 識別子は API モデルの に基づいて`serviceId`おり、すべてのスペースをアンダースコアに置き換え、すべての文字を小文字に置き換えます。したがって、サービス ID キーは `elastic_beanstalk` になり、このサービスの設定は `elastic_beanstalk =` の行から開始されます。`services` セクションで使用するすべてのサービス識別子キーのリストについては、「サービス固有のエンドポイントの識別子」を参照してください。

```
[services testing-s3-and-eb]
s3 =
  endpoint_url = http://localhost:4567
elastic_beanstalk =
  endpoint_url = http://localhost:8000

[profile dev]
services = testing-s3-and-eb
```

サービス設定セクションは複数のプロファイルで使用できます。たとえば、2つのプロファイルが同じ定義 `services` を使用し、他のプロファイルプロパティを変更することができます。

```
[services testing-s3]
s3 =
  endpoint_url = https://localhost:4567

[profile testing-json]
output = json
services = testing-s3

[profile testing-text]
output = text
services = testing-s3
```

ロールベースの認証情報を使用してプロファイル内のエンドポイントを設定します

プロファイルに IAM Assume Role 機能の `source_profile` パラメータによって設定されたロールベースの認証情報がある場合、SDK は指定されたプロファイルのサービス設定のみを使用します。

ロールチェーンされたプロファイルは使用されません。例えば、次の共有 config ファイルを使用します。

```
[profile A]
credential_source = Ec2InstanceMetadata
endpoint_url = https://profile-a-endpoint.aws/

[profile B]
source_profile = A
role_arn = arn:aws:iam::123456789012:role/roleB
services = profileB

[services profileB]
ec2 =
  endpoint_url = https://profile-b-ec2-endpoint.aws
```

プロファイル B を使用してコード内で Amazon EC2 を呼び出すと、エンドポイントは `https://profile-b-ec2-endpoint.aws` として解決されます。コードが他のサービスにリクエストを送信した場合、エンドポイントの解決はカスタムロジックには従いません。エンドポイントはプロファイル A で定義されたグローバルエンドポイントには解決されません。グローバルエンドポイントを B プロファイルに対して有効にするには、プロファイル B 内で直接 `endpoint_url` を設定する必要があります。`source_profile` 設定の詳細については、ロール認証情報プロバイダーを引き受けます を参照してください。

設定の優先順位

この機能の設定は同時に使用できますが、1 つのサービスにつき 1 つの値が優先されます。特定のに対して行われた API コールの場合 AWS のサービス、値の選択には次の順序が使用されます。

1. コードまたはサービスクライアント自体に設定されている明示的な設定は、他の設定よりも優先されます。
   - の場合 AWS CLI、これは`--endpoint-url`コマンドラインパラメータによって提供される値です。SDK の場合、明示的な割り当ては、 AWS のサービス クライアントまたは設定オブジェクトをインスタンス化するときに設定したパラメータの形式をとることができます。
2. サービス固有の環境変数 (`AWS_ENDPOINT_URL_DYNAMODB` など) によって提供される値。
3. `AWS_ENDPOINT_URL` グローバルエンドポイント環境変数によって提供される値。
4. `endpoint_url` 設定によって得られる値は、`config` 共有ファイルの `services` セクション内のサービス ID キーの下にネストされます。

5. 共有 `config` ファイルの `profile` 内で直接指定された `endpoint_url` 設定によって得られる値。
6. それぞれのデフォルトのエンドポイント URL AWS のサービス が最後に使用されます。

AWS SDKsとの互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。JVM システムプロパティ設定は、 AWS SDK for Java と AWS SDK for Kotlin でのみサポートされます。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK for C++	なし	
SDK for Go V2 (1.x)	あり	
SDK for Go 1.x (V1)	なし	
SDK for Java 2.x	なし	
SDK for Java 1.x	なし	
SDK for JavaScript 3.x	あり	
SDK for JavaScript 2.x	なし	

SDK	サポート	注意または詳細情報
SDK for Kotlin	あり	
SDK for .NET 3.x	あり	
SDK for PHP 3.x	あり	
SDK for Python (Boto3)	あり	
SDK for Ruby 3.x	あり	
SDK for Rust	なし	
のツール PowerShell	あり	

サービス固有のエンドポイントの識別子

次の表の識別子の使用方法と使用場所については、「サービス固有のエンドポイント」を参照してください。

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
AccessAnalyzer	ac ly	AWS_ENDPOINT_URL_ACCESSANALYZER
Account	ac	AWS_ENDPOINT_URL_ACCOUNT
ACM	ac	AWS_ENDPOINT_URL_ACM
ACM PCA	ac	AWS_ENDPOINT_URL_ACM_PCA
Alexa For Business	a _b	AWS_ENDPOINT_URL_ALEXA_FOR_BUSINESS
amp	ar	AWS_ENDPOINT_URL_AMP
Amplify	ar	AWS_ENDPOINT_URL_AMPLIFY
AmplifyBackend	ar cl	AWS_ENDPOINT_URL_AMPLIFYBACKEND
AmplifyUIBuilder	ar bu	AWS_ENDPOINT_URL_AMPLIFYUIBUILDER

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
API Gateway	ap ay	AWS_ENDPOINT_URL_API_GATEWAY
ApiGatewayManagementApi	ap yn nt	AWS_ENDPOINT_URL_APIGATEWAYMANAGEMENTAPI
ApiGatewayV2	ap yv	AWS_ENDPOINT_URL_APIGATEWAYV2
AppConfig	ap	AWS_ENDPOINT_URL_APPCONFIG
AppConfigData	ap da	AWS_ENDPOINT_URL_APPCONFIGDATA
AppFabric	ap	AWS_ENDPOINT_URL_APPFABRIC
Appflow	ap	AWS_ENDPOINT_URL_APPFLOW
AppIntegrations	ap at	AWS_ENDPOINT_URL_APPINTEGRATIONS

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Application Auto Scaling	ap or ca	AWS_ENDPOINT_URL_APPLICATION_AUTO_SCALING
Application Insights	ap or ts	AWS_ENDPOINT_URL_APPLICATION_INSIGHTS
ApplicationCostProfiler	ap or f:	AWS_ENDPOINT_URL_APPLICATIONCOSTPROFILER
App Mesh	ap	AWS_ENDPOINT_URL_APP_MESH
AppRunner	ap	AWS_ENDPOINT_URL_APPRUNNER
AppStream	ap	AWS_ENDPOINT_URL_APPSTREAM
AppSync	ap	AWS_ENDPOINT_URL_APPSYNC

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
ARC Zonal Shift	a _	AWS_ENDPOINT_URL_ARC_ZONAL_SHIFT
Artifact	a	AWS_ENDPOINT_URL_ARTIFACT
Athena	a	AWS_ENDPOINT_URL_ATHENA
AuditManager	a g	AWS_ENDPOINT_URL_AUDITMANAGER
Auto Scaling	a i	AWS_ENDPOINT_URL_AUTO_SCALING
Auto Scaling Plans	a i	AWS_ENDPOINT_URL_AUTO_SCALING_PLANS
b2bi	b	AWS_ENDPOINT_URL_B2BI
Backup	b	AWS_ENDPOINT_URL_BACKUP
Backup Gateway	b t	AWS_ENDPOINT_URL_BACKUP_GATEWAY

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
BackupStorage	ba ra	AWS_ENDPOINT_URL_BACKUPSTORAGE
Batch	ba	AWS_ENDPOINT_URL_BATCH
BCM Data Exports	bc ex	AWS_ENDPOINT_URL_BCM_DATA_EXPORTS
Bedrock	be	AWS_ENDPOINT_URL_BEDROCK
Bedrock Agent	be ge	AWS_ENDPOINT_URL_BEDROCK_AGENT
Bedrock Agent Runtime	be ge in	AWS_ENDPOINT_URL_BEDROCK_AGENT_RUNTIME
Bedrock Runtime	be un	AWS_ENDPOINT_URL_BEDROCK_RUNTIME
billingconductor	bi nc	AWS_ENDPOINT_URL_BILLINGCONDUCTOR

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Braket	b	AWS_ENDPOINT_URL_BRAKET
Budgets	bu	AWS_ENDPOINT_URL_BUDGETS
Cost Explorer	co o	AWS_ENDPOINT_URL_COST_EXPLORER
chatbot	ch	AWS_ENDPOINT_URL_CHATBOT
Chime	ch	AWS_ENDPOINT_URL_CHIME
Chime SDK Identity	ch _	AWS_ENDPOINT_URL_CHIME_SDK_IDENTITY
Chime SDK Media Pipelines	ch _m pe	AWS_ENDPOINT_URL_CHIME_SDK_MEDIA_PIPELINES
Chime SDK Meetings	ch _m	AWS_ENDPOINT_URL_CHIME_SDK_MEETINGS

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Chime SDK Messaging	chime_messaging	AWS_ENDPOINT_URL_CHIME_SDK_MESSAGING
Chime SDK Voice	chime_voice	AWS_ENDPOINT_URL_CHIME_SDK_VOICE
CleanRooms	cleanrooms	AWS_ENDPOINT_URL_CLEANROOMS
CleanRoomsML	cleanroomsml	AWS_ENDPOINT_URL_CLEANROOMSML
Cloud9	cloud9	AWS_ENDPOINT_URL_CLOUD9
CloudControl	cloudcontrol	AWS_ENDPOINT_URL_CLOUDCONTROL
CloudDirectory	clouddirectory	AWS_ENDPOINT_URL_CLOUDDIRECTORY
CloudFormation	cloudformation	AWS_ENDPOINT_URL_CLOUDFORMATION

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
CloudFront	c t	AWS_ENDPOINT_URL_CLOUDFRONT
CloudFront KeyValueStore	c t_ es	AWS_ENDPOINT_URL_CLOUDFRONT_KEYVALUESTORE
CloudHSM	c	AWS_ENDPOINT_URL_CLOUDHSM
CloudHSM V2	c v	AWS_ENDPOINT_URL_CLOUDHSM_V2
CloudSearch	c ch	AWS_ENDPOINT_URL_CLOUDSEARCH
CloudSearch Domain	c ch	AWS_ENDPOINT_URL_CLOUDSEARCH_DOMAIN
CloudTrail	c l	AWS_ENDPOINT_URL_CLOUDTRAIL

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
CloudTrail Data	cl	AWS_ENDPOINT_URL_CLOUDTRAIL_DATA
CloudWatch	ch	AWS_ENDPOINT_URL_CLOUDWATCH
codeartifact	ca	AWS_ENDPOINT_URL_CODEARTIFACT
CodeBuild	co	AWS_ENDPOINT_URL_CODEBUILD
CodeCatalyst	cy	AWS_ENDPOINT_URL_CODECATALYST
CodeCommit	ct	AWS_ENDPOINT_URL_CODECOMMIT
CodeDeploy	cy	AWS_ENDPOINT_URL_CODEDEPLOY
CodeGuru Reviewer	re	AWS_ENDPOINT_URL_CODEGURU_REVIEWER

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
CodeGuru Security	co se	AWS_ENDPOINT_URL_CODEGURU_SECURITY
CodeGuruProfiler	co ro	AWS_ENDPOINT_URL_CODEGURUPROFILER
CodePipeline	co in	AWS_ENDPOINT_URL_CODEPIPELINE
CodeStar	co	AWS_ENDPOINT_URL_CODESTAR
CodeStar connections	co co ns	AWS_ENDPOINT_URL_CODESTAR_CONNECTIONS
codestar notifications	co no ic	AWS_ENDPOINT_URL_CODESTAR_NOTIFICATIONS
Cognito Identity	co de	AWS_ENDPOINT_URL_COGNITO_IDENTITY

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Cognito Identity Provider	co de ro	AWS_ENDPOINT_URL_COGNITO_IDENTITY_PROVIDER
Cognito Sync	co yn	AWS_ENDPOINT_URL_COGNITO_SYNC
Comprehend	co d	AWS_ENDPOINT_URL_COMPREHEND
ComprehendMedical	co dr	AWS_ENDPOINT_URL_COMPREHENDMEDICAL
Compute Optimizer	co pt	AWS_ENDPOINT_URL_COMPUTE_OPTIMIZER
Config Service	co rv	AWS_ENDPOINT_URL_CONFIG_SERVICE
Connect	co	AWS_ENDPOINT_URL_CONNECT

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Connect Contact Lens	co on ns	AWS_ENDPOINT_URL_CONNECT_CONTACT_LENS
ConnectCampaigns	co mp	AWS_ENDPOINT_URL_CONNECTCAMPAIGNS
ConnectCases	co se	AWS_ENDPOINT_URL_CONNECTCASES
ConnectParticipant	co rt	AWS_ENDPOINT_URL_CONNECTPARTICIPANT
ControlTower	co we	AWS_ENDPOINT_URL_CONTROLTOWER
Cost Optimization Hub	co mi hu	AWS_ENDPOINT_URL_COST_OPTIMIZATION_HUB

serviceId	共有AWSconイルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Cost and Usage Report Service	co us oi ce	AWS_ENDPOINT_URL_COST_AND_USAGE_REPORT_SERVICE
Customer Profiles	cu pi	AWS_ENDPOINT_URL_CUSTOMER_PROFILES
DataBrew	da	AWS_ENDPOINT_URL_DATABREW
DataExchange	da ng	AWS_ENDPOINT_URL_DATAEXCHANGE
Data Pipeline	da li	AWS_ENDPOINT_URL_DATA_PIPELINE
DataSync	da	AWS_ENDPOINT_URL_DATASYNC
DataZone	da	AWS_ENDPOINT_URL_DATAZONE
DAX	da	AWS_ENDPOINT_URL_DAX

serviceId	共有AWS**ce**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Detective	de	AWS_ENDPOINT_URL_DETECTIVE
Device Farm	de rn	AWS_ENDPOINT_URL_DEVICE_FARM
DevOps Guru	de ru	AWS_ENDPOINT_URL_DEVOPS_GURU
Direct Connect	di nn	AWS_ENDPOINT_URL_DIRECT_CONNECT
Application Discovery Service	ap on e: ce	AWS_ENDPOINT_URL_APPLICATION_DISCOVERY_SERVICE
DLM	dl	AWS_ENDPOINT_URL_DLM
Database Migration Service	da m: _s	AWS_ENDPOINT_URL_DATABASE_MIGRATION_SERVICE

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
DocDB	do	AWS_ENDPOINT_URL_DOCDB
DocDB Elastic	do st	AWS_ENDPOINT_URL_DOCDB_ELASTIC
drs	dr	AWS_ENDPOINT_URL_DRS
Directory Service	di _s	AWS_ENDPOINT_URL_DIRECTORY_SERVICE
DynamoDB	dy	AWS_ENDPOINT_URL_DYNAMODB
DynamoDB Streams	dy st	AWS_ENDPOINT_URL_DYNAMODB_STREAMS
EBS	eb	AWS_ENDPOINT_URL_EBS
EC2	ec	AWS_ENDPOINT_URL_EC2
EC2 Instance Connect	ec no ct	AWS_ENDPOINT_URL_EC2_INSTANCE_CONNECT
ECR	ec	AWS_ENDPOINT_URL_ECR

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
ECR PUBLIC	ec c	AWS_ENDPOINT_URL_ECR_PUBLIC
ECS	ec	AWS_ENDPOINT_URL_ECS
EFS	ef	AWS_ENDPOINT_URL_EFS
EKS	el	AWS_ENDPOINT_URL_EKS
EKS Auth	el	AWS_ENDPOINT_URL_EKS_AUTH
Elastic Inference	el nf	AWS_ENDPOINT_URL_ELASTIC_INFERENCE
ElastiCache	el he	AWS_ENDPOINT_URL_ELASTICACHE
Elastic Beanstalk	el ea	AWS_ENDPOINT_URL_ELASTIC_BEANSTALK
Elastic Transcoder	el ra	AWS_ENDPOINT_URL_ELASTIC_TRANSCODER

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Elastic Load Balancing	e o c	AWS_ENDPOINT_URL_ELASTIC_LOAD_BALANCING
Elastic Load Balancing v2	e o c	AWS_ENDPOINT_URL_ELASTIC_LOAD_BALANCING_V2
EMR	er	AWS_ENDPOINT_URL_EMR
EMR containers	er in	AWS_ENDPOINT_URL_EMR_CONTAINERS
EMR Serverless	er r	AWS_ENDPOINT_URL_EMR_SERVERLESS
EntityResolution	er o	AWS_ENDPOINT_URL_ENTITYRESOLUTION
Elasticsearch Service	e a i	AWS_ENDPOINT_URL_ELASTICSEARCH_SERVICE

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
EventBridge	ev ge	AWS_ENDPOINT_URL_EVENTBRIDGE
Evidently	ev	AWS_ENDPOINT_URL_EVIDENTLY
finspace	f:	AWS_ENDPOINT_URL_FINSPACE
finspace data	f: da	AWS_ENDPOINT_URL_FINSPACE_DATA
Firehose	f:	AWS_ENDPOINT_URL_FIREHOSE
fis	f:	AWS_ENDPOINT_URL_FIS
FMS	fr	AWS_ENDPOINT_URL_FMS
forecast	fo	AWS_ENDPOINT_URL_FORECAST
forecastquery	fo ue	AWS_ENDPOINT_URL_FORECASTQUERY
FraudDetector	f: ct	AWS_ENDPOINT_URL_FRAUDDETECTOR

serviceId	共有AWSconイルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
FreeTier	f	AWS_ENDPOINT_URL_FREETIER
FSx	fs	AWS_ENDPOINT_URL_FSX
GameLift	ga	AWS_ENDPOINT_URL_GAMELIFT
Glacier	gl	AWS_ENDPOINT_URL_GLACIER
Global Accelerator	gl ce	AWS_ENDPOINT_URL_GLOBAL_ACCELERATOR
Glue	gl	AWS_ENDPOINT_URL_GLUE
grafana	gr	AWS_ENDPOINT_URL_GRAFANA
Greengrass	gr s	AWS_ENDPOINT_URL_GREENGRASS
GreengrassV2	gr sv	AWS_ENDPOINT_URL_GREENGRASSV2
GroundStation	gr t	AWS_ENDPOINT_URL_GROUNDSTATION

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
GuardDuty	gu	AWS_ENDPOINT_URL_GUARDDUTY
Health	he	AWS_ENDPOINT_URL_HEALTH
HealthLake	he e	AWS_ENDPOINT_URL_HEALTHLAKE
Honeycode	ho	AWS_ENDPOINT_URL_HONEYCODE
IAM	ia	AWS_ENDPOINT_URL_IAM
identitystore	id to	AWS_ENDPOINT_URL_IDENTITYSTORE
imagebuilder	im de	AWS_ENDPOINT_URL_IMAGEBUILDER
ImportExport	im ot	AWS_ENDPOINT_URL_IMPORTEXPORT

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Inspector	in	AWS_ENDPOINT_URL_INSPECTOR
Inspector Scan	in _s	AWS_ENDPOINT_URL_INSPECTOR_SCAN
Inspector2	in 2	AWS_ENDPOINT_URL_INSPECTOR2
InternetMonitor	in on	AWS_ENDPOINT_URL_INTERNETMONITOR
IoT	io	AWS_ENDPOINT_URL_IOT
IoT Data Plane	io p	AWS_ENDPOINT_URL_IOT_DATA_PLANE
IoT Jobs Data Plane	io da e	AWS_ENDPOINT_URL_IOT_JOBS_DATA_PLANE

serviceId	共有AWSconイルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
IoT 1Click Devices Service	ioK__S	AWS_ENDPOINT_URL_IOT_1CLICK_DEVICES_SERVICE
IoT 1Click Projects	ioK_s	AWS_ENDPOINT_URL_IOT_1CLICK_PROJECTS
IoTAnalytics	ioio	AWS_ENDPOINT_URL_IOTANALYTICS
IotDeviceAdvisor	ioad	AWS_ENDPOINT_URL_IOTDEVICEADVISOR
IoT Events	ios	AWS_ENDPOINT_URL_IOT_EVENTS
IoT Events Data	ios_	AWS_ENDPOINT_URL_IOT_EVENTS_DATA
IoTFleetHub	iouh	AWS_ENDPOINT_URL_IOTFLEETHUB

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
IoTFleetWise	io is	AWS_ENDPOINT_URL_IOTFLEETWISE
IoTSecureTunneling	io tu	AWS_ENDPOINT_URL_IOTSECURETUNNELING
IoTSiteWise	io se	AWS_ENDPOINT_URL_IOTSITEWISE
IoTThingsGraph	io gr	AWS_ENDPOINT_URL_IOTTHINGSGRAPH
IoTTwinMaker	io ke	AWS_ENDPOINT_URL_IOTTWINMAKER
IoT Wireless	io es	AWS_ENDPOINT_URL_IOT_WIRELESS
ivs	iv	AWS_ENDPOINT_URL_IVS
IVS RealTime	iv in	AWS_ENDPOINT_URL_IVS_REALTIME

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
ivschat	iv	AWS_ENDPOINT_URL_IVSCHAT
Kafka	ka	AWS_ENDPOINT_URL_KAFKA
KafkaConnect	ka ec	AWS_ENDPOINT_URL_KAFKACONNECT
kendra	ke	AWS_ENDPOINT_URL_KENDRA
Kendra Ranking	ke nk	AWS_ENDPOINT_URL_KENDRA_RANKING
Keyspaces	ke	AWS_ENDPOINT_URL_KEYSPACES
Kinesis	ki	AWS_ENDPOINT_URL_KINESIS
Kinesis Video Archived Media	ki id iv a	AWS_ENDPOINT_URL_KINESIS_VIDEO_ARCHIVED_MEDIA

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Kinesis Video Media	k: ic a	AWS_ENDPOINT_URL_KINESIS_VIDEO_MEDIA
Kinesis Video Signaling	k: ic a:	AWS_ENDPOINT_URL_KINESIS_VIDEO_SIGNALING
Kinesis Video WebRTC Storage	k: ic tc e	AWS_ENDPOINT_URL_KINESIS_VIDEO_WEBRTC_STORAGE
Kinesis Analytics	k: na	AWS_ENDPOINT_URL_KINESIS_ANALYTICS
Kinesis Analytics V2	k: na v2	AWS_ENDPOINT_URL_KINESIS_ANALYTICS_V2
Kinesis Video	k: ic	AWS_ENDPOINT_URL_KINESIS_VIDEO

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
KMS	kms	AWS_ENDPOINT_URL_KMS
LakeFormation	lakeformation	AWS_ENDPOINT_URL_LAKEFORMATION
Lambda	lambda	AWS_ENDPOINT_URL_LAMBDA
Launch Wizard	launchwizard	AWS_ENDPOINT_URL_LAUNCH_WIZARD
Lex Model Building Service	lex-models	AWS_ENDPOINT_URL_LEX_MODEL_BUILDING_SERVICE
Lex Runtime Service	lex-runtime	AWS_ENDPOINT_URL_LEX_RUNTIME_SERVICE
Lex Models V2	lex-models-v2	AWS_ENDPOINT_URL_LEX_MODELS_V2
Lex Runtime V2	lex-runtime-v2	AWS_ENDPOINT_URL_LEX_RUNTIME_V2

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
License Manager	l: ar	AWS_ENDPOINT_URL_LICENSE_MANAGER
License Manager Linux Subscriptions	l: ar nu r:	AWS_ENDPOINT_URL_LICENSE_MANAGER_LINUX_SUBSCRIPTIONS
License Manager User Subscriptions	l: ar e: ip	AWS_ENDPOINT_URL_LICENSE_MANAGER_USER_SUBSCRIPTIONS
Lightsail	l:	AWS_ENDPOINT_URL_LIGHTSAIL
Location	lo	AWS_ENDPOINT_URL_LOCATION
CloudWatch Logs	c: h_	AWS_ENDPOINT_URL_CLOUDWATCH_LOGS
CloudWatch Logs	c: h_	AWS_ENDPOINT_URL_CLOUDWATCH_LOGS

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
LookoutEquipment	lo u:	AWS_ENDPOINT_URL_LOOKOUTEQUIPMENT
LookoutMetrics	lo t:	AWS_ENDPOINT_URL_LOOKOUTMETRICS
LookoutVision	lo s:	AWS_ENDPOINT_URL_LOOKOUTVISION
m2	m2	AWS_ENDPOINT_URL_M2
Machine Learning	ma ea	AWS_ENDPOINT_URL_MACHINE_LEARNING
Macie2	ma	AWS_ENDPOINT_URL_MACIE2
ManagedBlockchain	ma oc	AWS_ENDPOINT_URL_MANAGEDBLOCKCHAIN
ManagedBlockchain Query	ma oc qu	AWS_ENDPOINT_URL_MANAGEDBLOCKCHAIN_QUERY

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Marketplace Agreement	marketplaceagreement	AWS_ENDPOINT_URL_MARKETPLACE_AGREEMENT
Marketplace Catalog	marketplacecatalog	AWS_ENDPOINT_URL_MARKETPLACE_CATALOG
Marketplace Deployment	marketplacedeployment	AWS_ENDPOINT_URL_MARKETPLACE_DEPLOYMENT
Marketplace Entitlement Service	marketplaceentitlementservice	AWS_ENDPOINT_URL_MARKETPLACE_ENTITLEMENT_SERVICE
Marketplace Commerce Analytics	marketplacecommerceanalytics	AWS_ENDPOINT_URL_MARKETPLACE_COMMERCE_ANALYTICS

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
MediaConnect	me ec	AWS_ENDPOINT_URL_MEDIACONNECT
MediaConvert	me e:	AWS_ENDPOINT_URL_MEDIACONVERT
MediaLive	me	AWS_ENDPOINT_URL_MEDIALIVE
MediaPackage	me ag	AWS_ENDPOINT_URL_MEDIAPACKAGE
MediaPackage Vod	me ag	AWS_ENDPOINT_URL_MEDIAPACKAGE_VOD
MediaPackageV2	me ag	AWS_ENDPOINT_URL_MEDIAPACKAGEV2
MediaStore	me e	AWS_ENDPOINT_URL_MEDIASTORE
MediaStore Data	me e_	AWS_ENDPOINT_URL_MEDIASTORE_DATA

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
MediaTailor	me o:	AWS_ENDPOINT_URL_MEDIATAILOR
Medical Imaging	me ma	AWS_ENDPOINT_URL_MEDICAL_IMAGING
MemoryDB	me	AWS_ENDPOINT_URL_MEMORYDB
Marketplace Metering	ma ce ng	AWS_ENDPOINT_URL_MARKETPLACE_METERING
Migration Hub	m: _h	AWS_ENDPOINT_URL_MIGRATION_HUB
mgn	mg	AWS_ENDPOINT_URL_MGN
Migration Hub Refactor Spaces	m: _h ct es	AWS_ENDPOINT_URL_MIGRATION_HUB_REFAC TOR_SPACES

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
MigrationHub Config	mhg	AWS_ENDPOINT_URL_MIGRATIONHUB_CONFIG
MigrationHubOrchestrator	mht	AWS_ENDPOINT_URL_MIGRATIONHUBORCHESTRATOR
MigrationHubStrategy	mhgy	AWS_ENDPOINT_URL_MIGRATIONHUBSTRATEGY
Mobile	mo	AWS_ENDPOINT_URL_MOBILE
mq	mo	AWS_ENDPOINT_URL_MQ
MTurk	mt	AWS_ENDPOINT_URL_MTURK
MWAA	mw	AWS_ENDPOINT_URL_MWAA
Neptune	ne	AWS_ENDPOINT_URL_NEPTUNE

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Neptune Graph	neptune-graph	AWS_ENDPOINT_URL_NEPTUNE_GRAPH
neptunedata	neptune-data	AWS_ENDPOINT_URL_NEPTUNEDATA
Network Firewall	network-firewall	AWS_ENDPOINT_URL_NETWORK_FIREWALL
NetworkManager	network-manager	AWS_ENDPOINT_URL_NETWORKMANAGER
NetworkMonitor	network-monitor	AWS_ENDPOINT_URL_NETWORKMONITOR
nimble	nimble	AWS_ENDPOINT_URL_NIMBLE
OAM	oam	AWS_ENDPOINT_URL_OAM
Omics	omics	AWS_ENDPOINT_URL_OMICS
OpenSearch	opensearch	AWS_ENDPOINT_URL_OPENSEARCH

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
OpenSearchServerless	op hs ss	AWS_ENDPOINT_URL_OPENSEARCHSERVERLESS
OpsWorks	op	AWS_ENDPOINT_URL_OPSWORKS
OpsWorksCM	op m	AWS_ENDPOINT_URL_OPSWORKSCM
Organizations	or io	AWS_ENDPOINT_URL_ORGANIZATIONS
OSIS	os	AWS_ENDPOINT_URL_OSIS
Outposts	ou	AWS_ENDPOINT_URL_OUTPOSTS
p8data	p8	AWS_ENDPOINT_URL_P8DATA
p8data	p8	AWS_ENDPOINT_URL_P8DATA
Panorama	pa	AWS_ENDPOINT_URL_PANORAMA

serviceId	共有AWSconイルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Payment Cryptography	pa ry hy	AWS_ENDPOINT_URL_PAYMENT_CRYPTOGRAPHY
Payment Cryptography Data	pa ry hy	AWS_ENDPOINT_URL_PAYMENT_CRYPTOGRAPHY_DATA
Pca Connector Ad	pc ct	AWS_ENDPOINT_URL_PCA_CONNECTOR_AD
Personalize	pe ze	AWS_ENDPOINT_URL_PERSONALIZE
Personalize Events	pe ze	AWS_ENDPOINT_URL_PERSONALIZE_EVENTS
Personalize Runtime	pe ze e	AWS_ENDPOINT_URL_PERSONALIZE_RUNTIME
PI	p:	AWS_ENDPOINT_URL_PI

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Pinpoint	p	AWS_ENDPOINT_URL_PINPOINT
Pinpoint Email	p er	AWS_ENDPOINT_URL_PINPOINT_EMAIL
Pinpoint SMS Voice	p sr	AWS_ENDPOINT_URL_PINPOINT_SMS_VOICE
Pinpoint SMS Voice V2	p sr _	AWS_ENDPOINT_URL_PINPOINT_SMS_VOICE_V2
Pipes	p	AWS_ENDPOINT_URL_PIPES
Polly	p	AWS_ENDPOINT_URL_POLLY
Pricing	p	AWS_ENDPOINT_URL_PRICING
PrivateNetworks	p tw	AWS_ENDPOINT_URL_PRIVATENETWORKS
Proton	p	AWS_ENDPOINT_URL_PROTON

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
QBusiness	qb	AWS_ENDPOINT_URL_QBUSINESS
QConnect	qc	AWS_ENDPOINT_URL_QCONNECT
QLDB	ql	AWS_ENDPOINT_URL_QLDB
QLDB Session	qldb-session	AWS_ENDPOINT_URL_QLDB_SESSION
QuickSight	quicksight	AWS_ENDPOINT_URL_QUICKSIGHT
RAM	ram	AWS_ENDPOINT_URL_RAM
rbin	rbin	AWS_ENDPOINT_URL_RBIN
RDS	rds	AWS_ENDPOINT_URL_RDS
RDS Data	rds-data	AWS_ENDPOINT_URL_RDS_DATA
Redshift	redshift	AWS_ENDPOINT_URL_REDSHIFT

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Redshift Data	redata	AWS_ENDPOINT_URL_REDSHIFT_DATA
Redshift Serverless	redshifts	AWS_ENDPOINT_URL_REDSHIFT_SERVERLESS
Rekognition	rekognition	AWS_ENDPOINT_URL_REKOGNITION
repostspace	repostspace	AWS_ENDPOINT_URL_REPOSTSPACE
resiliencehub	resilience	AWS_ENDPOINT_URL_RESILIENCEHUB
Resource Explorer 2	resourceexplorer2	AWS_ENDPOINT_URL_RESOURCE_EXPLORER_2
Resource Groups	resourcegroups	AWS_ENDPOINT_URL_RESOURCE_GROUPS

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Resource Groups Tagging API	resourcegroupstaggingapi	AWS_ENDPOINT_URL_RESOURCE_GROUPS_TAGGING_API
RoboMaker	robomaker	AWS_ENDPOINT_URL_ROBOMAKER
RolesAnywhere	rolesanywhere	AWS_ENDPOINT_URL_ROLESANYWHERE
Route 53	route53	AWS_ENDPOINT_URL_ROUTE_53
Route53 Recovery Cluster	route53recoverycluster	AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CLUSTER
Route53 Recovery Control Config	route53recoverycontrolconfig	AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CONTROL_CONFIG

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Route53 Recovery Readiness	ro ec ea	AWS_ENDPOINT_URL_ROUTE53_RECOVERY_READINESS
Route 53 Domains	ro do	AWS_ENDPOINT_URL_ROUTE_53_DOMAINS
Route53Resolver	ro so	AWS_ENDPOINT_URL_ROUTE53RESOLVER
RUM	ru	AWS_ENDPOINT_URL_RUM
S3	s3	AWS_ENDPOINT_URL_S3
S3 Control	s3 l	AWS_ENDPOINT_URL_S3_CONTROL
S3Outposts	s3 s	AWS_ENDPOINT_URL_S3OUTPOSTS
SageMaker	sa	AWS_ENDPOINT_URL_SAGEMAKER

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
SageMaker A2I Runtime	sa _a in	AWS_ENDPOINT_URL_SAGEMAKER_A2I_RUNTIME
Sagemaker Edge	sa _e	AWS_ENDPOINT_URL_SAGEMAKER_EDGE
SageMaker FeatureStore Runtime	sa _f to in	AWS_ENDPOINT_URL_SAGEMAKER_FEATURESTORE_RUNTIME
SageMaker Geospatial	sa _g a	AWS_ENDPOINT_URL_SAGEMAKER_GEOSPATIAL
SageMaker Metrics	sa _m	AWS_ENDPOINT_URL_SAGEMAKER_METRICS
SageMaker Runtime	sa _r	AWS_ENDPOINT_URL_SAGEMAKER_RUNTIME

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
savingsplans	sa an	AWS_ENDPOINT_URL_SAVINGSPLANS
Scheduler	sc	AWS_ENDPOINT_URL_SCHEDULER
schemas	sc	AWS_ENDPOINT_URL_SCHEMAS
SimpleDB	s:	AWS_ENDPOINT_URL_SIMPLEDB
Secrets Manager	se an	AWS_ENDPOINT_URL_SECRETS_MANAGER
SecurityHub	se ub	AWS_ENDPOINT_URL_SECURITYHUB
SecurityLake	se al	AWS_ENDPOINT_URL_SECURITYLAKE
ServerlessApplicationRepository	se sa io to	AWS_ENDPOINT_URL_SERVERLESSAPPLICATIONREPOSITORY

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Service Quotas	se uo	AWS_ENDPOINT_URL_SERVICE_QUOTAS
Service Catalog	se at	AWS_ENDPOINT_URL_SERVICE_CATALOG
Service Catalog AppRegistry	se at p:	AWS_ENDPOINT_URL_SERVICE_CATALOG_APP REGISTRY
ServiceDiscovery	se so	AWS_ENDPOINT_URL_SERVICEDISCOVERY
SES	se	AWS_ENDPOINT_URL_SES
SESv2	se	AWS_ENDPOINT_URL_SESV2
Shield	sh	AWS_ENDPOINT_URL_SHIELD
signer	s:	AWS_ENDPOINT_URL_SIGNER

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
SimSpaceWeaver	s: ea	AWS_ENDPOINT_URL_SIMSPACEWEAVER
SMS	sr	AWS_ENDPOINT_URL_SMS
Snow Device Managemen t	sr ce me	AWS_ENDPOINT_URL_SNOW_DEVICE_MANAGEMENT
Snowball	sr	AWS_ENDPOINT_URL_SNOWBALL
SNS	sr	AWS_ENDPOINT_URL_SNS
SQS	sc	AWS_ENDPOINT_URL_SQS
SSM	ss	AWS_ENDPOINT_URL_SSM
SSM Contacts	ss ct	AWS_ENDPOINT_URL_SSM_CONTACTS
SSM Incidents	ss er	AWS_ENDPOINT_URL_SSM_INCIDENTS
Ssm Sap	ss	AWS_ENDPOINT_URL_SSM_SAP

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
SSO	ss	AWS_ENDPOINT_URL_SSO
SSO Admin	ss	AWS_ENDPOINT_URL_SSO_ADMIN
SSO OIDC	ss	AWS_ENDPOINT_URL_SSO_OIDC
SFN	st	AWS_ENDPOINT_URL_SFN
Storage Gateway	st at	AWS_ENDPOINT_URL_STORAGE_GATEWAY
STS	st	AWS_ENDPOINT_URL_STS
SupplyChain	su in	AWS_ENDPOINT_URL_SUPPLYCHAIN
Support	su	AWS_ENDPOINT_URL_SUPPORT
Support App	su pp	AWS_ENDPOINT_URL_SUPPORT_APP
SWF	sw	AWS_ENDPOINT_URL_SWF

serviceId	共有AWS**co**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
synthetics	sy s	AWS_ENDPOINT_URL_SYNTHETICS
Textract	te	AWS_ENDPOINT_URL_TEXTRACT
Timestream InfluxDB	t: m_ b	AWS_ENDPOINT_URL_TIMESTREAM_INFLUXDB
Timestream Query	t: m_	AWS_ENDPOINT_URL_TIMESTREAM_QUERY
Timestream Write	t: m_	AWS_ENDPOINT_URL_TIMESTREAM_WRITE
tnb	tr	AWS_ENDPOINT_URL_TNB
Transcribe	t: e	AWS_ENDPOINT_URL_TRANSCRIBE
Transfer	t:	AWS_ENDPOINT_URL_TRANSFER

serviceId	共有AWS**ce**イルのサービス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Translate	t	AWS_ENDPOINT_URL_TRANSLATE
TrustedAdvisor	t v	AWS_ENDPOINT_URL_TRUSTEDADVISOR
VerifiedPermissions	ve e s	AWS_ENDPOINT_URL_VERIFIEDPERMISSIONS
Voice ID	vo	AWS_ENDPOINT_URL_VOICE_ID
VPC Lattice	vp ce	AWS_ENDPOINT_URL_VPC_LATTICE
WAF	wa	AWS_ENDPOINT_URL_WAF
WAF Regional	wa na	AWS_ENDPOINT_URL_WAF_REGIONAL
WAFV2	wa	AWS_ENDPOINT_URL_WAFV2

serviceId	共有AWSconfigファイルのサービス識別子キー	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
WellArchitected	well-architected	AWS_ENDPOINT_URL_WELLARCHITECTED
Wisdom	wisdom	AWS_ENDPOINT_URL_WISDOM
WorkDocs	workdocs	AWS_ENDPOINT_URL_WORKDOCS
WorkLink	worklink	AWS_ENDPOINT_URL_WORKLINK
WorkMail	workmail	AWS_ENDPOINT_URL_WORKMAIL
WorkMailMessageFlow	workmailmessageflow	AWS_ENDPOINT_URL_WORKMAILMESSAGEFLOW
WorkSpaces	workspaces	AWS_ENDPOINT_URL_WORKSPACES
WorkSpaces Thin Client	workspaces-thin-client	AWS_ENDPOINT_URL_WORKSPACES_THIN_CLIENT

serviceId	共有AWS**co**ィルのサビス識別子キ	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
WorkSpaces Web	wo s_	AWS_ENDPOINT_URL_WORKSPACES_WEB
XRay	x:	AWS_ENDPOINT_URL_XRAY

スマート設定デフォルト

スマート構成デフォルト機能を使用すると、 AWS SDK は他の構成設定に対して事前定義され最適化されたデフォルト値を提供できます。

この機能を設定するには、以下のように使用します。

defaults_mode- AWS **config** 共有ファイル設定, **AWS_DEFAULTS_MODE** - 環境変数, **aws.defaultsMode**-JVM システムプロパティ:Java/Kotlin のみ

この設定では、アプリケーションアーキテクチャに合ったモードを選択できます。これにより、アプリケーションに最適なデフォルト値が使用できるようになります。 AWS SDK 設定に明示的に値が設定されている場合は、常にその値が優先されます。 AWS SDK 設定の値が明示的に設定されておらず、レガシー設定と等しくない場合、defaults_modeこの機能はアプリケーションに最適化されたさまざまな設定に異なるデフォルト値を提供することができます。設定には、HTTP 通信設定、再試行動作、サービスの地域エンドポイント設定、および SDK 関連のあらゆる設定が含まれる可能性があります。この機能を使用するお客様は、一般的な使用シナリオに

合わせた新しいデフォルト設定を取得できます。defaults_mode が legacy と等しくない場合は、SDK をアップグレードするときにアプリケーションのテストを行うことをおすすめします。これは、ベストプラクティスの進化によってこの機能のデフォルト値が変わる可能性があるためです。

デフォルト値: legacy

注意：SDK の新しいメジャーバージョンでのデフォルトは standard になります。

有効値:

- legacy – SDK によって異なり、defaults_mode が確立される前から存在していたデフォルト設定を使用します。
- standard – ほとんどのシナリオで安全に実行できる最新の推奨デフォルト値を使用します。
- in-region— 標準モードをベースとしており、 AWS のサービス 同じモード内から呼び出しを行うアプリケーションに合わせた最適化が含まれています AWS リージョン。
- cross-region— 標準モードをベースとしており、 AWS のサービス 異なるリージョンで呼び出しを行うアプリケーションに合わせた最適化が含まれています。
- mobile – 標準モードに基づいて構築されており、モバイルアプリケーションに合わせた最適化が含まれています。
- auto – 標準モードに基づいて構築されており、実験的な機能が含まれています。SDK はランタイム環境を検出して適切な設定を自動的に決定しようとします。自動検出はヒューリスティックに基づいてあり、100% の精度は得られません。ランタイム環境を特定できない場合は、standard モードが使用されます。自動検出では、インスタンスのメタデータとユーザーデータをクエリすることがあり、レイテンシーが発生する可能性があります。起動時のレイテンシーがアプリケーションにとって最重要な場合は、代わりに明示的な defaults_mode を選択することをおすすめします。

config ファイルにこの値を設定する例を以下に示します。

```
[default]
defaults_mode = standard
```

以下のパラメータは、defaults_mode の選択に基づいて最適化される可能性があります。

- retryMode – SDK が再試行を試みる方法を指定します。再試行動作 を参照してください。
- stsRegionalEndpoints— AWS Security Token Service (AWS STS) AWS のサービス との通信に使用するエンドポイントを SDK がどのように決定するかを指定します。AWS STS 地域化されたエンドポイント を参照してください。

- `s3UsEast1RegionalEndpoints`— SDK `us-east-1` がリージョンの Amazon S3 AWS との通信に使用するサービスエンドポイントを決定する方法を指定します。
- `connectTimeoutInMillis` – ソケットで初めて接続を試みた後、タイムアウトするまでの時間。クライアントが接続ハンドシェイクの完了を受け取らない場合、クライアントは断念しオペレーションは失敗します。
- `tlsNegotiationTimeoutInMillis` – CLIENT HELLO メッセージが送信されてから、クライアントとサーバーが暗号を完全にネゴシエートしてキーを交換するまでの TLS ハンドシェイクにかかる最大時間。

各設定のデフォルト値は、アプリケーションで選択した `defaults_mode` によって異なります。これらの値は、現在以下のように設定されています（変更される可能性があります）。

パラメータ	**standard** モード	**in-region** モード	**cross-region** モード	**mobile** モード
`retryMode`	`standard`	`standard`	`standard`	`standard`
`stsRegionalEndpoints`	`regional`	`regional`	`regional`	`regional`
`s3UsEast1RegionalEndpoints`	`regional`	`regional`	`regional`	`regional`
`connectTimeoutInMillis`	3100	1100	3100	30000
`tlsNegotiationTimeoutInMillis`	3100	1100	3100	30000

たとえば、選択した defaults_mode が standard の場合、standard の値が（有効な retry_mode オプションから）retry_mode に割り当てられ、regional の値が（有効な stsRegionalEndpoints オプションから）stsRegionalEndpoints に割り当てられます。

SDK AWS との互換性

以下の SDK は、このトピックで説明する機能と設定をサポートします。部分的な例外があれば、すべて記載されています。すべての JVM システムプロパティ設定は、 AWS SDK for Java とでのみサポートされます。 AWS SDK for Kotlin

SDK	サポート	注意または詳細情報
AWS CLI v2	No	
SDK for C++	Yes	最適化されていないパラメーター：stsRegionalEndpoints 、s3UsEast1RegionalEndpoints 、tlsNegotiationTimeoutInMillis 。
SDK for Go V2 (1.x)	Yes	最適化されていないパラメーター：retryMode 、stsRegionalEndpoints 、s3UsEast1RegionalEndpoints 。
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	Yes	最適化されていないパラメーター：stsRegionalEndpoints 。
SDK for Java 1.x	No	
3.x JavaScript 用の SDK	Yes	最適化されていないパラメーター：stsRegion

SDK	サポート	注意または詳細情報
		alEndpoints 、s3UsEast1RegionalEndpoints 、tlsNegotiationTimeoutInMillis 。connectTimeoutInMillis は connectionTimeout と呼ばれます。
2.x JavaScript 用の SDK	No	
SDK for Kotlin	No	
SDK for .NET 3.x	Yes	最適化されていないパラメーター：connectTimeoutInMillis 、tlsNegotiationTimeoutInMillis 。
SDK for PHP 3.x	Yes	最適化されていないパラメーター：tlsNegotiationTimeoutInMillis 。
SDK for Python (Boto3)	Yes	最適化されていないパラメーター：tlsNegotiationTimeoutInMillis 。
SDK for Ruby 3.x	Yes	
SDK for Rust	No	

SDK	サポート	注意または詳細情報
以下のためのツール PowerShell	Yes	最適化されていないパラメーター：connectTimeoutInMillis 、tlsNegotiationTimeoutInMillis 。

AWS Common Runtime (CRT) ライブラリ

AWS Common Runtime (CRT) ライブラリは SDK の基本ライブラリです。CRT は C で書かれた独立パッケージのモジュラーファミリーで、各パッケージはパフォーマンスが高く、必要なさまざまな機能にフットプリントを最小限に抑えます。これらの機能はすべての SDK に共通で共有しているため、コードの再利用、最適化、精度が向上します。パッケージは以下のとおりです。

- awslabs/aws-c-auth：AWS クライアント側認証 (標準認証情報プロバイダーと署名 (sigv4))
- awslabs/aws-c-cal：暗号プリミティブ型、ハッシュ (MD5、SHA256、SHA256 HMAC)、署名者、AES
- awslabs/aws-c-common：基本データ構造、スレッド / 同期プリミティブ型、バッファ管理、stdlib 関連関数
- awslabs/aws-c-compression：圧縮アルゴリズム (ハフマンエンコーディング / デコーディング)
- awslabs/aws-c-event-stream：イベントストリームメッセージ処理 (ヘッダー、プレリュード、ペイロード、crc/トレーラー) 、イベントストリーム経由のリモートプロシージャ呼び出し (RPC) 実装
- awslabs/aws-c-http：C99 による、HTTP/1.1 仕様と、HTTP/2 仕様の実装
- awslabs/aws-c-io：ソケット (TCP、UDP)、DNS、パイプ、イベントループ、チャネル、SSL/TLS
- awslabs/aws-c-iot：C99 による、AWS IoT クラウドサービスのデバイスとの統合の実装
- awslabs/aws-c-mqtt：モノのインターネット (IoT) 向けの標準の軽量メッセージングプロトコル
- awslabs/aws-c-s3：Amazon S3 サービスと通信するための C99 ライブラリ実装。高帯域幅の Amazon EC2 インスタンスでスループットを最大化するように設計されています
- awslabs/aws-c-sdkutils：AWS プロファイルを解析および管理するためのユーティリティライブラリ
- awslabs/aws-checksums：効率的なソフトウェア実装へのフォールバック機能を備えた、クロスプラットフォームのハードウェア加速化による CRC32c と CRC32
- awslabs/aws-lc：Google BoringSSL プロジェクトと OpenSSL プロジェクトのコードに基づいて、AWS とその顧客向けに AWS 暗号チームが管理している汎用暗号ライブラリ
- awslabs/s2n：C99 による TLS/SSL プロトコルの 実装。セキュリティを優先して小型かつ高速に動作するように設計

CRT は Go を除くすべての SDK で使用できます。

CRT の依存関係

CRT ライブラリは複雑な関係と依存関係を形成しています。これらの関係を知っておくと、CRT をソースから直接構築する必要がある場合に役立ちます。ただし、ほとんどのユーザーは、自分の言語 SDK (C++ の場合は AWS SDK、Java の場合は AWS SDK など) または自分の言語の IoT デバイス SDK (C++ の場合は AWS IoT SDK、Java の場合は AWS IoT SDK など) を使用して CRT 機能にアクセスします。以下の図の「言語 CRT バインディング」ボックスは、特定の言語 SDK の CRT ライブラリをラップするパッケージを示しています。これは `aws-crt-*` 形式のパッケージの集まりで、「`*`」は SDK 言語 (`aws-crt-cpp` や `aws-crt-java` など) です。

CRT ライブラリの階層的な依存関係を以下に示します。

AWS SDKsメンテナンスポリシー

概要

このドキュメントでは、モバイルおよび IoT SDKsを含む AWS Software Development Kits (SDKs とツールのメンテナンスポリシー、およびそれらの基盤となる依存関係の概要を説明します。 AWS は、新規または更新された AWS APIsのサポート、新機能、機能強化、バグ修正、セキュリティパッチ、ドキュメントの更新を含む可能性のある更新を AWS SDKs とツールに定期的に提供します。更新では、依存関係、言語ランタイム、オペレーティングシステムの変更にも対処できます。 AWS SDK リリースはパッケージマネージャー (Maven、PyPI など) に公開され NuGet、 でソースコードとして利用できます GitHub。

最新の機能、セキュリティアップデート、および基盤となる依存関係に遅れないように、 up-to-date SDK リリースを引き続き使用することをお勧めします。サポート対象外の SDK バージョンを継続して使用することはお勧めできません。ユーザーの判断で行ってください。

バージョニング

AWS SDK リリースバージョンは X.Y.Z の形式で、X はメジャーバージョンを表します。SDK のメジャーバージョンを増やすということは、その SDK がその言語の新しいイディオムやパターンをサポートするために大幅に変更されたことを意味します。メジャーバージョンは、パブリックインターフェイス (クラス、メソッド、タイプなど) 、動作、またはセマンティクスが変更された時点で導入されます。アプリケーションを最新の SDK バージョンで動作させるには、更新する必要があります。メジャーバージョンは、 AWSに記載されているアップグレードガイドラインに従って慎重に更新することが重要です。

SDK メジャーバージョンのライフサイクル

メジャー SDKs およびツールバージョンのライフサイクルは、以下に概説する 5 つのフェーズで構成されます。

- 開発者プレビュー (フェーズ 0) -このフェーズでは SDK のサポートはされないため、本番環境では使用できません。また、SDK は早期アクセスとフィードバックのみを目的としています。今後のリリースでは、非互換性の変更が導入される可能性があります。がリリースを安定した製品として AWS 識別すると、リリース候補としてマークする場合があります。リリース候補は、重大なバ

グが発生しない限り GA リリースの準備ができており、フル AWS サポートを受けることができます。

- 一般提供 (GA) (フェーズ 1) - このフェーズでは、SDKs が完全にサポートされています。 AWS は、新しい サービスのサポート、既存の サービスの API 更新、バグとセキュリティの修正を含む定期的な SDK リリースを提供します。ツールの場合、 AWS は新機能の更新とバグ修正を含む定期的なリリースを提供します。 AWS は、少なくとも 24 か月間、 SDK の GA バージョンをサポートします。
- メンテナンスのお知らせ (フェーズ 2) - AWS SDK がメンテナンスモードになる少なくとも 6 か月前に、パブリックなお知らせを行います。この期間中、SDK は引き続き完全にサポートされます。通常、メンテナンスモードは次のメジャーバージョンが GA に移行されると同時に発表されます。
- メンテナンス (フェーズ 3) -メンテナンスモードでは、 AWS は重大なバグ修正とセキュリティ問題のみに対処するよう SDK のリリースを制限します。SDK は、新規または既存のサービスの API 更新を受け取ることも、新しいリージョンをサポートするように更新されることもありません。特に指定がない限り、メンテナンスモードのデフォルト期間は 12 か月です。
- サポート終了 (フェーズ 4) - : SDK がサポート終了になると、更新やリリースは受け取れなくなります。以前に公開されたリリースは、引き続きパブリックパッケージマネージャーを通じて利用可能になり、コードは に残ります GitHub。 GitHub リポジトリはアーカイブできます。に達した SDK の使用は end-of-support 、ユーザーの裁量で行われます。ユーザーには新しいメジャーバージョンへのアップグレードをお勧めします。

以下は、SDK メジャーバージョンのライフサイクルの視覚的な図です。以下に示すタイムラインは例示であり、拘束力はないことに注意してください。

依存関係のライフサイクル

ほとんどの AWS SDKsには、言語ランタイム、オペレーティングシステム、サードパーティーのライブラリやフレームワークなど、基盤となる依存関係があります。これらの依存関係は、通常、言語コミュニティや特定のコンポーネントを所有するベンダーに関係しています。各コミュニティまたはベンダーは、製品の独自の end-of-support スケジュールを公開します。

基礎となるサードパーティの依存関係を分類するには、以下の用語が使用されます。

- オペレーティングシステム (OS)：例としては、Amazon Linux AMI、Amazon Linux 2、Windows 2008、Windows 2012、Windows 2016 などがあります。

- 言語ランタイム：例としては、Java 7、Java 8、Java 11、.NET Core、.NET Standard、.NET PCL などがあります。
- サードパーティのライブラリ / フレームワーク：例としては、OpenSSL、.NET Framework 4.5、Java EE などがあります。

コミュニティまたはベンダーが依存関係のサポートを終了した後も、少なくとも 6 か月間は SDK の依存関係をサポートし続けることが当社の方針です。ただし、このポリシーは、特定の依存関係によって異なる場合があります。

> **Note**
>
> AWS は、SDK のメジャーバージョンを増やすことなく、基盤となる依存関係のサポートを停止する権利を保持します。

コミュニケーションの方法

メンテナンスのお知らせは、以下のように伝えられます。

- 該当するアカウントには、特定の SDK バージョンのサポートを終了する計画を知らせる E メールが送信されます。E メールには end-of-support、 へのパスの概要、キャンペーンのタイムラインの指定、アップグレードガイダンスが記載されています。
- AWS API リファレンスドキュメント、ユーザーガイド、SDK 製品マーケティングページ、GitHub readme (s) などの SDK ドキュメントが更新され、キャンペーンのタイムラインが示され、影響を受けるアプリケーションのアップグレードに関するガイダンスが提供されます。
- へのパスを概説し end-of-support、キャンペーンのタイムラインを繰り返す AWS ブログ記事が公開されます。
- 非推奨の警告が SDKsドキュメントへのパス end-of-support とリンクが概説されます。

AWS SDKs「」を参照してください[バージョンのサポートマトリックス](#)。

AWS SDKsとツールのバージョンサポートマトリックス

以下のマトリックスは、利用可能な AWS Software Development Kit (SDK) メジャーバージョンのリストと、それらがメンテナンスライフサイクルのどの段階にあるか、関連するタイムラインを示しています。 AWS SDKs「」を参照してください[メンテナンスポリシー](#)。

SDK	メジャーバージョン	現在のフェーズ	一般提供日	メモ
AWS CLI	1.x	一般提供	9/2/2013	
AWS CLI	2.x	一般提供	2/10/2020	
SDK for C++	1.x	一般提供	9/2/2015	
SDK for Go V2	V2 1.x	一般提供	1/19/2021	
SDK for Go	1.x	メンテナンスのお知らせ	11/19/2015	詳細と日付については、「お知らせ」を参照してください。
SDK for Java	1.x	メンテナンスのお知らせ	3/25/2010	詳細と日付については、「お知らせ」を参照してください。
SDK for Java	2.x	一般提供	2018 年 11 月 20 日	
の SDK JavaScript	1.x	サポートの終了	5/6/2013	
の SDK JavaScript	2.x	メンテナンスのお知らせ	6/19/2014	詳細と日付については、「お知らせ」を参照してください。

SDK	メジャーバージョン	現在のフェーズ	一般提供日	メモ
の SDK JavaScript	3.x	一般提供	12/15/2020	
SDK for Kotlin	1.x	一般提供	11/27/2023	
SDK for .NET	1.x	サポートの終了	2009 年 11 月	
SDK for .NET	2.x	サポートの終了	11/8/2013	
SDK for .NET	3.x	一般提供	7/28/2015	
SDK for PHP	2.x	サポートの終了	11/2/2012	
SDK for PHP	3.x	一般提供	5/27/2015	
SDK for Python (Boto2)	1.x	サポートの終了	7/13/2011	
SDK for Python (Boto3)	1.x	一般提供	6/22/2015	
SDK for Python (Botocore)	1.x	一般提供	6/22/2015	
SDK for Ruby	1.x	サポートの終了	7/14/2011	
SDK for Ruby	2.x	サポートの終了	2/15/2015	
SDK for Ruby	3.x	一般提供	8/29/2017	
SDK for Rust	1.x	一般提供	11/27/2023	
SDK for Swift	1.x	開発者プレビュー		
のツール PowerShell	2.x	サポートの終了	11/8/2013	

SDK	メジャーバージョン	現在のフェーズ	一般提供日	メモ
のツール PowerShell	3.x	サポートの終了	7/29/2015	
のツール PowerShell	4.x	一般提供	11/21/2019	

AWS SDKsドキュメント履歴

次の表は、「AWS SDK およびツールリファレンスガイド」への重要な追加と更新をまとめたものです。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

変更	説明	日付
SDK for Java 1.x システムプロパティ	AWS SDK for Java 1.x でサポートされている JVM システム構成設定の詳細を追加します。	2024 年 5 月 30 日
設定の更新	JVM システム構成設定を追加します。	2024 年 3 月 27 日
互換性テーブルの更新	SDK サポートの互換性の更新、IAM Identity Center の手順の更新。	2024 年 2 月 20 日
コンテナ認証情報の更新。IMDS の更新。	Amazon EKS 向けのサポートを追加しました。IMDSv1 フォールバックを無効にする設定を追加しました。	2023 年 12 月 29 日
リクエスト圧縮	リクエスト圧縮機能の設定を追加しました。	2023 年 12 月 27 日
互換性に関する表	SDK とツールの機能に関する互換性テーブルが更新され、SDK for Kotlin、SDK for Rust、および AWS Tools for PowerShellが含まれるようになりました。	2023 年 12 月 10 日
認証の更新	SDK およびツールでサポートされている認証方法を更新しました。	2023 年 7 月 1 日

IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM 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。	2023 年 2 月 27 日
SSO の更新	新しい SSO トークン設定の SSO 認証情報を更新しました。	2022 年 11 月 19 日
設定の更新	一般設定と Amazon S3 マルチリージョンアクセスポイントのサポート表を更新しました。	2022 年 11 月 17 日
設定の更新	IMDS クライアントと IMDS 認証情報が明確になるように更新しました。環境変数を更新しました。	2022 年 11 月 4 日
ウェルカムページを更新	Amazon の発表 CodeWhisperer。	2022 年 9 月 22 日
シングルサインオンのサービス名の変更	AWS SSO が と呼ばれるようになったことを反映する更新 AWS IAM Identity Center。	2022 年 7 月 26 日
設定の更新	設定ファイルの詳細とサポートされる設定のマイナーな更新。	2022 年 6 月 15 日
更新	本ガイドのほぼすべての部分を大幅に更新。	2022 年 2 月 1 日
初回リリース	このガイドの最初のリリースが一般に公開されています。	2020 年 3 月 13 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

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。