aws

リファレンスガイド

AWS SDKs および ツール

AWS SDKs および ツール: リファレンスガイド

Table of Contents

AWS SDKs および ツールリファレンスガイド

多くの SDKsおよびツールは、共有設計仕様または共有ライブラリを通じて、いくつかの一般的な機能を共有します。

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 および ツールを設定する AWS SDKs方法。
- 認証とアクセス – で開発 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 共通ランタイム (CRT) ライブラリ - ほぼすべての 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SDKs。
- AWS SDKsメンテナンスポリシー では、モバイルおよびモノのインターネット (IoTSDKs) を含む Software Development Kit () とツールのメンテナンスポリシーとバージョニング AWS 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。

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。ここに示されている情報に加えて、使用している SDKまたはツールの特定のガイドを使用する必要があります。以下に示しているのは、このガイドの関連資料のセクションを含む SDKおよび ツールです。

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。
• SDK または ツール	AWS SDKsメンテナンスポリシー
• AWS Cloud9 • AWS CDK • AWS Toolkit for Azure DevOps • AWS Toolkit for JetBrains • AWS Toolkit for Visual Studio • AWS Toolkit for Visual Studio Code	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 Serverless Application Model • AWS CodeArtifact • AWS CodeBuild • Amazon CodeCatalyst • AWS CodeCommit • AWS CodeDeploy • AWS CodePipeline	
• AWS CLI • AWS SDK for C++ • AWS SDK for Go • AWS SDK for Java • AWS SDK for JavaScript • AWS SDK for Kotlin • AWS SDK for .NET • AWS SDK for PHP • AWS SDK for Python (Boto3) • AWS SDK for Ruby • AWS SDK for Rust • AWS SDK for Swift • AWS Tools for Windows PowerShell	構成 認証とアクセス 設定リファレンス AWS 共通ランタイム (CRT) ライブラリ AWS SDKsメンテナンスポリシー AWS SDKs およびツールバージョンのサポート

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、「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あり、 AWS アプリケーションの理解、構築、拡張、運用に役立ちます。構築を加速するために 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 コンテンツが強化されてい

ます。詳細については、「Amazon Q Developer ユーザーガイド」の「Amazon Q Developer 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、 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IDE。Amazon Q IDEプラグインと拡張機能により、 で生成 AI 支援が可能になりますIDE。各IDEツールキットの詳細については、前の表の Toolkit ユーザーガイドを参照してください。での Amazon Q の使用の詳細についてはIDE、Amazon Q デベロッパーガイドのIDEトピックの「Amazon Q の使用」を参照してください。

AWS IDE ツールキットと Amazon Q は、今後の AWS Toolkit と Amazon Q の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ことがあります。収集されたデータは、 AWS Toolkit と Amazon Q の使用状況を定量化します。

すべての AWS IDE Toolkits と Amazon Q 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 Toolkits および Amazon Q 拡張機能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 Toolkit GitHub リポジトリのリソースドキュメントを参照してください。

- AWS Amazon Q を使用した Visual Studio Toolkit
- AWS Toolkit for Visual Studio Code VS Code 用の および Amazon Q 拡張機能
- AWS Toolkit for JetBrains および 用の Amazon Q プラグイン JetBrains
- Amazon Q for Eclipse

AWS Toolkit でアクセスできる特定の AWS サービスは、追加のクライアント側のテレメトリデータを収集する場合があります。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、`config`と`credentials`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 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 ）、、およびその他のツールで使用されるキー AWS SDKsと値のペアの一連の設定です。設定値は、プロファイルが使用されるときに SDK/tool の一部を設定するためにプロファイルにアタッチされます。これらのファイルは、値がアプリケーション、プロセス、またはユーザーのローカル環境に影響するという点でSDKs「共有」されます。

共有 credentials ファイルconfigと ファイルの両方が、ASCII文字 (UTF-8 エンコード) のみを含むプレーンテキストファイルです。これらは、一般的に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。このプロファイルの名前は、ユーザーがプロファイルを指定しない場合に で使用されるデフォルトのプロファイルdefaultであるためSDKです。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、 を操作する場合は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

この環境変数が名前付きプロファイルまたは「デフォルト」に設定されている場合、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

の SDK for Kotlin JVM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。は、サービスクライアントSDK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。

> **Note**
>
> アプリケーションが複数のアプリケーションを実行しているサーバー上にある場合は、デフォルトのプロファイルではなく、常に名前付きプロファイルを使用することをお勧めします。デフォルトのプロファイルは、環境内のすべての AWS アプリケーションによって自動的に取得され、それらの間で共有されます。したがって、他のユーザーがアプリケーションのデフォルトプロファイルを更新すると、他のユーザーに意図しない影響を与える可能性があります。これを防ぐには、共有configファイルに名前付きプロファイルを定義し、コードに名前付きプロファイルを設定して、アプリケーションでその名前付きプロファイルを使

用します。環境変数またはJVMシステムプロパティを使用して、スコープがアプリケーションにのみ影響することがわかっている場合は、名前付きプロファイルを設定できます。

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発生するまで、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の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 SDKまたは ツールの 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-ssoSampleRole」のトークンを使用して、「111122223333」アカウントの IAM「」ロールの短期 AWS 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**
>
> このセクションでは、サービス固有のエンドポイントのカスタマイズservicesをサポートしており、この機能を含む SDKsおよび ツールでのみ使用できます。でこの機能が使用可能かどうかを確認するにはSDK、サービス固有のエンドポイントとの互換性 AWS SDKsについては、「」を参照してください。

services セクション定義行は、 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の名前を付けます。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 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。角括弧の間にプロファイル名自体のみを使用します。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...
```

credentials 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、aws_access_key_idaws_secret_access_key、および のみです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。aws_access_key_id、aws_secret_access_key、および aws_session_token グローバル設定を設定します。

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

credentials ファイルで名前付きプロファイルを使用するかdefault「」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ファイルの設定と結合されます。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
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 でサポートされている環境変数のリストについてはSDKs、「」を参照してください環境変数の一覧。

オプションの優先順位

- 環境変数を使用して設定を指定すると、共有 AWS `config` ファイルと `credentials` ファイル内のプロファイルからロードされた値が上書きされます。
- 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すると、対応する環境変数または設定ファイルのプロファイルのいずれか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...zrkuWJOgQs8IZZaIv2BXIa2R4Ol
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すると、現在のセッションの期間のみ値が保存されます。環境変数設定をすべての PowerShell および コマンドプロンプトセッションで永続化するには、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保存します。または、 変数を PowerShell プロファイルに追加することで、以降のすべての PowerShell セッションに設定できます。環境変数の保存またはセッション間での永続化の詳細については、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。

例えば、 を使用すると AWS Lambda、環境変数を直接設定できます。詳細については、「 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。 AWS Lambda

Serverless Framework では、多くの場合、SDK環境設定のプロバイダーキーの下の `serverless.yml` ファイルに環境変数を設定できます。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

を使用して開発 AWS するときに、コードが で認証される方法を確立する必要があります AWS のサービス。 AWS リソースへのプログラムによるアクセスは、環境や利用可能な AWS アクセスに応じて、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 SDKまたは ツールの Identity Center 認証 – セキュリティのベストプラクティスとして、 を IAM Identity Center AWS Organizations と共に使用して、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。でユーザーを作成したり AWS IAM Identity Center、Microsoft Active Directory を使用したり、2.0 ID SAML プロバイダー (IdP) を使用したり、IdP を に個別にフェデレーションしたりできます AWS アカウント。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、 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Amazon Web Services 全般のリファレンス。
- IAM Roles Anywhere – IAM Roles Anywhere を使用して、 の外部で実行されるサーバー、コンテナ、アプリケーションなどのワークロードIAMの で一時的なセキュリティ認証情報を取得できます AWS。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。
- AWS 認証情報を持つロールを想定する – それ以外の場合にアクセスできない AWS リソースに一時的にアクセスするIAMロールを引き受けることができます。
- AWS アクセスキー – 利便性が低い、または AWS リソースのセキュリティリスクを高める可能性が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

コードが で実行されている場合 AWS、認証情報をアプリケーションに自動的に使用できます。例えば、アプリケーションが Amazon Elastic Compute Cloud でホストされていて、そのリソースに関連付けられたIAMロールがある場合、認証情報はアプリケーションに自動的に利用可能になります。同様に、Amazon ECSまたは Amazon EKSコンテナを使用する場合、IAMロールの認証情報セットは、SDKの認証情報プロバイダーチェーンを介してコンテナ内で実行されるコードによって自動的に取得できます。

- Amazon EC2インスタンスでの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で とやり取りするには、次の方法があります。

- AWS CloudShell を使用して、コンソールから AWS CLI コマンドを実行します。
- ソフトウェア開発チーム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の使用を検討してください。

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

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 AWS、ウェブ ID フェデレーションを使用して一時的な AWS セキュリティ認証情報を動的にリクエストするようにアプリケーションを構築します。

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。代わりに、アプリケーションユーザーは、Amazon、Facebook、Google、またはその他の OpenID Connect () 互換 IdP による口グインなど、有名な外部 ID プロバイダー (IdP OIDC) を使用してサインインできます。認証トークンを受信し、そのトークンを AWS の一時的なセキュリティ認証情報と交換して、 のリソースを使用するアクセス許可を持つIAMロールにマッピングできます AWS アカウント。

これを SDKまたは ツールに設定する方法については、「」を参照してくださいウェブ ID または OpenID Connect を使用してロールを引き受ける。

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「モバイルアプリでの 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 リソースへのアクセスを安全に制御するための以下の情報が記載されています。

- IAM ID (ユーザー、ユーザーグループ、ロール) — の ID の基本を理解します AWS。
- におけるセキュリティのベストプラクティス IAM – 責任共有モデル に従って AWS アプリケーションを開発するときに従うべきセキュリティの推奨事項。

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のキーオプションと管理プラクティスにアクセスします。

AWS ビルダー ID

は、既に所有 AWS アカウント している、または作成したいものをすべて AWS ビルダー ID 補完します。は、作成した AWS リソースのコンテナ AWS アカウント として機能し、それらのリソースのセキュリティ境界を提供しますが、 は個人としてユーザー AWS ビルダー ID を表します。でサインイン AWS ビルダー ID すると、Amazon や Amazon などの開発者ツール CodeWhisperer やサービスにアクセスできます CodeCatalyst。

- AWS サインイン ユーザーガイドの でサインイン AWS ビルダー IDする – を作成して使用する方法 AWS ビルダー ID と、ビルダー ID が提供する内容について説明します。
- CodeWhisperer と による認証 AWS Toolkit - ユーザーガイドのビルダー ID – が CodeWhisperer を使用する方法を説明します AWS ビルダー ID。 CodeWhisperer
- Amazon CodeCatalyst ユーザーガイド CodeCatalyst の 概念 AWS ビルダー ID - で、 がどのように CodeCatalyst を使用するかを説明します AWS ビルダー ID。

IAM SDKまたは ツールの Identity Center 認証

AWS IAM Identity Center は、AWS 非コンピューティング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 リソースで開発している場合は 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、IAMIdentity 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、IAMIdentity Center に手動でサインインするウェブの場所です。の形式は `d-xxxxxxxxxx.awsapps.com/start`または URLです*`your_subdomain`*`.awsapps.com/start`。 AWS アクセスポータルにサインインすると、そのユーザー用に設定された AWS アカウント および 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/tool 認証プロセスに必要な設定値を取得します。
- AWS CLI は、コードによるAPI呼び出しに 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する

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、AWS IAM Identity Center 「 ユーザーガイド」のIAM「Identity Center の有効化」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 経由でアクセスを確立していない

1. AWS IAM Identity Center ユーザーガイドのデフォルトの IAM Identity Center ディレクトリでユーザーアクセスを設定の手順に従って、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 ディレクトリ手順でユーザーアクセスを設定するをもう一度実行しますが、今回は以下を実行します。

   - *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、この手順でこれを置き換えます。
   - 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、この手順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。ツールを使用するPowerUserAccessときに を選択します/SDK。

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。クラウド管理者がユーザー 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 AWS 管理する アクセスポータル AWS から に既にアクセスしている

AWS アクセスポータル AWS から にサインインします。クラウド管理者から PowerUserAccess (開発者) アクセス許可を付与されている場合は、アクセス権のある 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 にすでにアクセスしている

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 SDKsとツールを設定する

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。雇用主がこれを提供するURLか、「ステップ 1: アクセスを確立する」の後に E メールで受け取ることができます。そうでない場合は、 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https://console.aws.amazon.com/singlesignon/。

   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 `Issuer URL`と `SSO Region`の値を取得します。注: `SSO Start URL`は と互換性がありま`Issuer URL`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。

a. プロンプト に`SSO Start URL`、 用に取得した値を入力します`Issuer URL`。

b. CLI プロファイル名には、開始*`default`*するときに 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ます。

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

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

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。SDK または ツールが IAM Identity Center の認証情報を使用する前に、 が認証情報プロバイダーチェーンを理解する優先されるため、静的認証情報を削除する必要があります。

SDKs および 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の詳細については、「」を参照してくださいIAM Identity Center 認証を理解する。

設定したセッションの長さに応じて、アクセスは最終的に期限切れになり、 SDKまたは ツール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、 AWS CLI を使用して `aws sso login` コマンドを実行します。

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

SDKs および ツールのすべての IAM Identity Center プロバイダー設定の詳細については、このガイドIAM Identity Center 認証情報プロバイダーの「」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		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
管理コンソール		
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の設定 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。

AWS 認証情報を持つロールを想定する

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、まず引き受ける特定のロールを作成または識別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 () によって一意に識別されますARN。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 または別の である可能性があります AWS アカウント。IAM ロールの詳細については、「 ユーザーガイド」のIAM「ロールの使用」を参照してください。 IAM

IAM ロールが特定されたら、そのロールから信頼されている場合は、ロールによって付与されたアクセス許可を使用するように SDKまたは ツールを設定できます。

> **Note**
>
> 可能な限りリージョンエンドポイントを使用し、 を設定することが AWS ベストプラクティスですAWS リージョン。

IAM ロールを引き受ける

ロールを引き受けると、 は一連の一時的なセキュリティ認証情報 AWS STS 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。通常、このタイプの引き受けは、あるアカウントの AWS 認証情報を持っているが、アプリケーションが別のアカウントのリソースにアクセスする必要がある場合に使用されます。

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、まず引き受ける特定のロールを作成または識別する必要があります。IAM ロールは、ロール を使用して一意に識別されますARN。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。これを設定するには、「 ユーザーガイド」のIAM「ロールの作成」を参照してください。 IAM

ステップ 2: SDKまたは ツールを設定する

SDK または ツールを設定して、 `credential_source` または から認証情報を取得しますsource_profile。

`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ソースします。

`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。

プロファイルでこれを指定すると、 SDKまたは ツールは対応する AWS STS `AssumeRole`API呼び出しを自動的に実行します。ロールを引き受けて一時的な認証情報を取得して使用するには、共有 AWS config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 で作成したIAMロールから
- `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 ID または OpenID Connect を使用してロールを引き受ける

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、まず引き受ける特定のロールを作成または識別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) によっ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。ロールを使用する信頼されたエンティティは、ウェブ ID プロバイダー、OpenID Connect(OIDC) 、またはSAMLフェデレーションである場合があります。IAM ロールの詳細については、「 IAMユーザーガイド」の「ロールを引き受ける方法」を参照してください。

でIAMロールを設定した後SDK、そのロールが ID プロバイダーを信頼するように設定されている場合は、一時的な AWS 認証情報を取得SDKするために、そのロールを引き受けるように をさらに設定できます。

> **Note**
>
> 可能な限りリージョンエンドポイントを使用し、 を設定することが AWS ベストプラクティスですAWS リージョン。

ウェブアイデンティティまたは OpenID Connect でのフェデレーション

Login With Amazon、Facebook、Google などのパブリック ID プロバイダーのJSON ウェブトークン (JWTs) を使用して、 を使用して一時的な AWS 認証情報を取得できますAssumeRoleWithWebIdentity。使用方法によっては、ID トークンまたはアクセストークンと呼ばれるJWTs場合があります。 EntraId や など、 の検出プロトコルと互換性のある ID プロバイダー (IdPs) OIDCからJWTs発行された を使用することもできます PingFederate。

Amazon Elastic Kubernetes Service を使用している場合、この機能を使用すると、Amazon EKSクラスター内のサービスアカウントごとに異なるIAMロールを指定できます。この Kubernetes 機能はポッドJWTsに配信され、この認証情報プロバイダーはこの認証情報プロバイダーを使用して一時的な AWS 認証情報を取得します。この Amazon EKS設定の詳細については、「Amazon EKSユーザーガイドIAM」の「サービスアカウントのロール」を参照してください。ただし、より簡単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 Identity を使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 とのフェデレーションを設定するには、IAMID プロバイダーを使用して、外部 IdP とその設定 AWS について に通知します。これにより、 AWS アカウント と外部 IdP の間に信頼が確立されます。認証にJSONウェブトークン (JWT) を使用するSDKように 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に使用される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 ID または OpenID Connect フェデレーション (コンソール) のロールの作成」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

認証に からJSONウェブトークン (JWT) を使用するように SDKまたは ツールを設定します AWS STS 。

プロファイルでこれを指定すると、 SDKまたは ツールは対応する AWS STS `AssumeRoleWithWebIdentity`API呼び出しを自動的に行います。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して使用するには、共有 AWS `config`ファイルで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 で作成したIAMロールから
- `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SDKsおよび ツールのコアライブラリには含まれていません。Amazon Cognito にアクセスするにはAPI、 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ます。での使用については AWS SDKs、Amazon Cognito デベロッパーガイド」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 SDKまたは ツールの 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

拡張セッション期間オプションIAM SDKまたは ツールの Identity Center 認証で使用するように 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 アクセスポータルで使用できる一時的な認証情報をコピーして使用できます。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。

ベストプラクティス: 認証情報ファイルでアクセスキーとトークンを手動で管理する代わりに、アプリケーションは以下から配信される一時的な認証情報を使用することをお勧めします。

- Amazon Elastic Compute Cloud または でアプリケーションを実行するなどのコンピューティング AWS サービス AWS Lambda。
- 認証情報プロバイダーチェーンの別のオプション。 などIAM SDKまたは ツールの Identity Center 認証。
- またはプロセス認証情報プロバイダー、 を使用して一時的な認証情報を取得します。

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する

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、ファイル ~/.aws/credentials 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 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。 SDK
4. AWS アクセスポータルにサインインします。
5. アクセス AWS ポータルからIAMロールの認証情報をコピーするには、認証情報を手動で更新する手順に従ってください。

a. リンクされた手順のステップ 4 では、開発ニーズに合わせてアクセスを許可するIAMロール名を選択します。このロールには通常、 PowerUserAccessまたは Developer のような名前が付けられます。

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。

がサービスクライアントSDKを作成すると、これらの一時的な認証情報にアクセスし、リクエストごとに使用します。ステップ 5a で選択したIAMロールの設定によって、一時的な認証情報が有効である期間が決まります。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**
>
> セキュリティ上のリスクを回避するため、専用ソフトウェアの開発時や実際のデータの使用時に、認証にIAMユーザーを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。トピックのセキュリティのベストプラクティスIAMを確認し、できるだけ早く IAM Identity Center または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

- アカウントのルート認証情報NOTを使用して リソースにアクセスします AWS 。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。
- リテラルアクセスキーまたは認証情報をアプリケーションファイルNOTに入れます。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。
- プロジェクトエリアに認証情報を含むファイルNOTを含めます。
- 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) IAMタスクのタスクに ロールを使用します。
- Amazon EC2インスタンスで実行されているアプリケーションには 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「管理ユーザーの作成」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 AWS や でのアプリケーションの実行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 1: IAM ユーザーを作成する

- IAM 「 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ときに役立ちます。
  - 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として、ポリシーを直接アタッチを選択します。
    - 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、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. アクセスキーの作成ステップ 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 の両方で使用する同じタイプのキーを生成しますSDKs。
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 ファイルのダウンロードを選択して、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。これは使用を開始する方法ですが、できるだけ早く IAM Identity Center または他の一時的な認証情報に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インスタンスでのIAMロールの使用

この例では、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。 EC2

Amazon Elastic Compute Cloud インスタンスでアプリケーションを実行する AWS SDKには、 IAM ロールを作成し、Amazon EC2インスタンスにそのロールへのアクセス権を付与します。詳細については、IAM「Amazon ユーザーガイド」の「Amazon のロールEC2」を参照してください。 EC2

IAM ロールを作成する

開発するアプリケーションは AWS SDK、アクションを実行 AWS のサービス するために少なくとも 1 つにアクセスする可能性があります。アプリケーションの実行に必要なアクセス許可を付与する IAMロールを作成します。

この手順では、例として Amazon S3 への読み取り専用アクセスを許可するロールを作成します。ガイドの多くは AWS SDK、Amazon S3 から読み取る「開始方法」チュートリアルがあります。

1. にサインイン AWS Management Console し、 でIAMコンソールを開きますhttps://console.aws.amazon.com/iam/。
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. ユースケース で Amazon EC2を選択し、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。Amazon EC2インスタンスを作成するときに必要になるため、この名前を覚えておいてください。

Amazon EC2インスタンスを起動し、IAMロールを指定する

IAM ロールを使用して Amazon EC2インスタンスを作成して起動するには、次の手順を実行します。

- 「Amazon ユーザーガイド」の「インスタンスをすばやく起動する」を参照してください。 EC2 ただし、最終送信ステップの前に、以下も実行してください。
  - 詳細 のIAMインスタンスプロファイル で、前のステップで作成したロールを選択します。

この IAMと Amazon EC2のセットアップでは、アプリケーションを Amazon EC2インスタンスにデプロイでき、アプリケーションは Amazon S3 サービスへの読み取りアクセス権を持ちます。

EC2 インスタンスに接続する

Amazon EC2インスタンスに接続して、アプリケーションをそのインスタンスに転送し、アプリケーションを実行できるようにします。インスタンスの作成時にキーペア（ログイン）で使用したキーペアのプライベート部分を含む PEM ファイル、つまり ファイルが必要です。

これを行うには、インスタンスタイプのガイダンスに従います。Linux インスタンスに接続するか、Windows インスタンスに接続する。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。

> **Note**
>
> Linux または macOS ターミナルでは、セキュアコピーコマンドを使用してアプリケーションをコピーできます。キーペアscpで を使用するには、次のコマンドを使用できます: `scp -i path/to/key file/to/copy ec2-user@ec2-xx-xx-xxx-xxx.compute.amazonaws.com:~` 。
> Windows の詳細については、「Windows インスタンスへのファイルの転送」を参照してください。

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 インスタンスでアプリケーションを実行する

1. ローカルドライブから Amazon EC2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。
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）アプリケーションがIAMロールから提供された認証情報を使用していることを確認します。

    a. にサインイン AWS Management Console し、 で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
    b. インスタンスを選択します。
    c. アクション 、セキュリティ を選択し、IAMロールの変更 を選択します。
    d. IAM ロール では、IAMロール なし を選択してIAMロールをデタッチします。

e. IAM ロールの更新 を選択します。

f. 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 APIsに言語固有の を提供します AWS のサービス。認証、再試行動作など、API通話を正常に行うために必要な負担の一部を負担します。そのためには、 SDKsには、リクエストに使用する認証情報を取得し、各サービスで使用する設定を維持し、グローバル設定に使用する値を取得する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 および Tools 標準化された認証情報プロバイダー – 複数の にまたがって標準化された一般的な認証情報プロバイダーSDKs。
- AWS SDKs およびツールの標準化された機能 – 複数の で標準化された一般的な機能SDKs。

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には AWS のサービス、各 にクライアントクラス/オブジェクトSDKsを使用します AWS のサービス。例えば、アプリケーションが Amazon にアクセスする必要がある場合EC2、アプリケーションはそのサービスとインターフェイスする 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 ではSDKs、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、リクエストを行うサービスごとに新しいクライアントを作成し、別の設定を使用して同じサービスにリクエストを行う必要があります。

設定の優先順位

グローバル設定は、ほとんどの でサポートされ、 全体に広範な影響を与える機能SDKs、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 AWS のサービス。すべての SDKs には、グローバル設定の値を見つけるためにチェックする一連の場所 (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で入力するオペレーションごとのパラメータの形式になります。の場合SDK、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、場合によっては個々の を呼び出すときにもかまいませんAPI。

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. は、共有credentialsファイルで設定SDKをチェックします。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、 はそれSDKを使用します。
   - AWS_PROFILE 環境変数またはaws.profileJVMシステムプロパティを使用して、SDK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**
>
> 一部の SDKsおよび ツールでは、異なる順序でチェックされる場合があります。また、一部の SDKs および ツールでは、パラメータを保存および取得する他の方法をサポートしています。例えば、 は SDK Store と呼ばれる追加のソース AWS SDK for .NET をサポートしています。SDK または ツールに固有のプロバイダーの詳細については、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。例えば、共有configファイルにプロファイルを設定した場合、 SDKまたは ツールが他の場所を最初にチェック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) またはコードでの設定は、その 1 つのコマンドの他のすべての値を上書きします。

設定ページ

このガイドの設定リファレンスセクションのページには、さまざまなメカニズムで設定できる利用可能な設定の詳細が記載されています。次のテーブルには、設定ファイルと認証情報ファイルの設定、環境変数、および (Java と Kotlin の場合SDKs) コードの外部で機能を設定するために使用でき

るJVM設定が一覧表示されます。各リスト内のリンクされたトピックごとに、対応する設定ページに移動します。

- Config ファイル設定リスト
- Credentials ファイル設定リスト
- 環境変数の一覧
- JVM システムプロパティリスト

各認証情報プロバイダーまたは機能には、その機能の設定に使用される設定が一覧表示されるページがあります。設定ごとに、設定を設定ファイルに追加するか、環境変数を設定するか、JVMシステムプロパティを設定して (Java と Kotlin のみ) 値を設定できます。各設定には、説明の詳細の上にあるブロックで値を設定するサポートされているすべての方法が一覧表示されます。優先順位は異なりますが、結果の機能は設定方法に関係なく同じです。

説明には、何もしない場合に有効になるデフォルト値が含まれる。また、その設定の有効な値も定義します。

例えば、リクエスト圧縮機能ページから設定を見てみましょう。

サンプルdisable_request_compression設定の情報は、以下を伝達します。

- コードベース外でリクエストの圧縮を制御するには、3 つの同等の方法があります。次のいずれかを行うことができます。
  - を使用して設定ファイルに設定 disable_request_compression
  - を使用して環境変数として設定する AWS_DISABLE_REQUEST_COMPRESSION
  - または、Java または Kotlin を使用している場合はSDK、 を使用してJVMシステムプロパティとして設定します。 aws.disableRequestCompression

  > **Note**
  >
  > また、コード内で同じ機能を直接設定する方法もありますが、このリファレンスでは、各に固有のため、これについては説明していませんSDK。コード自体で設定を設定する場合は、特定のSDKガイドまたはAPIリファレンスを参照してください。

- 何もしない場合、値はデフォルトで になりますfalse。
- このブール値設定の有効な値は trueと のみですfalse。

各機能ページの下部には、 テーブルとの互換性 AWS SDKsがあります。

この表は、 がページにリストされている設定SDKをサポートしているかどうかを示しています。Supported 列には、次の値を持つサポートレベルが表示されます。

- Yes – 設定は、記述されているSDKように によって完全にサポートされています。
- Partial – 一部の設定がサポートされているか、動作が説明から逸脱しています。の場合Partial、追加のメモは偏差を示します。
- No – どの設定もサポートされていません。これにより、同じ機能がコードで達成されるかどうかが主張されるわけではなく、リストされている外部設定がサポートされていないことを意味します。

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 および ツールは、一意の設定と環境変数をサポートする場合もあります。個人SDKまたはツールのみがサポートす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	
account_id_endpoint_mode	アカウントベースのエンドポイント	
api_versions	一般設定	
aws_access_key_id	AWS アクセスキー	
aws_account_id	アカウントベースのエンドポイント	
aws_secret_access_key	AWS アクセスキー	
aws_session_token	AWS アクセスキー	

設定名	詳細	
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_service_endpoint_mode	IMDS 認証情報プロバイダー	
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	

設定名	詳細	
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	
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	

設定名	詳細	
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 リージョンエンドポイント	
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 および ツールは、一意の設定と環境変数をサポートする場合もあります。個人SDKまたはツールのみがサポートす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 でサポートされている環境変数SDKsを次の表に示します。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 および ツールは、一意の設定と環境変数をサポートする場合もあります。個人SDKまたはツールのみがサポートす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_ACCOUNT_ID	アカウントベースのエンドポイント	
AWS_ACCOUNT_ID_ENDPOINT_MODE	アカウントベースのエンドポイント	
AWS_CA_BUNDLE	一般設定	

設定名	詳細	
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	
AWS_EC2_METADATA_DISABLED	IMDS 認証情報プロバイダー	

設定名	詳細	
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_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	

設定名	詳細	
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_ROLE_ARN	ロール認証情報プロバイダーを引き受けます	
AWS_ROLE_SESSION_NAME	ロール認証情報プロバイダーを引き受けます	
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	
AWS_SDK_UA_APP_ID	アプリケーション ID	

設定名	詳細	
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 (をターゲットとするJVM) 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の設定方法については、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.accountId	アカウントベースのエンドポイント	

設定名	詳細	
aws.accountIdEndpointMode	アカウントベースのエンドポイント	
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 および Tools 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に標準化され、多くの で同じように動作しますSDKs。この一貫性により、複数の 間でコーディングする際の生産性と明確性が向上しますSDKs。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、特定の SDK を参照してくださいAPI。

> **Important**
>
> すべての 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、プロバイダー内のすべての側面もSDKsサポートしているわけではありません。

トピック

- 認証情報プロバイダーチェーンを理解する
- SDK- 固有およびツール固有の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を理解する

すべての SDKs には、 へのリクエストに使用する有効な認証情報を見つけるためにチェックする一連の場所 (またはソース) があります AWS のサービス。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、認証情報プロバイダーチェーンと呼ばれます。

標準化された認証情報プロバイダーの 1 つを使用する場合、 AWS SDKsは常に有効期限が切れると認証情報を自動的に更新しようとします。組み込みの認証情報プロバイダーチェーンを使用すると、チェーンで使用しているプロバイダーに関係なく、認証情報を更新できます。これSDKを行うには、追加のコードは必要ありません。

それぞれで使用される個別のチェーンはSDK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 Connect で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	Amazon、Facebook、Google、またはその他の OpenID Connect () 互換の IdP によるログインなど、よく知られた外部 ID プロバイダー (IdP OIDC) を使用してサインインします。 AWS Security Token Service () のJSONウェブトークン (JWT) 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。

認証情報プロバイダー	説明
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て、他のリソースにアクセスします。(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。IAM Roles Anywhere を含む外部ソースまたはプロセスから認証情報を取得します。
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介して配信されます。

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。

SDK- 固有およびツール固有の認証情報プロバイダーチェーン

SDKまたは ツールの特定の認証情報プロバイダーチェーンの詳細に直接移動するには、以下から SDKまたは ツールを選択します。

- AWS CLI
- SDK C++ 用
- SDK for Go
- SDK Java の場合
- SDK の JavaScript

- SDK Kotlin の場合
- SDK の 。NET
- SDK の PHP
- SDK Python 用 (Boto3)
- SDK Ruby の場合
- SDK Rust の場合
- SDK Swift の場合
- のツール PowerShell

AWS アクセスキー

> **Warning**
>
> セキュリティリスクを回避するため、専用ソフトウェアの開発時や実際のデータの使用時に、認証にIAMユーザーを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。は AWS SDKこれらの AWS 認証情報を自動的に使用して へのAPIリクエストに署名するため AWS、ワークロードは AWS リソースとデータに安全かつ便利にアクセスできます。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。長期認証情報の使用は推奨されません。

> **Note**
>
> AWS がこれらの一時的な認証情報を更新できない場合は、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に使用され、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の場所です。

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。

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。セキュリティに関する推奨事項については、「」の「セキュリティのベストプラクティスIAM」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
```

との互換性 AWS SDKs

このトピックで説明されている機能と設定を以下SDKsに示します。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 C++ 用	あり	共有 config ファイルはサポートされていません。
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用の	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	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 Kotlin 用の	あり	
SDK の 。NET 3.x	あり	環境変数はサポートされていません。
SDK 3.x PHP の場合	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用の	あり	
SDK Rust の	あり	
SDK Swift の	あり	
のツール PowerShell	あり	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、まず引き受ける特定のロールを作成または識別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) によって一

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。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、またはフェSAMLデレーションです。

IAM ロールが特定されたら、そのロールから信頼されている場合は、ロールによって付与されたアクセス許可を使用するように 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「AWS 認証情報を持つロールを想定する」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 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アクセス許可を持つ認証情報を 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

有効な値:

- 環境 – SDKまたはツールが環境変数 `AWS_ACCESS_KEY_ID`および `AWS_SECRET_ACCESS_KEY`からソース認証情報を取得するように指定します。
- Ec2InstanceMetadata – SDKまたはツールがIAMEC2インスタンスプロファイルにアタッチされた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IAMECSコンテナにアタッチされた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 から取得する必要があることを示すために、これを `config` ファイルに設定する例EC2：

```
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の表示」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： なし。

有効な値： 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。 IAM

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ユーザーが使用する必要がある多要素認証 (MFA) デバイスの ID また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ときに必要です。の詳細についてはMFA、「 IAMユーザーガイド」のAWS 「 での多要素認証IAM」を参照してください。

デフォルト値： なし。

有効な値： 値は、ハードウェアデバイスのシリアル番号 (などGAHT12345678) 、または仮想MFAデバイスの Amazon リソースネーム (ARN) のいずれかです。の形式ARNは次のとおりです。 arn:aws:iam::*account-id*:mfa/*mfa-device-name*

config ファイルにこれを設定する例を以下に示します。

この例では、アカウント用に作成されMyMFADevice、ユーザーに対して有効になっている という仮想MFAデバイスを前提としています。

```
mfa_serial = arn:aws:iam::123456789012:mfa/MyMFADevice
```

role_arn - 共有 AWS **config**ファイル設定, **AWS_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オペレーションの実行に使用する 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、次のようにフォーマットされた IAMロールARNの である必要があります。 arn:aws:iam::*account-id*:role/*role-name*

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アタッチされた認証情報、または Amazon ECSコンテナインスタンスのいずれかを使用します。
- web_identity_token_file – モバイルアプリケーションまたはウェブアプリケーションで認証されたユーザーに対して、パブリック ID プロバイダーまたは OpenID Connect (OIDC) 互換 ID プロバイダーを使用します。

role_session_name - 共有 AWS **config**ファイル設定, **AWS_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。詳細については、「 AWS CloudTrail ユーザーガイド」の「 CloudTrail userIdentity 要素」を参照してください。

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。このチェーンは、 が認証情報を持つプロファイル SDKを検出すると停止します。ロールの連鎖は、 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、増やすことはできません。詳細については、「 IAMユーザーガイド」の「ロールの用語と概念」を参照してください。

デフォルト値： なし。

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
```

```
role_session_name = ProfileARoleSession

[profile B]
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
```

前の例では、プロファイルはリンクされたAプロファイルの認証情報を自動的に検索するようにSDKまたは ツールに指示していますB。この場合、Bプロファイルは が提供する認証情報ヘルパーツールを使用して の AWS 認証情報IAM Roles Anywhereを取得しますSDK。これらの一時的な認証情報は、次に AWS リソースへのアクセスに使用されます。指定されたロールには、 コマンド、 APIメソッドなど AWS のサービス、リクエストされたコードの実行を許可する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。プロファイルによって実行されるすべてのアクションAには、 CloudTrail ログにロールセッション名が含まれます。

ロールの連鎖の 2 番目の例では、Amazon Elastic Compute Cloud インスタンスにアプリケーションがあり、そのアプリケーションに別のロールを引き受けさせたい場合に、次の設定を使用できます。

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

[profile B]
credential_source=Ec2InstanceMetadata
```

プロファイルAは、Amazon EC2インスタンスの認証情報を使用して指定されたロールを引き受け、認証情報を自動的に更新します。

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

サポートされている 2.0 プロバイダーまたは OpenID Connect ID ID OAuth 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。 OpenID

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引WebIdentityToken数として渡します。

デフォルト値： なし。

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、ID プロバイダーから提供された OAuth 2.0 アクセストークンまたは OpenID Connect トークンが含まれている必要があります。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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定を以下SDKsに示します。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 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 C++ 用	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 Go V2 用 (1.x)	はい	
SDK Go 1.x (V1) 用の	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。 duration_seconds 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 Java 1.x	部分的	`credential_source` サポートされません。 はサポートmfa_serial されません。 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 Kotlin 用の	はい	
SDK の 。NET 3.x	はい	
SDK 3.x PHP 用	はい	
SDK for Python (Boto3)	はい	
SDK Ruby 3.x 用の	はい	
SDK Rust 用の	はい	
SDK Swift の	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) のお客様に役立ちます。SDKs 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ようとします。

Amazon を使用する場合はECS、認証情報の分離、認可、監査可能性を向上させるためにタスクIAMロールを使用することをお勧めします。設定すると、Amazon は SDKsおよび 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ECSを設定します。この機能ECSに Amazon 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 を使用する場合はEKS、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、再利用可能性、スケーラビリティを向上させるために Amazon EKS Pod Identity を使用することをお勧めします。Pod とIAMロールの両方が Kubernetes サービスアカウントに関連付けられ、アプリケーションの認証情報を管理します。Amazon EKSPod Identity の詳細については、「Amazon ユーザーガイド」の「Amazon EKS Pod ID」を参照してください。 EKS 設定すると、Amazon は AWS_CONTAINER_CREDENTIALS_FULL_URIおよび ツールが認証情報を取得するために使用する SDKsおよび AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE環境変数EKSを設定します。セットアップ情報については、「Amazon ユーザーガイド」の「Amazon EKS Pod Identity Agent のセットアップ」または「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。 EKS AWS

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

認証情報のリクエストを行うときにSDK使用する の完全なHTTPURL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

認証情報のリクエストを行うときにSDK使用する の相対HTTPURL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 に追加されます169.254.170.2。

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は環境変数の値を使用してHTTPリクエストに認証ヘッダーを設定します。

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
```

との互換性 AWS SDKs

以下は、このトピックで説明されている機能と設定SDKsをサポートしています。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 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	可能	
SDK C++ の場合	可能	
SDK Go V2 用 (1.x)	可能	
SDK Go 1.x (V1) の場合	可能	
SDK for Java 2.x	可能	AWS_CONTAINER_CREDENTIALS_FULL_URI および AWS_CONTAINER_AUTHORIZATION_TOKEN は Lambda SnapStart for Java にも使用されます。
SDK Java 1.x 用	可能	AWS_CONTAINER_CREDENTIALS_FULL_URI および AWS_CONTAINER_AUTHORIZATION_TOKEN は Lambda SnapStart for Java にも使用されます。

SDK	サポート	注意または詳細情報
SDK JavaScript 3.x の場合	可能	
SDK JavaScript 2.x の場合	可能	
SDK Kotlin の場合	可能	
SDK の 。NET 3.x	可能	
SDK 3.x PHP の場合	可能	
SDK Python 用 (Boto3)	可能	
SDK Ruby 3.x 用	可能	
SDK Rust の場合	可能	
SDK Swift の場合	可能	
のツール PowerShell	可能	

IAM Identity Center 認証情報プロバイダー

この認証メカニズムでは、 AWS IAM Identity Center を使用して、コード AWS のサービス の へのシングルサインオン (SSO) アクセスを取得します。

> **Note**
>
> API ドキュメントでは、IAMIdentity Center AWS SDK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ます。

IAM Identity Center を有効にしたら、共有 AWS configファイルで設定のプロファイルを定義します。このプロファイルは、 IAM Identity Center アクセスポータルに接続するために使用されます。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。が設定から一時的な認証情報SDKを取得し、 AWS のサービス 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、次の 2 つの方法があります。

- (推奨) SSOトークンプロバイダー設定 — セッション継続時間を延長します。カスタムセッション期間のサポートが含まれています。
- 更新不可のレガシー設定 — 固定の 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。

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

SDKs および 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の詳細については、「」を参照してくださいIAM Identity Center 認証を理解する。

前提条件

最初に IAM Identity Center を有効にする必要があります。IAM Identity Center 認証の有効化の詳細については、AWS IAM Identity Center 「 ユーザーガイド」の「有効化 AWS IAM Identity Center」を参照してください。

> **Note**
>
> または、完全な前提条件と、このページで詳しく説明されている必要な共有configファイル設定については、「」のセットアップ手順を参照してくださいIAM SDKまたは ツールの Identity Center 認証。

SSO トークンプロバイダーの設定

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期間の詳細については、「 AWS IAM Identity Center ユーザーガイド」の AWS 「 アクセスポータルと IAM Identity Center 統合アプリケーションのセッション期間を設定する」を参照してください。

config ファイルの sso-session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ます。この変数は、 AWS 認証情報を取得するために使用できます。config ファイル内のこのセクションの詳細については、「」を参照してください設定ファイルの形式。

次の共有configファイルの例では、devプロファイルを使用して IAM Identity Center 認証情報をリクエストする SDKまたは ツールを設定します。

```
[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
```

前の例では、 sso-sessionセクションを定義してプロファイルに関連付ける方法を示しています。通常、 sso_account_idおよび は、 が AWS 認証情報をSDKリクエストできるように profileセクションで設定sso_role_nameする必要があります。sso_region、sso_start_url、および は sso-sessionセクション内で設定sso_registration_scopesする必要があります。

sso_account_id および sso_role_nameは、SSOトークン設定のすべてのシナリオで必要というわけではありません。アプリケーション AWS のサービス がベアラー認証をサポートす

る のみを使用する場合、従来の AWS 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。サービスがベアラートークン認可をサポートしているかどうかを確認するには、個々の AWS のサービス ガイドを参照してください。

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 のメカニズムです。OAuth 2.0 は、ユーザーのアカウントへのアプリケーションのアクセスを制限します。前の例では、アカウントとロールsso_registration_scopesを一覧表示するために必要なアクセスを提供するようにを設定します。

次の例は、複数のプロファイルで同じsso-session設定を再利用する方法を示しています。

```
[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
```

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の設定 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

この機能を設定するには、以下のように使用します。

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

組織の IAM Identity Center 発行者URLまたは アクセスポータル URL を指す URL。詳細については、「 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。 AWS IAM Identity Center

この値を見つけるには、IAMIdentity Center コンソールを開き、Dashboard を表示して、 AWS アクセスポータル URLを見つけます。

- または、 のバージョン 2.22.0 以降では AWS CLI、代わりにAWS 発行者 URLの値を使用できます。

sso_region - 共有 AWS **config**ファイル設定

IAM Identity Center ポータルホスト AWS リージョン を含む 、つまり IAM Identity Center を有効にする前に選択した リージョン。これはデフォルトの AWS リージョンとは独立しており、異なる場合があります。

AWS リージョン とそのコードの完全なリストについては、「」の「リージョンエンドポイント」を参照してくださいAmazon Web Services 全般のリファレンス。この値を見つけるには、IAM Identity Center コンソールを開き、ダッシュボードを表示してリージョンを見つけます。

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

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

使用可能なアカウントのリストを表示するには、IAMIdentity Center コンソールに移動し、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表示することもできます。例えば、 AWS CLI メソッド list-accounts を呼び出すことができます。

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

ユーザーの結果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。ロールは、 で AWS アカウント 指定された 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 () ではなく、ロール名を使用しますARN。

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、ユーザーが割り当てた に対するアクセスレベルを定義します AWS アカウント。

ごとに使用可能なアクセス許可セットのリストを表示するには AWS アカウント、 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 アカウント テーブルにリストされている正しいアクセス許可セット名を選択します。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、使用可能なアクセス許可セットのリストを表示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 - 共有 AWS **config**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。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。IAM Identity Center サービスから更新トークンを返すには、 の最小スコープを付与sso:account:accessする必要があります。使用可能なアクセススコープオプションのリストについては、「 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。

これらのスコープは、登録されたOIDCクライアントとクライアントによって取得されたアクセストークンに対して承認されるようにリクエストされたアクセス許可を定義します。スコープは、IAMIdentity Center ベアラートークン認可エンドポイントへのアクセスを許可します。

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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定を以下SDKsに示します。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 C++ 用	はい	
SDK Go V2 用 (1.x)	はい	
SDK Go 1.x (V1) 用の	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	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	いいえ	
SDK for JavaScript 3.x	はい	
SDK for JavaScript 2.x	はい	
SDK Kotlin 用の	はい	

SDK	サポート	注意または詳細情報
SDK の 。NET 3.x	はい	
SDK 3.x PHP 用	はい	
SDK for Python (Boto3)	はい	
SDK Ruby 3.x 用の	はい	
SDK Rust の	部分的	更新不可のレガシー設定のみ。
SDK Swift の	はい	
のツール PowerShell	はい	

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、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できるインスタンスに関するデータを提供します。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の操作」を参照してください。Amazon EC2は、インスタンスにさまざまなビットの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に提供します。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。SDKs 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。セッショントークンIMDSを使用するより安全なバージョンのインスタンスメタデータサービスバージョン 2 (IMDSv2) がデフォルトで使用されます。再試行不可能な条件 (HTTPエラーコード 403、404、405) が原因でこれが失敗した場合、IMDSv1は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

Amazon EC2 Instance Metadata Service (IMDS) を使用して認証情報を取得しようとするかどうか。

デフォルト値: `false`。

有効な値:

- **true** – 認証情報の取得IMDSに を使用しないでください。
- **false** – IMDSを使用して認証情報を取得します。

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

がIMDSv2失敗した場合、インスタンスメタデータサービスバージョン 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 SDKs 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 との互換性 AWS SDKs を参照してください。

デフォルト値: `false`。

有効な値:

- **true** – をフォールバックIMDSv1として使用しないでください。
- **false** – フォールバックIMDSv1として を使用します。

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

のエンドポイントIMDS。この値は、 および のツールが AWS SDKs Amazon EC2インスタンスメタデータを検索するデフォルトの場所を上書きします。

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

のエンドポイントモードIMDS。

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 の一部です認証情報プロバイダーチェーンを理解する。ただし、IMDS認証情報プロバイダーは、このシリーズに含まれる他の複数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。または、認証情報プロバイダーチェーンに依存して有効な認証情報を返すプロバイダーを自動的に検出する代わりに、コードでのIMDS認証情報プロバイダーの使用を指定します。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が有効な認証情報で設定されていない場合、 SDK は Amazon EC2 Instance Metadata Service (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、不要なネットワークアクティビティを防ぎ、Amazon EC2 Instance Metadata Service がなりすまされる可能性のある信頼されていないネットワークのセキュリティを強化できます。

> **Note**
>
> AWS SDK 有効な認証情報で設定されたクライアントは、これらの設定にかかわらず、認証情報の取得IMDSに を使用することはありません。

Amazon EC2IMDS認証情報の使用を無効にする

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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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 AWS SDK for Kotlin のみがサポートします。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK C++ 用	はい	
SDK Go V2 用 (1.x)	はい	
SDK Go 1.x (V1) の場合	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 Java 2.x 用	はい	
SDK Java 1.x 用	部分的	JVM システムプロパティ: com.amazonaws.sdk.disableEc2MetadataV1 の代わりに を使用しaws.disableEc2MetadataV1 aws.ec2MetadataServiceEndpoint 、aws.ec2MetadataServiceEndpointMode サポートされていません。
SDK JavaScript 3.x の場合	はい	
SDK JavaScript 2.x の場合	はい	
SDK Kotlin の場合	はい	IMDSv1 フォールバックを使用しません。
SDK 用。NET 3.x	はい	

SDK	サポート	注意または詳細情報
SDK 3.x PHP の場合	はい	
SDK Python 用 (Boto3)	はい	
SDK Ruby 3.x 用	はい	
SDK Rust の場合	はい	IMDSv1 フォールバックを使用しません。
SDK Swift の場合	はい	
のツール PowerShell	はい	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

SDKs は、カスタムユースケースのために認証情報プロバイダーチェーンを拡張する方法を提供します。このプロバイダーは、オンプレミスの認証情報ストアから認証情報を取得したり、オンプレミスの ID プロバイダーと統合したりするなど、カスタム実装を提供するために使用できます。

例えば、IAMRoles Anywhere は `credential_process` を使用して、アプリケーションに代わって一時的な認証情報を取得します。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。

> **Note**
>
> 以下は、外部プロセスから認証情報を調達する方法を説明するもので、 の外部でソフトウェアを実行している場合に使用できます。 AWS。 で を構築する場合 AWS コンピューティングリソース、他の認証情報プロバイダーを使用します。このオプションを使用する場合は、

オペレーティングシステムのセキュリティのベストプラクティスを使用して、設定ファイルが可能な限りロックされていることを確認してください。SDKs と のためStdErr、カスタム認証情報ツールが にシークレット情報を書き込まないことを確認します。 AWS CLI はそのような情報をキャプチャしてログに記録し、不正なユーザーに公開する可能性があります。

この機能を設定するには、以下のように使用します。

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

使用する認証情報を生成SDKまたは取得するために、ユーザーに代わって または ツールが実行する外部コマンドを指定します。設定SDKは、 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 にJSONデータを書き込むのを待ちますstdout。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を理解する 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。パスはオペレーティングシステムによって異なります。

  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"
  ```

- 専用プロファイル内で指定できます。

  ```
  [profile cred_process]
  credential_process = /Users/username/process.sh
  region = us-east-1
  ```

認証情報プログラムからの有効な出力

は、プロファイルで指定されたコマンドSDKを実行し、標準出力ストリームからデータを読み取ります。スクリプトまたはバイナリプログラムのどちらで、指定したコマンドは、次の構文STDOUTに一致するJSON出力を で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フォーマットされたタイムスタンプです。Expiration キーがツールの出力に存在しない場合、 は認証情報が更新しない長期的な認証情報であるSDKと仮定します。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、ロール認証情報を引き受ける方法のように、外部プロセス認証情報を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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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

SDK	サポート	注意または詳細情報
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	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	あり	
SDK JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	あり	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	

SDK	サポート	注意または詳細情報
SDK Rust 用	あり	
SDK Swift 用	あり	
のツール PowerShell	あり	

AWS SDKs およびツールの標準化された機能

多くの機能は、一貫したデフォルトに標準化され、多くの で同じように動作しますSDKs。この一貫性により、複数の にまたがってコーディングする際の生産性と明確さが向上しますSDKs。すべての設定はコードで上書きできます。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 がすべての機能SDKs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

- アカウントベースのエンドポイント
- アプリケーション ID
- Amazon EC2 インスタンスメタデータ
- Amazon S3 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
- AWS リージョン
- AWS STS リージョンエンドポイント
- デュアルスタックとFIPSエンドポイント
- エンドポイント検出

- 一般設定
- IMDS クライアント
- 再試行動作
- リクエスト圧縮
- サービス固有のエンドポイント
- スマート設定デフォルト

アカウントベースのエンドポイント

アカウントベースのエンドポイントは、 AWS アカウント ID を使用してこの機能をサポートするサービスの AWS のサービス リクエストのルーティングを合理化することで、高いパフォーマンスとスケーラビリティを確保します。アカウントベースのエンドポイントをサポートする AWS SDK、認証情報プロバイダー、サービスを使用すると、 SDKはリージョンエンドポイントではなく、アカウントベースのエンドポイントを自動的に構築して使用します。アカウントベースのエンドポイントは の形式をとりhttps://*<account-id>*.ddb.*<region>*.amazonaws.com、 *<account-id>*は AWS アカウント ID に置き換えられ、 *<region>*は に置き換えられます。 AWS リージョン

デフォルトでは、アカウント ID はリクエストが処理され、エンドポイントの構築に使用されるときに収集されます。認証情報解決は、リクエストが処理されたときにも発生し、エンドポイント解決の方法を変更する可能性があります。使用している認証情報プロバイダーによっては、アカウント ID のソースが異なる場合があります。

この機能を設定するには、以下のように使用します。

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

AWS アカウント ID。アカウントベースのエンドポイントルーティングに使用されます。 AWS アカウント ID の形式は 111122223333 です。

アカウントベースのエンドポイントルーティングは、一部のサービスでより良いリクエストパフォーマンスを提供します。

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

この設定は、必要に応じてアカウントベースのエンドポイントルーティングをオフにし、アカウントベースのルールをバイパスするために使用されます。

デフォルト値: `preferred`

有効な値:

- **`preferred`** – エンドポイントには、利用可能な場合はアカウント ID を含める必要があります。
- **`disabled`** – 解決されたエンドポイントにはアカウント ID は含まれません。
- **`required`** – エンドポイントにはアカウント ID を含める必要があります。アカウント ID が利用できない場合、 はエラーをSDKスローします。

との互換性 AWS SDKs

以下は、このトピックで説明されている機能と設定SDKsをサポートしています。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 AWS SDK for Kotlin のみでサポートされています。

SDK	サポート	SDK バージョン でリリース	注意または詳細情報
AWS CLI v2	不可		
SDK C++ 用	不可		
SDK Go V2 用 (1.x)	可能	v1.35.0	
SDK Go 1.x (V1) の場合	不可		
SDK Java 2.x 用	可能	v2.28.4	
SDK Java 1.x 用	可能	v1.12.771	

SDK	サポート	SDK バージョン でリリース	注意または詳細情報
SDK JavaScript 3.x の場合	可能	v3.656.0	
SDK JavaScript 2.x の場合	不可		
SDK Kotlin の場合	可能	v1.3.37	
SDK の 。NET 3.x	不可		
SDK 3.x PHP の場合	可能	v3.318.0	
SDK Python 用 (Boto3)	不可		
SDK Ruby 3.x 用	可能	v1.123.0	
SDK Rust の場合	不可		
SDK Swift の場合	不可		
のツール PowerShell	不可		

アプリケーション ID

単一の AWS アカウント は、複数のカスタマーアプリケーションで を呼び出すために使用できます。 AWS のサービス。 アプリケーション ID を使用すると、 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 および サービスは、この

値をカスタマーコミュニケーションに表示するために 以外は使用または解釈しません。例えば、この値は運用 E メールや に含めることができます。 AWS Health Dashboard は、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。

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 内のアプリケーション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 アカウント が を呼び出す 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、必要に応じて値を工スケープします。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK C++ 用	あり	共有 config ファイルはサポートされていません。
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	なし	
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	なし	
SDK JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	なし	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	環境変数はサポートされていません。
SDK 3.x PHP 用	あり	

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

Amazon EC2 インスタンスメタデータ

Amazon EC2は、インスタンスメタデータサービス () と呼ばれるインスタンスでサービスを提供しますIMDS。このサービスの詳細については、「Amazon ユーザーガイド」の「インスタンスメタデータの使用」を参照してください。 EC2 IAM ロールで設定された Amazon EC2インスタンスで認証情報を取得しようとすると、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。

`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
```

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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 C++ 用	なし	
SDK Go V2 用 (1.x)	なし	
SDK Go 1.x (V1) 用	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 JavaScript 3.x 用	なし	
SDK JavaScript 2.x 用	なし	
SDK Kotlin 用	なし	
SDK の 。NET 3.x	なし	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	

SDK	サポート	注意または詳細情報
SDK Ruby 3.x 用	なし	
SDK Rust 用	なし	
SDK Swift 用	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。で 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でアクセスポイントの 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。バケットからオブジェクトARN`GetObject`を取得するアクセスポイントの使用や、バケットARN`PutObject`にオブジェクトを追加するアクセスポイントの使用など、特定のオペレーションに使用されます。

Amazon S3 アクセスポイントと の詳細についてはARNs、「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。 Amazon S3

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, コードで直接値を設定するには、特定の SDKを直接参照してください。

> この設定は、 がアクセスポイントSDKを使用するかどうかを制御します。 ARN AWS リージョン リクエストのリージョンエンドポイントを構築します。は、 SDKを検証します。 ARN AWS リージョン は同じ によって提供されます。 AWS クライアントの が設定されている パーティション AWS リージョン は、失敗する可能性が最も高いクロスパーティション呼び出しを防止します。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`

有効な値:

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

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK C++ 用	あり	
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	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	あり	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 JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	あり	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	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 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	なし	
SDK Swift 用	なし	
のツール PowerShell	あり	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は、複数の にある Amazon S3 バケットからのリクエストをアプリケーションが処理するために使用できるグローバルエンドポイントを提供します。AWS リージョン。 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て

い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、それらのアプリケーションを世界中のどこでも実行できます。

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 (ARNs) の詳細については、「Amazon 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の実行」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、 への依存関係SDKを通じて によって取得されますAWS 共通ランタイム (CRT) ライブラリ。

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` - 共有 AWS **`config`** 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, コードで直接値を設定するには、特定の SDKを直接参照してください。

この設定は、 がクロスリージョンリクエストSDKを試み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`

有効な値:

- **`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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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

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

AWS リージョン

AWS リージョン は、 を使用する際に理解しておくべき重要な概念です AWS のサービス。

を使用すると AWS リージョン、特定の地理的エリアに AWS のサービス 物理的に存在する 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。

ほとんどの AWS のサービス リクエストは、特定の地理的リージョンに関連付けられています。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。例えば、Amazon S3 と Amazon はクロスリージョンレプリケーションEC2をサポートしています。などの一部のサービスにはIAM、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。
- 各 AWS のサービスでサポートされているすべてのリージョンおよびエンドポイントの現在のリストを確認する方法については、の「サービスとエンドポイントの割り当て」を参照してください。

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には AWS のサービス、各 にクライアントクラス/オブジェクトSDKsを使用します AWS のサービス。例えばEC2、アプリケーションが Amazon 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する Amazon EC2クライアントオブジェクトを作成します。

コード自体で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、次の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、 に固有の、そのクライアントのインスタンス化中に指定されます SDK (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 - 共有 AWS **config**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 リクエスト AWS リージョン に使用するデフォルトを指定します。このリージョンは、使用する特定のリージョンで指定されていないSDKサービスリクエストに使用されます。

デフォルト値： なし。この値は明示的に指定する必要があります。

有効な値:

- 「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、この us-east-1 値により、エンドポイントは AWS リージョン 米国東部 (バージニア北部)に設定されます。
- aws-global は、 AWS Security Token Service （ AWS STS) や Amazon Simple Storage Service (Amazon S3) などのリージョンエンドポイントに加えて、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 SDKs には、アプリケーションコード内からデフォルトのリージョンを設定できる「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デベロッパーガイドを参照してください。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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 AWS SDK for Kotlin のみがサポートします。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	AWS CLI v2 は、 の値AWS_REGION の前に 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	はい	AWS CLI v1 はこの目的のために というAWS_DEFAULT_REGION 名前の環境変数を使用します。
SDK C++ 用	はい	
SDK Go V2 用 (1.x)	はい	
SDK Go 1.x (V1) の場合	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 Java 2.x 用	はい	

SDK	サポート	注意または詳細情報
SDK Java 1.x 用	はい	
SDK JavaScript 3.x の場合	はい	
SDK JavaScript 2.x の場合	はい	
SDK Kotlin の場合	はい	
SDK の 。NET 3.x	はい	
SDK 3.x PHP の場合	はい	
SDK Python 用 (Boto3)	はい	これは、この目的のために というAWS_DEFAULT_REGION 名前の環境変数SDKを使用します。
SDK Ruby 3.x 用	はい	
SDK Rust の場合	はい	
SDK Swift の場合	はい	
のツール PowerShell	はい	

AWS STS リージョンエンドポイント

AWS Security Token Service （ AWS STS) は、グローバルサービスとしてもリージョンサービスとしても利用できます。SDKs と の一部 AWS はデフォルトでグローバルサービスエンドポイント (`https://sts.amazonaws.com`) CLIsを使用し、一部はリージョンサービスエンドポイント () を使用しますhttps://sts.*{region_identifier}*.*{partition_domain}*。グローバルリクエストは、米国東部 (バージニア北部) リージョンにマッピングされます。 AWS STS エンドポイントの詳細については、 AWS Security Token Service APIリファレンスの「エンドポイント」を参照してください。または、 AWS Identity and Access Management ユーザーガイドの AWS STS で を管理する AWS リージョン方法を学びます。

可能な限りリージョンエンドポイントを使用し、 を設定することが AWS ベストプラクティスですAWS リージョン。商用以外のパーティションのお客様は、リージョンエンドポイントを使用する必要があります。すべての SDKsおよび ツールがこの設定をサポートしているわけではありませんが、グローバルエンドポイントとリージョンエンドポイントに関する動作が定義されています。詳細については、次のセクション を参照してください。

この設定をサポートする SDKsおよび ツールでは、お客様は以下を使用して機能を設定できます。

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

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

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新しいSDKメジャーバージョンは、デフォルトで になりますregional。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動作を使用する場合があります。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`** – グローバル AWS STS エンドポイント を使用しますsts.amazonaws.com。

- **regional** – SDKまたは ツールは、現在設定されているリージョンの AWS STS エンドポイントを常に使用します。例えば、クライアントが を使用するように設定されている場合us-west-2、 へのすべての呼び出し AWS STS は、グローバルエンドポイントではなくsts.us-west-2.amazonaws.com、リージョンsts.amazonaws.com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
```

との互換性 AWS SDKs

> **Note**
>
> 可能な限りリージョンエンドポイントを使用し、 を設定することが AWS ベストプラクティスですAWS リージョン。

次の表は、 SDKまたは ツールの概要を示しています。

- 設定をサポート: STSリージョンエンドポイントの共有configファイル変数と環境変数がサポートされるかどうか。
- デフォルト設定値 : サポートされている場合は、設定のデフォルト値。
- デフォルトのサービスクライアントターゲットSTSエンドポイント : 変更する設定が使用できない場合でも、クライアントが使用するデフォルトのエンドポイント。

- サービスクライアントのフォールバック動作 : リージョンエンドポイントを使用するはずだが、リージョンが設定されていない場合に がSDK行う動作。これは、デフォルトが原因でリージョンエンドポイントを使用しているか、 `regional`が 設定で選択されているかに関係なく動作です。

このテーブルでは、次の値も使用されます。

- グローバルエンドポイント : `https://sts.amazonaws.com`。
- リージョンエンドポイント : アプリケーションでAWS リージョン設定された に基づいています。
- **`us-east-1`** (リージョン) : `us-east-1`リージョンエンドポイントを使用しますが、一般的なグローバルリクエストよりも長いセッショントークンを使用します。

SDK		デフォルト設定値	デフォルトのサービスクライアントターゲットSTSエンドポイント	サービスクライアントのフォールバック動作	注意または詳細情報
AWS CLI v2	なし	該当なし	リージョンエンドポイント	グローバルエンドポイント	
AWS CLI v1	はい	`legacy`	グローバルエンドポイント	グローバルエンドポイント	
SDK C++ の場合	なし	該当なし	リージョンエンドポイント	`us-east-1` (リージョン)	
SDK Go V2 用 (1.x)	なし	該当なし	リージョンエンドポイント	リクエストの失敗	
SDK Go 1.x (V1) の場合	はい	`legacy`	グローバルエンドポイント	グローバルエンドポイント	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		デフォルト設定値	デフォルトのサービスクライアントターゲットSTSエンドポイント	サービスクライアントのフォールバック動作	注意または詳細情報
SDK Java 2.x 用	なし	該当なし	リージョンエンドポイント	リクエストの失敗	リージョンが設定されていない場合、 AssumeRole と AssumeRoleWithWebIdentity はグローバルSTSエンドポイントを使用します。
SDK Java 1.x 用	はい	legacy	グローバルエンドポイント	グローバルエンドポイント	
SDK JavaScript 3.x の場合	なし	該当なし	リージョンエンドポイント	リクエストの失敗	
SDK JavaScript 2.x の場合	はい	legacy	グローバルエンドポイント	グローバルエンドポイント	
SDK Kotlin の場合	なし	該当なし	リージョンエンドポイント	グローバルエンドポイント	
SDK の 。NET 3.x	はい	legacy	グローバルエンドポイント	グローバルエンドポイント	
SDK 3.x PHP の場合	はい	legacy	グローバルエンドポイント	リクエストの失敗	
SDK Python (Boto3) の場合	はい	legacy	グローバルエンドポイント	グローバルエンドポイント	

SDK		デフォルト設定値	デフォルトのサービスクライアントターゲットSTSエンドポイント	サービスクライアントのフォールバック動作	注意または詳細情報
SDK Ruby 3.x 用	はい	`regional`	リージョンエンドポイント	リクエストの失敗	
SDK Rust の場合	なし	該当なし	リージョンエンドポイント	リクエストの失敗	
SDK Swift の場合	なし	該当なし	リージョンエンドポイント	リクエストの失敗	
のツール PowerShell	はい	`legacy`	グローバルエンドポイント	グローバルエンドポイント	

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` - 共有 AWS **`config`** 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 の両方のIPv6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 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`

有効な値:

- **`true`** – SDKまたは ツールは、デュアルスタックのエンドポイントを使用してネットワークリクエストを実行しようとします。サービスおよび/または のデュアルスタックエンドポイントが存在しない場合 AWS リージョン、リクエストは失敗します。

- **`false`** – SDKまたは ツールは、デュアルスタックの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いません。

`use_fips_endpoint` - 共有 AWS **`config`** 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のセットです。政府機関、パートナー、および連邦政府とのビジネスを希望する者は、FIPSガイドラインに従う必要があります。標準と異なる AWS エンドポイント、 FIPSエンドポイントは、140-2 FIPS に準拠した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、 のサービスにFIPSエンドポイントが存在しない場合 AWS リージョン、 AWS 呼び出しが失敗することがあります。 サービス固有のエンドポイント および の `--endpoint-url`オプション AWS Command Line Interface この設定を上書きします。

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

デフォルト値: `false`

有効な値:

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

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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 C++ 用	あり	
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	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	なし	
SDK JavaScript 3.x 用	あり	
SDK JavaScript 2.x の場合	あり	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	

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

エンドポイント検出

SDKs 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にアクセスする (URLs を使用してさまざまなリソースにアクセスする) 一方で、 の柔軟性を維持します。 AWS URLs 必要に応じて を変更します。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。例えば、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 などの特定のオペレーションを完了しますCreateTable。

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, コードで直接値を設定するには、特定の SDKを直接参照してください。

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 リクエストはデフォルトで にtrue、Amazon DynamoDB リクエストはデフォルトで に設定されますfalse。

有効な値:

- **true** – は、エンドポイント検出がオプションであるサービスのエンドポイントを自動的に検出しようとSDKします。
- **false** – SDKは、エンドポイント検出がオプションであるサービスのエンドポイントを自動的に検出しようとしないでください。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK C++ 用	あり	
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	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 2.x は、環境変数名AWS_ENDPOINT_DISCOVERY_ENABLED に を使用します。

SDK	サポート	注意または詳細情報
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 JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	あり	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	部分的	Timestream でのみサポートされます。
SDK Swift 用	なし	
のツール PowerShell	あり	

一般設定

SDKs は、全体的なSDK動作を設定する一般的な設定をサポートしています。

この機能を設定するには、以下のように使用します。

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

ある程度 AWS のサービスは、下位互換性をサポートするために複数のAPIバージョンを維持します。デフォルトでは、 SDKおよび AWS CLI オペレーションは、利用可能な最新バージョンを使用しますAPI。リクエストに使用する特定のAPIバージョンを要求するには、プロファイルに api_versions設定を含めます。

デフォルト値： なし。 (最新バージョンAPIは で使用されます) SDK。

有効な値： これはネストされた設定で、その後にそれぞれが 1 行ずつ識別する 1 つ以上のインデントされた行が続きます。 AWS 使用する サービスとAPIバージョン。のドキュメントを参照してください。 AWS 利用可能なAPIバージョンを把握するための サービス。

この例では、2 つの に特定のAPIバージョンを設定します。 AWS config ファイル内の サービス。これらのAPIバージョンは、これらの設定を含むプロファイルで実行されるコマンドにのみ使用されます。他のサービスのコマンドは、そのサービスの の最新バージョンを使用します API。

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

ca_bundle - 共有 AWS **config** 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 (.pem拡張子を持つファイル) 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
```

オペレーティングシステムがパスを処理する方法やパス文字のエスケープ方法が異なるため、Windows の config ファイルでこの値を設定する例を次に示します。

```
[default]
ca_bundle = C:\\Users\\username\\.aws\\aws-custom-bundle.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 - 共有 AWS **config** ファイル設定

で結果のフォーマット方法を指定します。 AWS CLI およびその他の AWS SDKs および ツール。

デフォルト値: json

有効な値:

- **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。
- **yaml-stream** – 出力はストリーミングされ、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 AWS **config** ファイル設定

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

デフォルト値: true

有効な値:

- **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の検証を実行します。これにより、 SDKまたは ツールはパラメータが有効であることを確認し、いくつかのエラーを検出できます。SDK または ツールは、 にリクエストを送信する前に、無効なリクエストを拒否できます。 AWS サービスエンドポイント。
- **false** – SDKまたは ツールは、コマンドラインパラメータを 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。- 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、無効なリクエストを拒否する責任があります。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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 C++ 用	あり	
SDK Go V2 用 (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 Go 1.x (V1) 用	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	なし	

SDK	サポート	注意または詳細情報
SDK for Java 1.x	なし	
SDK JavaScript 3.x 用	あり	
SDK JavaScript 2.x の場合	あり	
SDK Kotlin 用	なし	
SDK の 。NET 3.x	なし	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	なし	
SDK Swift 用	なし	
のツール PowerShell	なし	

IMDS クライアント

SDKs 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バージョン 2 (IMDSv2) クライアントを実装します。の詳細についてはIMDSv2、「Amazon ユーザーガイド」の「 を使用するIMDSv2」を参照してください。 EC2 IMDS クライアントは、SDKコードベースで使用可能なクライアント設定オブジェクトを介して設定できます。

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TTLの 。

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

のエンドポイントIMDS。

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。

ほとんどの では、以下のオプションがサポートされていますSDKs。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

のエンドポイントモードIMDS。

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： のような有効な I/O ストリームSTDOUT。

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： によって異なりますSDK。

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

との互換性 AWS SDKs

以下は、このトピックで説明されている機能と設定SDKsをサポートしています。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 AWS SDK for Kotlin のみがサポートします。

SDK	サポート	注意または詳細情報
AWS CLI v2	はい	
SDK C++ 用	なし	
SDK Go V2 用 (1.x)	はい	
SDK Go 1.x (V1) の場合	はい	
SDK Java 2.x 用	はい	
SDK Java 1.x 用	はい	
SDK JavaScript 3.x の場合	はい	
SDK JavaScript 2.x の場合	はい	
SDK Kotlin の場合	なし	
SDK の 。NET 3.x	はい	

SDK	サポート	注意または詳細情報
SDK 3.x PHP の場合	はい	
SDK Python 用 (Boto3)	はい	
SDK Ruby 3.x 用	はい	
SDK Rust の場合	はい	
SDK Swift の場合	はい	
のツール PowerShell	はい	

再試行動作

再試行動作には、 へのリクエストに起因する障害からの復旧のSDKs試みに関する設定が含まれます。 AWS のサービス.

この機能を設定するには、以下のように使用します。

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 またはデベロッパー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： この値は に固有ですSDK。特定のSDKガイドまたは SDKのコードベースでデフォルトの を確認しますretry_mode。

有効な値:

- standard – (推奨) 全体の推奨再試行ルールセット AWS SDKs。このモードには、再試行されるエラーの標準セットが含まれており、再試行回数を自動的に調整して可用性と安定性を最大化します。このモードは、マルチテナントアプリケーションで安全に使用できます。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 – 標準モードの機能やクライアント側の自動レート制限を含む、特殊なユースケースにのみ適した再試行モード。この再試行モードは、アプリケーションテナントを分離するように注意しない限り、マルチテナントアプリケーションにはお勧めしません。詳細については、「standard 再試行モードとadaptive再試行モードの選択」を参照してください。このモードは実験的であり、将来的に動作が変わる可能性があります。
- legacy – (非推奨) に固有の SDK (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) 。

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 の場合 legacy – に固有のデフォルト値を使用します SDK (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。

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す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り、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。

標準	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するか、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。
サーキットブレークをサポートし、停止中にSDKが再試行するのを防ぎます。	サーキットブレークをサポートし、停止中にSDKが再試行するのを防ぎます。
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らせず、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、各リソース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. 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 が を使用する AWS リージョン と、リクエストをスロットリングするためにアクセスされるテーブル。つまり、コードがアクセスしている 1 つのテーブルは、他のテーブルよりもスロットリングされ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、R egion-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、特定の を参照してください。 AWS 使用している サービスの ドキュメントガイド。

再試行モードの実装の詳細

- AWS SDKs は、トークンバケットを使用して、リクエストを再試行するかどうか、および (再試行モードの場合は) adaptive リクエストを送信する速度を決定します。では、SDK再試行トークンバケットとリクエストレートトークンバケットの 2 つのトークンバケットが使用されます。

- 再試行トークンバケットは、停止中にアップストリームサービスとダウンストリームサービスを保護するために、 がSDK一時的に再試行を無効にする必要があるかどうかを判断するために使用されます。トークンは再試行する前にバケットから取得され、リクエストが成功するとトークンはバケットに返されます。再試行時にバケットが空の場合、 SDK はリクエストを再試行しません。
- リクエストレートトークンバケットは、リクエストの送信レートを決定するためにadaptive、再試行モードでのみ使用されます。トークンはリクエストの送信前にバケットから取得され、サービスによって返されるスロットリングレスポンスに基づいて動的に決定されたレートでバケットに返されます。

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:

このステップはadaptive再試行モードでのみ使用されます。このステップでは、リクエストレートトークンバケットからトークンを取得します。トークンが利用できない場合、トークンが利用可能になるまで待機します。には、待機ではなくリクエストに失敗するための設定オプションがあるSDK場合があります。バケット内のトークンは、クライアントが受信したスロットリングレスポンスの数に基づいて動的に決定されるレートで補充されます。

SendHTTPRequest:

このステップでは、リクエストを に送信します。 AWS。 最も多い AWS SDKs は、HTTPリクエストを行うときに、接続プールを使用して既存の接続を再利用する HTTPライブラリを使用します。通常、スロットリングエラーが原因でリクエストが失敗した場合、接続は再利用されますが、一時的なエラーが原因でリクエストが失敗した場合は再利用されません。

RequestBookkeeping:

リクエストが成功すると、トークンがトークンバケットに追加されます。adaptive 再試行モードのみの場合、リクエストレートトークンバケットのフィルレートは、受信したレスポンスのタイプ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。サービスからのHTTP応答を受信しない SDK が受信したエラーとして定義されます。

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 再試行動作は、サービスからのエラーコードやその他のデータと組み合わせて決定されます。

MAX_ATTEMPTS:

デフォルトの最大試行回数は、 retry_mode設定によって上書きされない限り、 max_attempts設定によって設定されます。

HasRetryQuota

このステップでは、再試行トークンバケットからトークンを取得します。再試行トークンバケットが空の場合、リクエストは再試行されません。

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。は、ジッターで切り捨てられたバイナリエクスポネンシャルバックオフSDKs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`

`MAX_BACKOFF = 20 seconds` ほとんどの では SDKs。確認のために、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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

SDK	サポート	注意または詳細情報
AWS CLI v2	あり	
SDK C++ 用	あり	
SDK Go V2 用 (1.x)	あり	

SDK	サポート	注意または詳細情報
SDK Go 1.x (V1) 用	なし	
SDK for Java 2.x	あり	
SDK for Java 1.x	あり	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 JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	なし	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 Kotlin 用	あり	
SDK の 。NET 3.x	あり	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	あり	

SDK	サポート	注意または詳細情報
SDK Swift 用	あり	
のツール PowerShell	あり	

リクエスト圧縮

> **Note**
>
> 設定ページのレイアウトを理解したり、 との互換性を解釈したりするのに役立ちます。AWS SDKs 次の表については、「」を参照してください設定ページ。

AWS SDKs および ツールは、 にリクエストを送信するときにペイロードを自動的に圧縮できます。AWS のサービス 圧縮ペイロードの受信をサポートする 。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。圧縮の場合、 SDKまたは ツールは、サービスと の両方でサポートされているエンコーディングアルゴリズムを選択しますSDK。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

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オンまたはオフにします。

デフォルト値: `false`

有効な値:

- **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 ツールが圧縮するリクエストボディの最小サイズをバイト単位で設定します。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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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

SDK	サポート	注意または詳細情報
SDK Go V2 用 (1.x)	あり	
SDK Go 1.x (V1) 用	なし	
SDK for Java 2.x	あり	
SDK for Java 1.x	なし	
SDK JavaScript 3.x 用	あり	
SDK JavaScript 2.x 用	なし	
SDK Kotlin 用	あり	
SDK の 。NET 3.x	あり	
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	あり	

SDK	サポート	注意または詳細情報
SDK Swift 用	なし	
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では、APIリクエストに選択したエンドポイントを使用し、その選択を永続化することができます。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を柔軟にサポートできます。 AWS 開発環境。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。個々の のエンドポイントを指定できます URL。 AWS のサービス.

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。

この設定は、共有 の `services`セクションでも使用できます。 AWS `config` 特定のサービスのカスタムエンドポイントを設定する ファイル。`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URLを含む 。には、オプションで 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URL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>。ここで、 <SERVICE>は です。 AWS のサービス 識別子。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URLを含む 。には、オプションで 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URL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 コマンドラインパラメータを に含めます。 AWS CLI コマンド、またはエンドポイントをクライアントコンストラクターURLに渡すと、常に有効になります。

デフォルト値: false

有効な値:

- **true** – SDKまたは ツールは、エンドポイント を設定するための共有configファイルまたは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URL。
- **false** – SDKまたは ツールは、共有configファイルまたは環境変数から利用可能なユーザー提供のエンドポイントを使用します。

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 にルーティングするにはURL、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 のリクエストをルーティングするには AWS のサービス をカスタムエンドポイント に URL、 AWS_ENDPOINT_URL_<SERVICE>環境変数を使用します。 Amazon DynamoDB には serviceIdの があります[DynamoDB]。このサービスの場合、エンドポイントURL環境変数は ですAWS_ENDPOINT_URL_DYNAMODB。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、 AWS Elastic Beanstalk には serviceIdの 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 URLに使用するサービス固有のエンドポイントと、他のすべてのサービスに使用す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 および URLsのサービス固有のエンドポイントを設定する方法を示します。 AWS Elastic Beanstalk 同じプロファイル内の 。 AWS Elastic Beanstalk には `serviceId`の 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。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

プロファイルにロールIAMを引き受ける機能用の `source_profile`パラメータで設定されたロールベースの認証情報がある場合、 は指定されたプロファイルのサービス設定SDK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
```

プロファイルを使用してコードで Amazon を呼び出すBとEC2、エンドポイントは として解決されますhttps://profile-b-ec2-endpoint.aws。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呼び出しの場合 AWS のサービスでは、次の順序を使用して値を選択します。

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。
   - 向けの 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。の場合SDK、明示的な割り当ては、 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取ることができます。 AWS のサービス クライアントまたは設定オブジェクト。
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 URLのデフォルトエンドポイント AWS のサービス が最後に使用されます。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

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

SDK	サポート	注意または詳細情報
SDK 3.x PHP 用	あり	
SDK for Python (Boto3)	あり	
SDK Ruby 3.x 用	あり	
SDK Rust 用	なし	
SDK Swift 用	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	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
ARC Zonal Shift	a: _S	AWS_ENDPOINT_URL_ARC_ZONAL_SHIFT
Artifact	a:	AWS_ENDPOINT_URL_ARTIFACT
Athena	at	AWS_ENDPOINT_URL_ATHENA
AuditManager	au ge	AWS_ENDPOINT_URL_AUDITMANAGER
Auto Scaling	au in	AWS_ENDPOINT_URL_AUTO_SCALING
Auto Scaling Plans	au in	AWS_ENDPOINT_URL_AUTO_SCALING_PLANS
b2bi	b2	AWS_ENDPOINT_URL_B2BI
Backup	ba	AWS_ENDPOINT_URL_BACKUP
Backup Gateway	ba te	AWS_ENDPOINT_URL_BACKUP_GATEWAY

serviceId	共有のサービス識別子キー AWS **co** fil	**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 ur	AWS_ENDPOINT_URL_BEDROCK_RUNTIME
billingconductor	bi nd	AWS_ENDPOINT_URL_BILLINGCONDUCTOR

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Braket	br	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 _r pe	AWS_ENDPOINT_URL_CHIME_SDK_MEDIA_PIPELINES
Chime SDK Meetings	ch _r	AWS_ENDPOINT_URL_CHIME_SDK_MEETINGS

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Chime SDK Messaging	ch _r g	AWS_ENDPOINT_URL_CHIME_SDK_MESSAGING
Chime SDK Voice	ch _v	AWS_ENDPOINT_URL_CHIME_SDK_VOICE
CleanRooms	cl s	AWS_ENDPOINT_URL_CLEANROOMS
CleanRoomsML	cl sr	AWS_ENDPOINT_URL_CLEANROOMSML
Cloud9	cl	AWS_ENDPOINT_URL_CLOUD9
CloudControl	cl rc	AWS_ENDPOINT_URL_CLOUDCONTROL
CloudDirectory	cl cf	AWS_ENDPOINT_URL_CLOUDDIRECTORY
CloudFormation	cl af	AWS_ENDPOINT_URL_CLOUDFORMATION

serviceId	共有のサービス識別子キー AWS **co** fil	**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 v2	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
CloudTrail Data	cl	AWS_ENDPOINT_URL_CLOUDTRAIL_DATA
CloudWatch	cl h	AWS_ENDPOINT_URL_CLOUDWATCH
codeartifact	co ac	AWS_ENDPOINT_URL_CODEARTIFACT
CodeBuild	co	AWS_ENDPOINT_URL_CODEBUILD
CodeCatalyst	co ys	AWS_ENDPOINT_URL_CODECATALYST
CodeCommit	co t	AWS_ENDPOINT_URL_CODECOMMIT
CodeDeploy	co y	AWS_ENDPOINT_URL_CODEDEPLOY
CodeGuru Reviewer	co re	AWS_ENDPOINT_URL_CODEGURU_REVIEWER

serviceId	共有のサービス識別子キー AWS **co** fil	**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 ir	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	共有のサービス識別子キー AWS **co** fil	**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 dn	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Connect Contact Lens	co ons	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Cost and Usage Report Service	co us o: ce	AWS_ENDPOINT_URL_COST_AND_USAGE_REPO RT_SERVICE
Customer Profiles	cu p:	AWS_ENDPOINT_URL_CUSTOMER_PROFILES
DataBrew	da	AWS_ENDPOINT_URL_DATABREW
DataExchange	da ng	AWS_ENDPOINT_URL_DATAEXCHANGE
Data Pipeline	da l:	AWS_ENDPOINT_URL_DATA_PIPELINE
DataSync	da	AWS_ENDPOINT_URL_DATASYNC
DataZone	da	AWS_ENDPOINT_URL_DATAZONE
DAX	da	AWS_ENDPOINT_URL_DAX

serviceId	共有のサービス識別子キー AWS **co** fil	**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 or 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	共有のサービス識別子キー AWS **co** fil	**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 ne ct	AWS_ENDPOINT_URL_EC2_INSTANCE_CONNECT
ECR	ec	AWS_ENDPOINT_URL_ECR

serviceId	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Elastic Load Balancing	el oa c:	AWS_ENDPOINT_URL_ELASTIC_LOAD_BALANCING
Elastic Load Balancing v2	el oa 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	el a: ic	AWS_ENDPOINT_URL_ELASTICSEARCH_SERVICE

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

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
FreeTier	fr	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	共有のサービス識別子キー AWS **co** fil	**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 ox	AWS_ENDPOINT_URL_IMPORTEXPORT

serviceId	共有のサービス識別子キー AWS **co** fil	**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 or	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
IoT 1Click Devices Service	ic k_ _S	AWS_ENDPOINT_URL_IOT_1CLICK_DEVICES_ SERVICE
IoT 1Click Projects	ic k_ s	AWS_ENDPOINT_URL_IOT_1CLICK_PROJECTS
IoTAnalytics	ic ic	AWS_ENDPOINT_URL_IOTANALYTICS
IotDeviceAdvisor	ic ac	AWS_ENDPOINT_URL_IOTDEVICEADVISOR
IoT Events	ic s	AWS_ENDPOINT_URL_IOT_EVENTS
IoT Events Data	ic s_	AWS_ENDPOINT_URL_IOT_EVENTS_DATA
IoTFleetHub	ic ul	AWS_ENDPOINT_URL_IOTFLEETHUB

serviceId	共有のサービス識別子キー AWS **co** fil	**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 ir	AWS_ENDPOINT_URL_IVS_REALTIME

serviceId	共有のサービス識別子キー AWS **co** fil	**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 nl	AWS_ENDPOINT_URL_KENDRA_RANKING
Keyspaces	ke	AWS_ENDPOINT_URL_KEYSPACES
Kinesis	ki	AWS_ENDPOINT_URL_KINESIS
Kinesis Video Archived Media	ki ic iv a	AWS_ENDPOINT_URL_KINESIS_VIDEO_ARCHIVED_MEDIA

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Kinesis Video Media	k: i a	AWS_ENDPOINT_URL_KINESIS_VIDEO_MEDIA
Kinesis Video Signaling	k: i a	AWS_ENDPOINT_URL_KINESIS_VIDEO_SIGNALING
Kinesis Video WebRTC Storage	k: i t e	AWS_ENDPOINT_URL_KINESIS_VIDEO_WEBRTC_STORAGE
Kinesis Analytics	k: n	AWS_ENDPOINT_URL_KINESIS_ANALYTICS
Kinesis Analytics V2	k: n v	AWS_ENDPOINT_URL_KINESIS_ANALYTICS_V2
Kinesis Video	k: i	AWS_ENDPOINT_URL_KINESIS_VIDEO

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
KMS	kr	AWS_ENDPOINT_URL_KMS
LakeFormation	la t:	AWS_ENDPOINT_URL_LAKEFORMATION
Lambda	la	AWS_ENDPOINT_URL_LAMBDA
Launch Wizard	la za	AWS_ENDPOINT_URL_LAUNCH_WIZARD
Lex Model Building Service	le _l _s	AWS_ENDPOINT_URL_LEX_MODEL_BUILDING_ SERVICE
Lex Runtime Service	le me e	AWS_ENDPOINT_URL_LEX_RUNTIME_SERVICE
Lex Models V2	le s_	AWS_ENDPOINT_URL_LEX_MODELS_V2
Lex Runtime V2	le me	AWS_ENDPOINT_URL_LEX_RUNTIME_V2

serviceId	共有のサービス識別子キー AWS **co** fil	**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_LIN UX_SUBSCRIPTIONS
License Manager User Subscriptions	l: ar e: ip	AWS_ENDPOINT_URL_LICENSE_MANAGER_USE R_SUBSCRIPTIONS
Lightsail	l:	AWS_ENDPOINT_URL_LIGHTSAIL
Location	l(	AWS_ENDPOINT_URL_LOCATION
CloudWatch Logs	c: h_	AWS_ENDPOINT_URL_CLOUDWATCH_LOGS
CloudWatch Logs	c: h_	AWS_ENDPOINT_URL_CLOUDWATCH_LOGS

serviceId	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Marketplace Agreement	ma ce er	AWS_ENDPOINT_URL_MARKETPLACE_AGREEMENT
Marketplace Catalog	ma ce g	AWS_ENDPOINT_URL_MARKETPLACE_CATALOG
Marketplace Deployment	ma ce me	AWS_ENDPOINT_URL_MARKETPLACE_DEPLOYMENT
Marketplace Entitlement Service	ma ce er vi	AWS_ENDPOINT_URL_MARKETPLACE_ENTITLEMENT_SERVICE
Marketplace Commerce Analytics	ma ce ce id	AWS_ENDPOINT_URL_MARKETPLACE_COMMERCE_ANALYTICS

serviceId	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**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: _I	AWS_ENDPOINT_URL_MIGRATION_HUB
mgn	m	AWS_ENDPOINT_URL_MGN
Migration Hub Refactor Spaces	m: _I ct es	AWS_ENDPOINT_URL_MIGRATION_HUB_REFAC TOR_SPACES

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
MigrationHub Config	mi hu g	AWS_ENDPOINT_URL_MIGRATIONHUB_CONFIG
MigrationHubOrchestrator	mi hu t:	AWS_ENDPOINT_URL_MIGRATIONHUBORCHESTRATOR
MigrationHubStrategy	mi hu gy	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Neptune Graph	ne ra	AWS_ENDPOINT_URL_NEPTUNE_GRAPH
neptunedata	ne ta	AWS_ENDPOINT_URL_NEPTUNEDATA
Network Firewall	ne i:	AWS_ENDPOINT_URL_NETWORK_FIREWALL
NetworkManager	ne na	AWS_ENDPOINT_URL_NETWORKMANAGER
NetworkMonitor	ne n:	AWS_ENDPOINT_URL_NETWORKMONITOR
nimble	n:	AWS_ENDPOINT_URL_NIMBLE
OAM	oa	AWS_ENDPOINT_URL_OAM
Omics	or	AWS_ENDPOINT_URL_OMICS
OpenSearch	op h	AWS_ENDPOINT_URL_OPENSEARCH

serviceId	共有のサービス識別子キー AWS **co** fil	**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 id	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	共有のサービス識別子キー AWS **co** fil	**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	pi	AWS_ENDPOINT_URL_PI

serviceId	共有のサービス識別子キー AWS **co** fil	**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: tv	AWS_ENDPOINT_URL_PRIVATENETWORKS
Proton	p:	AWS_ENDPOINT_URL_PROTON

serviceId	共有のサービス識別子キー AWS **co** fil	**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	ql ic	AWS_ENDPOINT_URL_QLDB_SESSION
QuickSight	qu t	AWS_ENDPOINT_URL_QUICKSIGHT
RAM	ra	AWS_ENDPOINT_URL_RAM
rbin	rb	AWS_ENDPOINT_URL_RBIN
RDS	rd	AWS_ENDPOINT_URL_RDS
RDS Data	rd	AWS_ENDPOINT_URL_RDS_DATA
Redshift	re	AWS_ENDPOINT_URL_REDSHIFT

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Redshift Data	re da	AWS_ENDPOINT_URL_REDSHIFT_DATA
Redshift Serverless	re se s	AWS_ENDPOINT_URL_REDSHIFT_SERVERLESS
Rekognition	re on	AWS_ENDPOINT_URL_REKOGNITION
repostspace	re ce	AWS_ENDPOINT_URL_REPOSTSPACE
resiliencehub	re eh	AWS_ENDPOINT_URL_RESILIENCEHUB
Resource Explorer 2	re ex 2	AWS_ENDPOINT_URL_RESOURCE_EXPLORER_2
Resource Groups	re gr	AWS_ENDPOINT_URL_RESOURCE_GROUPS

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Resource Groups Tagging API	re g: g(	AWS_ENDPOINT_URL_RESOURCE_GROUPS_TAG GING_API
RoboMaker	ro	AWS_ENDPOINT_URL_ROBOMAKER
RolesAnywhere	ro he	AWS_ENDPOINT_URL_ROLESANYWHERE
Route 53	ro	AWS_ENDPOINT_URL_ROUTE_53
Route53 Recovery Cluster	ro ec lu	AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CLUSTER
Route53 Recovery Control Config	ro ec or n1	AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CO NTROL_CONFIG

serviceId	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**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 FeatureSt ore Runtime	sa _f to in	AWS_ENDPOINT_URL_SAGEMAKER_FEATUREST ORE_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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
savingsplans	sa ar	AWS_ENDPOINT_URL_SAVINGSPLANS
Scheduler	sc	AWS_ENDPOINT_URL_SCHEDULER
schemas	sc	AWS_ENDPOINT_URL_SCHEMAS
SimpleDB	sd	AWS_ENDPOINT_URL_SIMPLEDB
Secrets Manager	se ar	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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
Service Quotas	se uc	AWS_ENDPOINT_URL_SERVICE_QUOTAS
Service Catalog	se at	AWS_ENDPOINT_URL_SERVICE_CATALOG
Service Catalog AppRegistry	se at pi	AWS_ENDPOINT_URL_SERVICE_CATALOG_APP REGISTRY
ServiceDiscovery	se sd	AWS_ENDPOINT_URL_SERVICEDISCOVERY
SES	se	AWS_ENDPOINT_URL_SES
SESv2	se	AWS_ENDPOINT_URL_SESV2
Shield	sh	AWS_ENDPOINT_URL_SHIELD
signer	si	AWS_ENDPOINT_URL_SIGNER

serviceId	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
SimSpaceWeaver	s: ea	AWS_ENDPOINT_URL_SIMSPACEWEAVER
SMS	sr	AWS_ENDPOINT_URL_SMS
Snow Device Management	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	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
synthetics	sy s	AWS_ENDPOINT_URL_SYNTHETICS
Textract	te	AWS_ENDPOINT_URL_TEXTRACT
Timestream InfluxDB	ti m_ b	AWS_ENDPOINT_URL_TIMESTREAM_INFLUXDB
Timestream Query	ti m_	AWS_ENDPOINT_URL_TIMESTREAM_QUERY
Timestream Write	ti m_	AWS_ENDPOINT_URL_TIMESTREAM_WRITE
tnb	tn	AWS_ENDPOINT_URL_TNB
Transcribe	ti e	AWS_ENDPOINT_URL_TRANSCRIBE
Transfer	ti	AWS_ENDPOINT_URL_TRANSFER

serviceId	共有のサービス識別子キー AWS **co** fil	**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	共有のサービス識別子キー AWS **co** fil	**AWS_ENDPOINT_URL_<SERVICE>** 環境変数
WellArchitected	we te	AWS_ENDPOINT_URL_WELLARCHITECTED
Wisdom	wi	AWS_ENDPOINT_URL_WISDOM
WorkDocs	wo	AWS_ENDPOINT_URL_WORKDOCS
WorkLink	wo	AWS_ENDPOINT_URL_WORKLINK
WorkMail	wo	AWS_ENDPOINT_URL_WORKMAIL
WorkMailMessageFlow	wo es w	AWS_ENDPOINT_URL_WORKMAILMESSAGEFLOW
WorkSpaces	wo s	AWS_ENDPOINT_URL_WORKSPACES
WorkSpaces Thin Client	wo s_ ie	AWS_ENDPOINT_URL_WORKSPACES_THIN_CLIENT

serviceId	共有のサービス識別子キー AWS **co** fil	**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

スマート設定のデフォルト機能では、 AWS SDKs は、他の構成設定用に事前定義された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 - 共有 AWS **config** 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。の場合 AWS SDK 設定には明示的に値が設定されており、その値は常に優先されます。の場合 AWS SDK 設定には明示的に設定された値はなく、レガシーと等しくないため、この機能defaults_modeはアプリケーションに最適化されたさまざまな設定に対して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、サービスリージョンエンドポイント設定、および場合によっては SDK関連の設定が含まれます。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 と等しくない場合はlegacy、 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実行することをお勧めします。これはSDK、ベストプラクティスが進化するにつれて、提供される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: の新しいメジャーバージョンSDKsはデフォルトで になりますstandard。

有効な値:

- legacy – の確立前に存在SDKしていた および によって異なるデフォルト設定を提供しますdefaults_mode。
- 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 – 標準モードに基づいて構築され、 を呼び出す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 のサービス 同じ 内から AWS リージョン.
- cross-region – 標準モードに基づいて構築され、 を呼び出す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 のサービス 別のリージョンにある。
- 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。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SDK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、インスタンスメタデータ がクエリされ、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再試行方法を指定します。「再試行動作」を参照してください。
- stsRegionalEndpoints – が をどのようにSDK判断するかを指定します。 AWS のサービスとの通信に使用する エンドポイント AWS Security Token Service (AWS STS) 。 「」を参照してくださいAWS STS リージョンエンドポイント。

- s3UsEast1RegionalEndpoints – が をどのようにSDK判断するかを指定します。 AWS us-east-1 リージョンの Amazon S3 と通信するために使用する サービスエンドポイント。
- 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 — CLIENTHELLOメッセージが送信された時点から、クライアントとサーバーが完全に暗号をネゴシエートし、キーを交換した時点まで、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。

との互換性 AWS SDKs

このトピックで説明されている機能と設定SDKsを以下に示します。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、 でサポートされています。 AWS SDK for Java と AWS SDK for Kotlin のみ。

SDK	サポート	注意または詳細情報
AWS CLI v2	なし	
SDK C++ 用	あり	最適化されていないパラメーター：stsRegionalEndpoints 、s3UsEast1RegionalEndpoints 、tlsNegotiationTimeoutInMillis 。
SDK Go V2 用 (1.x)	あり	最適化されていないパラメーター：retryMode 、stsRegionalEndpoints 、s3UsEast1RegionalEndpoints 。
SDK Go 1.x (V1) 用	なし	
SDK for Java 2.x	あり	最適化されていないパラメーター：stsRegionalEndpoints 。
SDK for Java 1.x	なし	
SDK JavaScript 3.x 用	あり	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。
SDK JavaScript 2.x 用	なし	
SDK Kotlin 用	なし	
SDK の 。NET 3.x	あり	最適化されていないパラメーター：connectTimeoutInMillis 、tlsNegotiationTimeoutInMillis 。
SDK 3.x PHP 用	あり	最適化されていないパラメーター：tlsNegotiationTimeoutInMillis 。
SDK for Python (Boto3)	あり	最適化されていないパラメーター：tlsNegotiationTimeoutInMillis 。
SDK Ruby 3.x 用	あり	
SDK Rust 用	なし	
SDK Swift 用	なし	

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

AWS 共通ランタイム (CRT) ライブラリ

AWS Common Runtime (CRT) ライブラリは、 のベースライブラリですSDKs。CRT は、C で記述された独立したパッケージのモジュールファミリーです。各パッケージは、さまざまな必要な機能に対して優れたパフォーマンスと最小限のフットプリントを提供します。これらの機能は共通であり、すべての で共有されるため、コードSDKs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 クライアント側の認証 (標準認証情報プロバイダーと署名 (sigv4))
- awslabs/aws-c-cal: 暗号化プリミティブタイプ、ハッシュ (MD5、SHA256、SHA256HMAC) 、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/trailer)、イベントストリームを介したリモートプロシージャコール (RPC) の実装
- awslabs/aws-c-http: HTTP/1.1 および HTTP/2 仕様の C99 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: AWS IoT クラウドサービスとデバイスの統合の C99 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: 高帯域幅の Amazon インスタンスのスループットを最大化するように設計された Amazon S3 サービスと通信するための C99 ライブラリの実装 EC2
- 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: クロスプラットフォームCRC32のハードウェアアクセラレーションCRC32cと効率的なソフトウェア実装へのフォールバック
- 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 Cryptography チームが AWS とその顧客のために管理する汎用暗号化ライブラリ
- awslabs/s2n: TLS/SSL プロトコルの C99 実装。セキュリティを優先して、小さく高速に設計

CRT は Go と Rust SDKsを除くすべての から使用できます。

CRT 依存関係

CRT ライブラリは、関係と依存関係の複雑な網を形成します。これらの関係を知ることは、ソースCRTから直接 を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、言語 SDK (C++ や AWS SDK Java の場合など AWS SDK) または言語 IoT デバイス SDK (C++ や AWS IoT Java SDK用の AWS IoT など) SDK を介してCRT機能にアクセスします。次の図では、言語CRTバインディングボックスは、特定の言語 のCRTライブラリをラップするパッケージを指しますSDK。これは 形式のパッケージのコレクションでaws-crt-*、「*」はSDK言語 (aws-crt-cppや など) ですaws-crt-java。

ライブラリの階層依存関係を次に示しますCRT。

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。

AWS SDKs およびツールバージョンのサポート

次の表は、使用可能な のリストを示しています。 AWS Software Development Kit (SDK) のメジャーバージョンと、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ます。のメジャーバージョンのライフサイクルの詳細については、「」を参照してください。AWS SDKs および ツールとその基盤となる依存関係については、「」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 C++ 用	1.x	一般提供	9/2/2015	
SDK Go V2 用	V2 1.x	一般提供	1/19/2021	
SDK Go 用	1.x	メンテナンス	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	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	
SDK の JavaScript	1.x	サポートの終了	5/6/2013	
SDK の JavaScript	2.x	メンテナンス	6/19/2014	詳細と日付については、「お知

SDK	メジャーバージョン	現在のフェーズ	一般提供日	メモ
				らせ」を参照してください。
SDK の JavaScript	3.x	一般提供	12/15/2020	
SDK Kotlin 用	1.x	一般提供	11/27/2023	
SDK の 。NET	1.x	サポートの終了	2009 年 11 月	
SDK の 。NET	2.x	サポートの終了	11/8/2013	
SDK の 。NET	3.x	一般提供	7/28/2015	
SDK の PHP	2.x	サポートの終了	11/2/2012	
SDK の 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 Ruby 用	1.x	サポートの終了	7/14/2011	
SDK Ruby 用	2.x	サポートの終了	2/15/2015	
SDK Ruby 用	3.x	一般提供	8/29/2017	
SDK Rust 用	1.x	一般提供	11/27/2023	
SDK Swift 用	1.x	一般提供	9/17/2024	

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	2.x	サポートの終了	11/8/2013	
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	

言及されていない SDKまたはツールを検索しますか？ 例えばSDKs、暗号化 SDKs、IoT デバイス SDKs、モバイル は、このガイドには含まれていません。これらの他のツールに関するドキュメントについては、「 で構築するツール」を参照してください。 AWS.

のドキュメント履歴 AWS SDKs および ツールリファレンスガイド

次の表は、 への重要な追加と更新を示しています。 AWS SDKs および ツールリファレンスガイド。このドキュメントの更新に関する通知については、RSSフィードをサブスクライブできます。

変更	説明	日付
設定リファレンスSDKへのSwift の追加	Swift SDK サポートをすべての 設定リファレンスに追加する との互換性 AWS SDKs テーブル。	2024 年 9 月 17 日
SDK for Java 1.x システムプロパティ	でサポートされているJVMシステム設定の詳細を追加する AWS SDK for Java 1.x。	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、IAMIdentity 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 更新。	Amazon のサポートを追加しましたEKS。IMDSv1 フォールバックを無効にする設定を追加。	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	for KotlinSDK、 SDK for SDK Rust、および を含めるように更新された および ツール機能の互換性テーブル AWS Tools for PowerShell.	2023 年 12 月 10 日

認証の更新	SDKs および ツールでサポートされている認証方法の更新。	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合わせてガイドを更新しました。詳細については、「 のセキュリティのベストプラクティスIAM」を参照してください。	2023 年 2 月 27 日
SSO 更新	新しいSSOトークン設定のSSO認証情報の更新。	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認証情報の明確さの更新。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	これを反映するための更新 AWS SSO は現在、 と呼ばれています。 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。