

Reference Guide

Amazon SDKs and Tools



Amazon SDKs and Tools Reference Guide **Amazon SDKs and Tools: Reference Guide**

Table of Contents

Amazon SDKs and Tools Reference Guide	1
Configuration	3
Shared config and credentials files	4
Profiles	4
Format of the config file	6
Format of the credentials file	9
Location of the shared files	9
Home directory resolution	10
Change the default location of these files	11
Environment variables	12
How to set environment variables	12
Serverless environment variable setup	13
JVM system properties	14
How to set JVM system properties	14
Authentication and access	16
Amazon Builder ID	18
IAM Identity Center authentication	18
Prerequisites	19
Configure programmatic access using IAM Identity Center	19
Refreshing portal access sessions	22
Understand IAM Identity Center authentication	22
IAM Roles Anywhere	26
Step 1: Configure IAM Roles Anywhere	26
Step 2: Use IAM Roles Anywhere	27
Assume a role	28
Assume an IAM role	29
Assume a role (web)	30
Federate with web identity or OpenID Connect	31
Amazon access keys	32
Use short-term credentials	32
Use long-term credentials	33
Short-term credentials	34
Long-term credentials	36
IAM roles for EC2 instances	39

	Create an IAM role	39
	Launch an Amazon EC2 instance and specify your IAM role	40
	Connect to the EC2 instance	. 40
	Run your application on the EC2 instance	40
	Trusted identity propagation	. 41
	Prerequisites for using the TIP plugin	. 41
	To use the TIP plugin in your code	. 42
Se	ettings reference	46
	Creating service clients	46
	Precedence of settings	. 46
	Understanding the settings pages of this guide	47
	Config file settings list	49
	Credentials file settings list	53
	Environment variables list	. 53
	JVM system properties list	. 58
	Standardized credential providers	. 61
	Understand the credential provider chain	. 62
	SDK-specific and tool-specific credential provider chains	. 63
	Amazon access keys	64
	Assume role provider	67
	Container provider	. 74
	IAM Identity Center provider	. 78
	IMDS provider	. 84
	Process provider	
	Standardized features	93
	Account-based endpoints	. 94
	Application ID	
	Amazon EC2 instance metadata	
	Amazon S3 access points	101
	Amazon S3 Multi-Region Access Points	
	Amazon Web Services Region	106
	Amazon STS Regional endpoints	
	Data Integrity Protections	
	Dual-stack and FIPS endpoints	119
	Endpoint discovery	122
	General configuration	124

Host prefix injection	128
IMDS client	132
Retry behavior	135
Request compression	141
Service-specific endpoints	143
Smart configuration defaults	
Common Runtime 1	194
CRT dependencies	195
Maintenance policy	196
Overview	196
Versioning	196
SDK major version lifecycle	196
Dependency lifecycle	197
Communication methods	198
Version lifecycle	199
Document history	

What is covered in the Amazon SDKs and Tools Reference Guide

Many SDKs and tools share some common functionality, either through shared design specifications or through a shared library.

This guide includes information regarding:

- Globally configuring Amazon SDKs and tools How to use the shared config and credentials files or environment variables to configure your Amazon SDKs and tools.
- <u>Authentication and access using Amazon SDKs and tools</u> Establish how your code or tool
 authenticates with Amazon when you develop with Amazon Web Services services.
- <u>Amazon SDKs and tools settings reference</u> Reference for all standardized settings available for authentication and configuration.
- <u>Amazon Common Runtime (CRT) libraries</u> Overview of the shared Amazon Common Runtime (CRT) libraries that are available to almost all SDKs.
- <u>Amazon SDKs and Tools maintenance policy</u> covers the maintenance policy and versioning for Amazon Software Development Kits (SDKs) and tools, including Mobile and Internet of Things (IoT) SDKs, and their underlying dependencies.

This Amazon SDKs and Tools Reference Guide is intended to be a base of information that is applicable to multiple SDKs and tools. The specific guide for the SDK or tool that you are using should be used in addition to any information presented here. The following are the SDK and tools which have relevant sections of material in this guide:

If you are using:	This guide's relevant sections for you are:
Any SDK or tool	Amazon SDKs and Tools maintenance policy
Amazon Cloud9Amazon CDK	Globally configuring Amazon SDKs and tools
Amazon Toolkit for Azure DevOps	Authentication and access using Amazon SDKs and tools
 Amazon Toolkit for JetBrains Amazon Toolkit for Visual Studio	Amazon SDKs and Tools maintenance policy

If you are using:	This guide's relevant sections for you are:
 Amazon Toolkit for Visual Studio Code Amazon Serverless Application Model Amazon CodeArtifact Amazon CodeBuild Amazon CodeCatalyst Amazon CodeCommit Amazon CodeDeploy 	
 Amazon CodePipeline Amazon CLI Amazon SDK for C++ 	Globally configuring Amazon SDKs and tools
 Amazon SDK for Go Amazon SDK for Java Amazon SDK for JavaScript 	Authentication and access using Amazon SDKs and tools Amazon SDKs and tools settings reference
 Amazon SDK for Kotlin Amazon SDK for .NET Amazon SDK for PHP Amazon SDK for Python (Boto3) 	Amazon Common Runtime (CRT) libraries Amazon SDKs and Tools maintenance policy Amazon SDKs and Tools version lifecycle
 Amazon SDK for Ruby Amazon SDK for Rust Amazon SDK for Swift Amazon Tools for Windows PowerShell 	

- For an overview of tools that can help you develop applications on Amazon, see <u>Tools to Build on</u> Amazon.
- For information on support, see the <u>Amazon Knowledge Center</u>.
- For Amazon terminology, see the <u>Amazon glossary</u> in the *Amazon Web Services Glossary Reference*.

Globally configuring Amazon SDKs and tools

With Amazon SDKs and other Amazon developer tools, such as the Amazon Command Line Interface (Amazon CLI), you can interact with Amazon service APIs. Before attempting that, however, you must configure the SDK or tool with the information that it needs to perform the requested operation.

This information includes the following items:

- **Credentials information** that identifies who is calling the API. The credentials are used to encrypt the request to the Amazon servers. Using this information, Amazon confirms your identity and can retrieve permissions policies associated with it. Then it can determine what actions you're allowed to perform.
- Other configuration details that you use to tell the Amazon CLI or SDK how to process the request, where to send the request (to which Amazon service endpoint), and how to interpret or display the response.

Each SDK or tool supports multiple sources that you can use to supply the required credential and configuration information. Some sources are unique to the SDK or tool, and you must refer to the documentation for that tool or SDK for the details on how to use that method.

However, the Amazon SDKs and tools support common settings from primary sources beyond the code itself. This section covers the following topics:

Topics

- Using shared config and credentials files to globally configure Amazon SDKs and tools
- <u>Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools</u>
- Using environment variables to globally configure Amazon SDKs and tools
- <u>Using JVM system properties to globally configure Amazon SDK for Java and Amazon SDK for Kotlin</u>

Using shared config and credentials files to globally configure Amazon SDKs and tools

The shared Amazon config and credentials files are the most common way that you can specify authentication and configuration to an Amazon SDK or tool.

The shared config and credentials files contain a set of profiles. A profile is a set of configuration settings, in key-value pairs, that is used by Amazon SDKs, the Amazon Command Line Interface (Amazon CLI), and other tools. Configuration values are attached to a profile in order to configure some aspect of the SDK/tool when that profile is used. These files are "shared" in that the values take affect for any applications, processes, or SDKs on the local environment for a user.

Both the shared config and credentials files are plaintext files that contain only ASCII characters (UTF-8 encoded). They take the form of what are generally referred to as INI files.

Profiles

Settings within the shared config and credentials files are associated with a specific profile. Multiple profiles can be defined within the file to create different setting configurations to apply in different development environments.

The [default] profile contains the values that are used by an SDK or tool operation if a specific named profile is not specified. You can also create separate profiles that you can explicitly reference by name. Each profile can use different settings and values as needed by your application and scenario.



Note

[default] is simply an unnamed profile. This profile is named default because it is the default profile used by the SDK if the user does not specify a profile. It does not provide inherited default values to other profiles. If you set something in the [default] profile and you don't set it in a named profile, then the value isn't set when you use the named profile.

Set a named profile

The [default] profile and multiple named profiles can exist in the same file. Use the following setting to select which profile's settings are used by your SDK or tool when running your code. Profiles can also be selected within code, or per-command when working with the Amazon CLI.

Configure this functionality by setting one of the following:

AWS_PROFILE - environment variable

When this environment variable is set to a named profile or "default", all SDK code and Amazon CLI commands use the settings in that profile.

Linux/macOS example of setting environment variables via command line:

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

Windows example of setting environment variables via command line:

```
setx AWS_PROFILE "my_default_profile_name"
```

aws.profile - JVM system property

For SDK for Kotlin on the JVM and the SDK for Java 2.x, you can <u>set the aws.profile system</u> <u>property</u>. When the SDK creates a service client, it uses the settings in the named profile unless the setting is overridden in code. The SDK for Java 1.x does not support this system property.

Note

If your application is on a server running multiple applications, we recommend you always used named profiles rather than the default profile. The default profile is automatically picked up by any Amazon application in the environment and is shared amongst them. Thus, if someone else updates the default profile for their application it can unintentionally impact the others. To safeguard against this, define a named profile in the shared config file and then use that named profile in your application by setting the named profile in your code. You can use the environment variable or JVM system property to set the named profile if you know that it's scope only affects your application.

Profiles 5

Format of the config file

The config file is organized into sections. A section is a named collection of settings, and continues until another section definition line is encountered.

The config file is a plaintext file that uses the following format:

- All entries in a section take the general form of setting-name=value.
- Lines can be commented out by starting the line with a hashtag character (#).

Section types

A section definition is a line that applies a name to a collection of settings. Section definition lines start and end with square brackets ([]). Inside the brackets, there is a section type identifier and a custom name for the section. You can use letters, numbers, hyphens (-), and underscores (_), but no spaces.

Section type: default

Example section definition line: [default]

[default] is the only profile that does not require the profile section identifier.

The following example shows a basic config file with a [default] profile. It sets the <u>region</u> setting. All settings that follow this line, up until another section definition is encountered, are part of this profile.

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

Section type: profile

Example section definition line: [profile dev]

The profile section definition line is a named configuration grouping that you can apply for different development scenarios. To better understand named profiles, see the preceding section on Profiles.

Format of the config file 6

The following example shows a config file with a profile section definition line and a named profile called foo. All settings that follow this line, up until another section definition is encountered, are part of this named profile.

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

Some settings have their own nested group of subsettings, such as the s3 setting and subsettings in the following example. Associate the subsettings with the group by indenting them by one or more spaces.

```
[profile test]
region = us-west-2
s3 =
   max_concurrent_requests=10
   max_queue_size=1000
```

Section type: sso-session

Example section definition line: [sso-session my-sso]

The sso-session section definition line names a group of settings that you use to configure a profile to resolve Amazon credentials using Amazon IAM Identity Center. For more information on configuring single sign-on authentication, see <u>Using IAM Identity Center to authenticate</u>

<u>Amazon SDK and tools</u>. A profile is linked to a sso-session section by a key-value pair where sso-session is the key and the name of your sso-session section is the value, such as sso-session = <name-of-sso-session-section>.

The following example configures a profile that will get short-term Amazon credentials for the "SampleRole" IAM role in the "111122223333" account using a token from the "my-sso". The "my-sso" sso-session section is referenced in the profile section by name using the sso-session key.

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

Format of the config file 7

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

Section type: services

Example section definition line: [services *dev*]



Note

The services section supports service-specific endpoint customizations and is only available in SDKs and tools that include this feature. To see if this feature is available for your SDK, see Support by Amazon SDKs and tools for service-specific endpoints.

The services section definition line names a group of settings that configures custom endpoints for Amazon Web Services service requests. A profile is linked to a services section by a keyvalue pair where services is the key and the name of your services section is the value, such as services = <name-of-services-section>.

The services section is further separated into subsections by <SERVICE> = lines, where <SERVICE> is the Amazon Web Services service identifier key. The Amazon Web Services service identifier is based on the API model's serviceId by replacing all spaces with underscores and lowercasing all letters. For a list of all service identifier keys to use in the services section, see Identifiers for service-specific endpoints. The service identifier key is followed by nested settings with each on its own line and indented by two spaces.

The following example uses a services definition to configure the endpoint to use for requests made only to the Amazon DynamoDB service. The "local-dynamodb" services section is referenced in the profile section by name using the services key. The Amazon Web Services service identifier key is dynamodb. The Amazon DynamoDB service subsection begins on the line dynamodb = . Any immediately following lines that are indented are included in that subsection and apply to that service.

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

Format of the config file

For more information on custom endpoint configuration, see Service-specific endpoints.

Format of the credentials file

The rules for the credentials file are generally identical to those for the config file, except that profile sections don't begin with the word profile. Use only the profile name itself between square brackets. The following example shows a credentials file with a named profile section called foo.

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

Only the following settings that are considered "secrets" or sensitive can be stored in the credentials file: aws_access_key_id,aws_secret_access_key, and aws_session_token. Although these settings can alternatively be placed in the shared config file, we recommend that you keep these sensitive values in the separate credentials file. This way, you can provide separate permissions for each file, if necessary.

The following example shows a basic credentials file with a [default] profile. It sets the aws_access_key_id,aws_secret_access_key, and aws_session_token global settings.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTR1
```

Regardless of whether you use a named profile or "default" in your credentials file, any settings here will be combined with any settings from your config file that uses the same profile name. If there are credentials in both files for a profile sharing the same name, the keys in the credentials file take precedence.

Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools

The shared Amazon config and credentials files are plaintext files that hold configuration information for the Amazon SDKs and tools. The files reside locally in your environment and used automatically by SDK code or by Amazon CLI commands that you run in that environment.

Format of the credentials file

For example, on your own computer or when developing on an Amazon Elastic Compute Cloud instance.

When the SDK or tool runs, it checks for these files and loads any available configuration settings. If the files do not already exist, then a basic file is automatically created by the SDK or tool.

By default, the files are in a folder named .aws that is placed in your home or user folder.

Operating system	Default location and name of files
Linux and macOS	~/.aws/config
	~/.aws/credentials
Windows	%USERPROFILE%\.aws\config
	%USERPROFILE%\.aws\credentials

Home directory resolution

- ~ is only used for home directory resolution when it:
- Starts the path
- Is followed immediately by / or a platform specific separator. On windows, ~/ and ~\ both resolve to the home directory.

When determining the home directory, the following variables are checked:

- (All platforms) The HOME environment variable
- (Windows platforms) The USERPROFILE environment variable
- (Windows platforms) The concatenation of HOMEDRIVE and HOMEPATH environment variables (\$HOMEDRIVE\$HOMEPATH)
- (Optional per SDK or tool) An SDK or tool-specific home path resolution function or variable

When possible, if a user's home directory is specified at the start of the path (for example, ~username/), it is resolved to the requested user name's home directory (for example, /home/username/.aws/config).

Home directory resolution 10

Change the default location of these files

You can use any of the following to override where these files are loaded from by the SDK or tool.

Use environment variables

The following environment variables can be set to change the location or name of these files from the default to a custom value:

- config file environment variable: AWS_CONFIG_FILE
- credentials file environment variable: AWS_SHARED_CREDENTIALS_FILE

Linux/macOS

You can specify an alternate location by running the following <u>export</u> commands on Linux or macOS.

```
$ 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

You can specify an alternate location by running the following setx commands on Windows.

```
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
```

For more information on configuring your system using environment variables, see <u>Using</u> environment variables to globally configure Amazon SDKs and tools.

Use JVM system properties

For the SDK for Kotlin running on the JVM and for SDK for Java 2.x, you can set the following JVM system properties to change the location or name of these files from the default to a custom value:

- config file JVM system property: aws.configFile
- credentials file environment variable: aws.sharedCredentialsFile

For instructions on how to set JVM system properties, see the section called "How to set JVM system properties". The SDK for Java 1.x does not support these system properties.

Using environment variables to globally configure Amazon SDKs and tools

Environment variables provide another way to specify configuration options and credentials when using Amazon SDKs and tools. Environment variables can be useful for scripting or temporarily setting a named profile as the default. For the list of environment variables supported by most SDKs, see Environment variables list.

Precedence of options

- If you specify a setting by using its environment variable, it overrides any value loaded from a profile in the shared Amazon config and credentials files.
- If you specify a setting by using a parameter on the Amazon CLI command line, it overrides any value from either the corresponding environment variable or a profile in the configuration file.

How to set environment variables

The following examples show how you can configure environment variables for the default user.

Linux, OS X, or Unix

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export
AWS_SESSION_TOKEN=AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
$ export AWS_REGION=us-west-2
```

Setting the environment variable changes the value used until the end of your shell session, or until you set the variable to a different value. You can make the variables persistent across future sessions by setting them in your shell's startup script.

Windows Command Prompt

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Environment variables 12

```
C:\> setx
AWS_SESSION_TOKEN AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
C:\> setx AWS_REGION us-west-2
```

Using <u>set</u> to set an environment variable changes the value used until the end of the current Command Prompt session, or until you set the variable to a different value. Using <u>setx</u> to set an environment variable changes the value used in both the current Command Prompt session and all Command Prompt sessions that you create after running the command. It does *not* affect other command shells that are already running at the time you run the command.

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...zrkuWJ0gQs8IZZaIv2BXIa2R40I

PS C:\> $Env:AWS_REGION="us-west-2"
```

If you set an environment variable at the PowerShell prompt as shown in the previous examples, it saves the value for only the duration of the current session. To make the environment variable setting persistent across all PowerShell and Command Prompt sessions, store it by using the **System** application in **Control Panel**. Alternatively, you can set the variable for all future PowerShell sessions by adding it to your PowerShell profile. See the <u>PowerShell documentation</u> for more information about storing environment variables or persisting them across sessions.

Serverless environment variable setup

If you use a serverless architecture for development, you have other options for setting environment variables. Depending on your container, you can use different strategies for code running in those containers to see and access environment variables, similar to non-cloud environments.

For example, with Amazon Lambda, you can directly set environment variables. For details, see Using Amazon Lambda environment variables in the *Amazon Lambda Developer Guide*.

In Serverless Framework, you can often set SDK environment variables in the serverless.yml file under the provider key under the environment setting. For information on the serverless.yml file, see General function settings in the Serverless Framework documentation.

Regardless of which mechanism you use to set container environment variables, there are some that are reserved by the container, such as those documented for Lambda at <u>Defined runtime</u> <u>environment variables</u>. Always consult the official documentation for the container that you're using to determine how environment variables are treated and whether there are any restrictions.

Using JVM system properties to globally configure Amazon SDK for Java and Amazon SDK for Kotlin

JVM system properties provide another way to specify configuration options and credentials for SDKs that run on the JVM such as the Amazon SDK for Java and the Amazon SDK for Kotlin. For a list of JVM system properties supported by SDKs, see <u>Settings reference</u>.

Precedence of options

- If you specify a setting by using its JVM system property, it overrides any value found in environment variables or loaded from a profile in the shared AWS config and credentials files.
- If you specify a setting by using its environment variable, it overrides any value loaded from a profile in the shared AWS config and credentials files.

How to set JVM system properties

You can set JVM system properties several ways.

On the command line

Set JVM system properties on the command-line when invoking the java command by using the -D switch. The following command configures the Amazon Web Services Region globally for all service clients unless you explicitly override the value in code.

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

If you need to set multiple JVM system properties, specify the -D switch multiple times.

With an environment variable

If you can't access the command line to invoke the JVM to run your application, you can use the JAVA_TOOL_OPTIONS environment variable to configure command-line options. This approach is

JVM system properties 14

useful in situations such as running an Amazon Lambda function on the Java runtime or running code in an embedded JVM.

The following example configures the Amazon Web Services Region globally for all service clients unless you explicitly override the value in code.

Linux, OS X, or Unix

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

Setting the environment variable changes the value used until the end of your shell session, or until you set the variable to a different value. You can make the variables persistent across future sessions by setting them in your shell's startup script.

Windows Command Prompt

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

Using set to set an environment variable changes the value used until the end of the current Command Prompt session, or until you set the variable to a different value. Using setx to set an environment variable changes the value used in both the current Command Prompt session and all Command Prompt sessions that you create after running the command. It does not affect other command shells that are already running at the time you run the command.

At runtime

You can also set JVM system properties at runtime in code by using the System.setProperty method as shown in the following example.

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

Important

Set any JVM system properties before you initialize SDK service clients, otherwise service clients may use other values.

Authentication and access using Amazon SDKs and tools

When you develop an Amazon SDK application or use Amazon tools to use Amazon Web Services services, you must establish how your code or tool authenticates with Amazon. You can configure programmatic access to Amazon resources in different ways, depending on the environment the code runs in and the Amazon access available to you.

Authentication options for code running locally (not in Amazon)

- Using IAM Identity Center to authenticate Amazon SDK and tools As a security best practice, we recommend using Amazon Organizations with IAM Identity Center to manage access across all your Amazon Web Services accounts. You can create users in Amazon IAM Identity Center, use Microsoft Active Directory, use a SAML 2.0 identity provider (IdP), or individually federate your IdP to Amazon Web Services accounts. To check if your Region supports IAM Identity Center, see Amazon IAM Identity Center endpoints and quotas in the Amazon Web Services General Reference.
- <u>Using IAM Roles Anywhere to authenticate Amazon SDKs and tools</u> You can use IAM Roles
 Anywhere to obtain temporary security credentials in IAM for workloads such as servers,
 containers, and applications that run outside of Amazon. To use IAM Roles Anywhere, your
 workloads must use X.509 certificates.
- <u>Assuming a role with Amazon credentials to authenticate Amazon SDKs and tools</u> You can assume an IAM role to temporarily access Amazon resources that you might not have access to otherwise.
- <u>Using Amazon access keys to authenticate Amazon SDKs and tools</u> Other options that might be less convenient or might increase the security risk to your Amazon resources.

Authentication options for code running within an Amazon environment

If your code runs on Amazon, credentials can be made automatically available to your application. For example, if your application is hosted on Amazon Elastic Compute Cloud, and there is an IAM role associated with that resource, the credentials are automatically made available to your application. Likewise, if you use Amazon ECS or Amazon EKS containers, the credentials set for the IAM role can be automatically obtained by the code running inside the container through the SDK's credential provider chain.

• <u>Using IAM roles to authenticate applications deployed to Amazon EC2</u> – Use IAM roles to securely run your application on an Amazon EC2 instance.

- You can programmatically interact with Amazon using IAM Identity Center in the following ways:
 - Use Amazon CloudShell to run Amazon CLI commands from the console.
 - To try cloud-based collaboration space for software development teams, consider using Amazon CodeCatalyst.

Authentication through a web-based identity provider - Mobile or client-based web applications

If you are creating mobile applications or client-based web applications that require access to Amazon, build your app so that it requests temporary Amazon security credentials dynamically by using web identity federation.

With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, app users can sign in using a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. They can receive an authentication token, and then exchange that token for temporary security credentials in Amazon that map to an IAM role with permissions to use the resources in your Amazon Web Services account.

To learn how to configure this for your SDK or tool, see <u>Assuming a role with web identity or</u> OpenID Connect to authenticate Amazon SDKs and tools.

For mobile applications, consider using Amazon Cognito. Amazon Cognito acts as an identity broker and does much of the federation work for you. For more information, see <u>Using Amazon</u> Cognito for mobile apps in the *IAM User Guide*.

More information about access management

The *IAM User Guide* has the following information about securely controlling access to Amazon resources:

- IAM Identities (users, user groups, and roles) Understand the basics of identities in Amazon.
- <u>Security best practices in IAM</u> Security recommendations to follow when developing Amazon applications according to the shared-responsibility model.

The Amazon Web Services General Reference has foundational basics on the following:

• <u>Understanding and getting your Amazon credentials</u> – Access key options and management practices for both console and programmatic access.

IAM Identity Center trusted identity propagation (TIP) plugin to access Amazon Web Services services

 <u>Using the TIP plugin to access Amazon Web Services services</u> – If you are creating an application for Amazon Q Business or other service that supports trusted identity propagation, and are using the Amazon SDK for Java or the Amazon SDK for JavaScript, you can use the TIP plugin for a streamlined authorization experience.

Amazon Builder ID

Your Amazon Builder ID complements any Amazon Web Services accounts you might already own or want to create. While an Amazon Web Services account acts as a container for Amazon resources you create and provides a security boundary for those resources, your Amazon Builder ID represents you as an individual. You can sign in with your Amazon Builder ID to access developer tools and services such as Amazon Q and Amazon CodeCatalyst.

- <u>Sign in with Amazon Builder ID</u> in the *Amazon Sign-In User Guide* Learn how to create and use an Amazon Builder ID and learn what the Builder ID provides.
- <u>CodeCatalyst concepts Amazon Builder ID</u> in the *Amazon CodeCatalyst User Guide* Learn how CodeCatalyst uses an Amazon Builder ID.

Using IAM Identity Center to authenticate Amazon SDK and tools

Amazon IAM Identity Center is the recommended method of providing Amazon credentials when developing an Amazon application on a non-Amazon compute service. For example, this would be something like your local development environment. If you are developing on an Amazon resource, such as Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Cloud9, we recommend getting credentials from that service instead.

In this tutorial, you establish IAM Identity Center access and will configure it for your SDK or tool by using the Amazon access portal and the Amazon CLI.

Amazon Builder ID 18

• The Amazon access portal is the web location where you manually sign in to the IAM Identity Center. The format of the URL is d-xxxxxxxxxx.awsapps.com/startor your_subdomain.awsapps.com/start. When signed in to the Amazon access portal, you can view Amazon Web Services accounts and roles that have been configured for that user. This procedure uses the Amazon access portal to get configuration values you need for the SDK/tool authentication process.

• The Amazon CLI is used to configure your SDK or tool to use IAM Identity Center authentication for API calls made by your code. This one-time process updates your shared Amazon config file, that is then used by your SDK or tool when you run your code.

Prerequisites

Before starting this procedure, you should have completed the following:

- If you do not have an Amazon Web Services account, <u>sign up for an Amazon Web Services</u> account.
- If you haven't enabled IAM Identity Center yet, <u>enable IAM Identity Center</u> by following the instructions in the *Amazon IAM Identity Center User Guide*.

Configure programmatic access using IAM Identity Center

Step 1: Establish access and select appropriate permission set

Choose one of the following methods to access your Amazon credentials.

I do not have established access through IAM Identity Center

- 1. Add a user and add administrative permissions by following the <u>Configure user access with the</u> <u>default IAM Identity Center directory procedure in the *Amazon IAM Identity Center User Guide*.</u>
- The AdministratorAccess permission set should not be used for regular development.
 Instead, we recommend using the predefined PowerUserAccess permission set, unless your employer has created a custom permission set for this purpose.

Follow the same <u>Configure user access with the default IAM Identity Center directory</u> procedure again, but this time:

Prerequisites 19

• Instead of creating the *Admin team* group, create a *Dev team* group, and substitute this thereafter in the instructions.

- You can use the existing user, but the user must be added to the new Dev team group.
- Instead of creating the AdministratorAccess permission set, create a
 PowerUserAccess permission set, and substitute this thereafter in the instructions.

When you are done, you should have the following:

- A Dev team group.
- An attached PowerUserAccess permission set to the Dev team group.
- Your user added to the Dev team group.
- Exit the portal and sign in again to see your Amazon Web Services accounts and options for Administrator or PowerUserAccess. Select PowerUserAccess when working with your tool/SDK.

I already have access to Amazon through a federated identity provider managed by my employer (such as Microsoft Entra or Okta)

Sign in to Amazon through your identity provider's portal. If your Cloud Administrator has granted you PowerUserAccess (developer) permissions, you see the Amazon Web Services accounts that you have access to and your permission set. Next to the name of your permission set, you see options to access the accounts manually or programmatically using that permission set.

Custom implementations might result in different experiences, such as different permission set names. If you're not sure which permission set to use, contact your IT team for help.

I already have access to Amazon through the Amazon access portal managed by my employer

Sign in to Amazon through the Amazon access portal. If your Cloud Administrator has granted you PowerUserAccess (developer) permissions, you see the Amazon Web Services accounts that you have access to and your permission set. Next to the name of your permission set, you see options to access the accounts manually or programmatically using that permission set.

I already have access to Amazon through a federated custom identity provider managed by my employer

Contact your IT team for help.

Step 2: Configure SDKs and tools to use IAM Identity Center

- 1. On your development machine, install the latest Amazon CLI.
 - a. See <u>Installing or updating the latest version of the Amazon CLI</u> in the *Amazon Command Line Interface User Guide*.
 - b. (Optional) To verify that the Amazon CLI is working, open a command prompt and run the aws --version command.
- 2. Sign in to the Amazon access portal. Your employer may provide this URL or you may get it in an email following **Step 1: Establish access**. If not, find your **Amazon access portal URL** on the **Dashboard** of https://console.amazonaws.cn/singlesignon/.
 - a. In the Amazon access portal, in the **Accounts** tab, select the individual account to manage. The roles for your user are displayed. Choose **Access keys** to get credentials for command line or programmatic access for the appropriate permission set. Use the predefined PowerUserAccess permission set, or whichever permission set you or your employer has created to apply least-privilege permissions for development.
 - In the Get credentials dialog box, choose either MacOS and Linux or Windows, depending on your operating system.
 - c. Choose the IAM Identity Center credentials method to get the Issuer URL and SSO Region values that you need for the next step. Note: SSO Start URL can be used interchangeably with Issuer URL.
- 3. In the Amazon CLI command prompt, run the aws configure sso command. When prompted, enter the configuration values that you collected in the previous step. For details on this Amazon CLI command, see Configure your profile with the aws configure sso wizard.
 - a. For the prompt SSO Start URL, enter the value you obtained for Issuer URL.
 - b. For **CLI profile name**, we recommend entering *default* when you are getting started. For information about how to set non-default (named) profiles and their associated environment variable, see Profiles.
- 4. (Optional) In the Amazon CLI command prompt, confirm the active session identity by running the aws sts get-caller-identity command. The response should show the IAM Identity Center permission set that you configured.
- 5. If you are using an Amazon SDK, create an application for your SDK in your development environment.

a. For some SDKs, additional packages such as SSO and SSOOIDC must be added to your application before you can use IAM Identity Center authentication. For details, see your specific SDK.

b. If you previously configured access to Amazon, review your shared Amazon credentials file for any Amazon access keys. You must remove any static credentials before the SDK or tool will use the IAM Identity Center credentials because of the Understand the credential provider chain precedence.

For a deep dive into how the SDKs and tools use and refresh credentials using this configuration, see How IAM Identity Center authentication is resolved for Amazon SDKs and tools.

To configure IAM Identity Center provider settings directly in the shared config file, see <u>IAM</u> Identity Center credential provider in this guide.

Refreshing portal access sessions

Your access will eventually expire and the SDK or tool will encounter an authentication error. When this expiration occurs depends on your configured session lengths. To refresh the access portal session again when needed, use the Amazon CLI to run the aws sso login command.

You can extend both the IAM Identity Center access portal session duration and the permission set session duration. This lengthens the amount of time that you can run code before you need to manually sign in again with the Amazon CLI. For more information, see the following topics in the *Amazon IAM Identity Center User Guide*:

- IAM Identity Center session duration Configure the duration of your users' Amazon access portal sessions
- Permission set session duration Set session duration

How IAM Identity Center authentication is resolved for Amazon SDKs and tools

Relevant IAM Identity Center terms

The following terms help you understand the process and configuration behind Amazon IAM Identity Center. The documentation for Amazon SDK APIs uses different names than IAM Identity Center for some of these authentication concepts. It's helpful to know both names.

The following table shows how alternative names relate to each other.

IAM Identity Center name	SDK API name	Description
Identity Center	SSO	Although Amazon Single Sign-On is renamed, the sso API namespaces will keep their original name for backward compatibility purposes. For more informati on, see IAM Identity Center rename in the Amazon IAM Identity Center User Guide.
IAM Identity Center console		The console you use to configure single sign-on.
Administrative console		
Amazon access portal URL		A URL unique to your IAM Identity Center account, like https://xxx.awsapps.com/start . You sign in to this portal using your IAM Identity Center sign-in credentials.
IAM Identity Center Access Portal session	Authentication session	Provides a bearer access token to the caller.
Permission set session		The IAM session that the SDK uses internally to make the Amazon Web Services service calls. In informal discussions,

IAM Identity Center name	SDK API name	Description
		you might see this incorrectly referred to as "role session."
Permission set credentials	Amazon credentials sigv4 credentials	The credentials the SDK actually uses for most Amazon Web Services service calls (specifically, all sigv4 Amazon Web Services service calls). In informal discussions, you might see this incorrectly referred to as "role credentials."
IAM Identity Center credentia l provider	SSO credential provider	How you get the credentials, such as the class or module providing the functionality.

Understand SDK credential resolution for Amazon Web Services services

The IAM Identity Center API exchanges bearer token credentials for sigv4 credentials. Most Amazon Web Services are sigv4 APIs, with a few exceptions like Amazon CodeWhisperer and Amazon CodeCatalyst. The following describes the credential resolution process for supporting most Amazon Web Services service calls for your application code through Amazon IAM Identity Center.

Start an Amazon access portal session

- Start the process by signing in to the session with your credentials.
 - Use the aws sso login command in the Amazon Command Line Interface (Amazon CLI). This starts a new IAM Identity Center session if you don't already have an active session.
- When you start a new session, you receive a refresh token and access token from IAM Identity Center. The Amazon CLI also updates an SSO cache JSON file with a new access token and refresh token and makes it available for use by SDKs.
- If you already have an active session, the Amazon CLI command reuses the existing session and will expire whenever the existing session expires. To learn how to set the length of an IAM

Identity Center session, see <u>Configure the duration of your users' Amazon access portal sessions</u> in the *Amazon IAM Identity Center User Guide*.

• The maximum session length has been extended to 90 days to reduce the need for frequent sign-ins.

How the SDK gets credentials for Amazon Web Services service calls

SDKs provide access to Amazon Web Services services when you instantiate a client object per service. When the selected profile of the shared Amazon config file is configured for IAM Identity Center credential resolution, IAM Identity Center is used to resolve credentials for your application.

• The credential resolution process is completed during runtime when a client is created.

To retrieve credentials for sigv4 APIs using IAM Identity Center single sign-on, the SDK uses the IAM Identity Center access token to get an IAM session. This IAM session is called a permission set session, and it provides Amazon access to the SDK by assuming an IAM role.

- The permission set session duration is set independently from the IAM Identity Center session duration.
 - To learn how to set the permission set session duration, see <u>Set session duration</u> in the Amazon IAM Identity Center User Guide.
- Be aware that the permission set credentials are also referred to as *Amazon credentials* and *sigv4* credentials in most Amazon SDK API documentation.

The permission set credentials are returned from a call to <u>getRoleCredentials</u> of the IAM Identity Center API to the SDK. The SDK's client object uses that assumed IAM role to make calls to the Amazon Web Services service, such as asking Amazon S3 to list the buckets in your account. The client object can continue to operate using those permission set credentials until the permission set session expires.

Session expiration and refresh

When using the <u>SSO token provider configuration</u>, the hourly access token obtained from IAM Identity Center is automatically refreshed using the refresh token.

• If the access token is expired when the SDK tries to use it, the SDK uses the refresh token to try to get a new access token. The IAM Identity Center compares the refresh token to your IAM

Identity Center access portal session duration. If the refresh token is not expired, the IAM Identity Center responds with another access token.

• This access token can be used to either refresh the permission set session of existing clients, or to resolve credentials for new clients.

However, if the IAM Identity Center access portal session is expired, then no new access token is granted. Therefore, the permission set duration cannot be renewed. It will expire (and access will be lost) whenever the cached permission set session length times out for existing clients.

Any code that creates a new client will fail authentication as soon as the IAM Identity Center session expires. This is because the permission set credentials are not cached. Your code won't be able to create a new client and complete the credential resolution process until you have a valid access token.

To recap, when the SDK needs new permission set credentials, the SDK first checks for any valid, existing credentials and uses those. This applies whether the credentials are for a new client or for an existing client with expired credentials. If credentials aren't found or they're not valid, then the SDK calls the IAM Identity Center API to get new credentials. To call the API, it needs the access token. If the access token is expired, the SDK uses the refresh token to try to get a new access token from the IAM Identity Center service. This token is granted if your IAM Identity Center access portal session is not expired.

Using IAM Roles Anywhere to authenticate Amazon SDKs and tools

You can use IAM Roles Anywhere to get temporary security credentials in IAM for workloads such as servers, containers, and applications that run outside of Amazon. To use IAM Roles Anywhere, your workloads must use X.509 certificates. Your Cloud Administrator should provide the certificate and private key needed to configure IAM Roles Anywhere as your credential provider.

Step 1: Configure IAM Roles Anywhere

IAM Roles Anywhere provides a way to get temporary credentials for a workload or process that runs outside of Amazon. A trust anchor is established with the certificate authority to get temporary credentials for the associated IAM role. The role sets the permissions your workload will have when your code authenticates with IAM Roles Anywhere.

IAM Roles Anywhere 26

For steps to set up the trust anchor, IAM role, and IAM Roles Anywhere profile, see Creating a trust anchor and profile in Amazon Identity and Access Management Roles Anywhere in the IAM Roles Anywhere User Guide.



Note

A profile in the IAM Roles Anywhere User Guide refers to a unique concept within the IAM Roles Anywhere service. It's not related to the profiles within the shared Amazon config file.

Step 2: Use IAM Roles Anywhere

To get temporary security credentials from IAM Roles Anywhere, use the credential helper tool provided by IAM Roles Anywhere. The credential tool implements the signing process for IAM Roles Anywhere.

For instructions to download the credential helper tool, see Obtaining temporary security credentials from Amazon Identity and Access Management Roles Anywhere in the IAM Roles Anywhere User Guide.

To use temporary security credentials from IAM Roles Anywhere with Amazon SDKs and the Amazon CLI, you can configure credential_process setting in the shared Amazon config file. The SDKs and Amazon CLI support a process credential provider that uses credential_process to authenticate. The following shows the general structure to set credential_process.

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

The credential-process command of the helper tool returns temporary credentials in a standard JSON format that is compatible with the credential_process setting. Note that the command name contains a hyphen but the setting name contains an underscore. The command requires the following parameters:

- private-key The path to the private key that signed the request.
- certificate The path to the certificate.
- role-arn The ARN of the role to get temporary credentials for.

- profile-arn The ARN of the profile that provides a mapping for the specified role.
- trust-anchor-arn The ARN of the trust anchor used to authenticate.

Your Cloud Administrator should provide the certificate and private key. All three ARN values can be copied from the Amazon Web Services Management Console. The following example shows a shared config file that configures retrieving temporary credentials from the helper tool.

```
[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
```

For optional parameters and additional helper tool details, see <u>IAM Roles Anywhere Credential</u> Helper on GitHub.

For details on the SDK configuration setting itself and the process credential provider, see <u>Process</u> credential provider in this guide.

Assuming a role with Amazon credentials to authenticate Amazon SDKs and tools

Assuming a role involves using a set of temporary security credentials to access Amazon resources that you might not have access to otherwise. These temporary credentials consist of an access key ID, a secret access key, and a security token. To learn more about Amazon Security Token Service (Amazon STS) API requests, see Actions in the Amazon Security Token Service API Reference.

To set up your SDK or tool to assume a role, you must first create or identify a specific *role* to assume. IAM roles are uniquely identified by a role Amazon Resource Name (ARN). Roles establish trust relationships with another entity. The trusted entity that uses the role might be an Amazon Web Services service or another Amazon Web Services account. To learn more about IAM roles, see Using IAM roles in the IAM User Guide.

After the IAM role is identified, if you are trusted by that role, you can configure your SDK or tool to use the permissions that are granted by the role.

Assume a role 28



Note

It is an Amazon best practice to use Regional endpoints whenever possible and to configure your Amazon Web Services Region.

Assume an IAM role

When assuming a role, Amazon STS returns a set of temporary security credentials. These credentials are sourced from another profile or from the instance or container that your code is running in. Most commonly this type of assuming a role is used when you have Amazon credentials for one account, but your application needs access to resources in another account.

Step 1: Set up an IAM role

To set up your SDK or tool to assume a role, you must first create or identify a specific role to assume. IAM roles are uniquely identified using a role ARN. Roles establish trust relationships with another entity, typically within your account or for cross-account access. To set this up, see Creating IAM roles in the IAM User Guide.

Step 2: Configure the SDK or tool

Configure the SDK or tool to source credentials from credential_source or source_profile.

Use credential_source to source credentials from an Amazon ECS container, an Amazon EC2 instance, or from environment variables.

Use source_profile to source credentials from another profile. source_profile also supports role chaining, which is hierarchies of profiles where an assumed role is then used to assume another role.

When you specify this in a profile, the SDK or tool automatically makes the corresponding Amazon STS AssumeRole API call for you. To retrieve and use temporary credentials by assuming a role, specify the following configuration values in the shared Amazon config file. For more details on each of these settings, see the Assume role credential provider settings section.

- role_arn From the IAM role you created in Step 1
- Configure either source_profile or credential_source

Assume an IAM role

- (Optional) duration_seconds
- (Optional) external_id
- (Optional) mfa_serial
- (Optional) role_session_name

credential_source = Ec2InstanceMetadata

The following examples show the configuration of both assume role options in a shared config file:

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

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

For details on all assume role credential provider settings, see <u>Assume role credential provider</u> in this guide.

Assuming a role with web identity or OpenID Connect to authenticate Amazon SDKs and tools

Assuming a role involves using a set of temporary security credentials to access Amazon resources that you might not have access to otherwise. These temporary credentials consist of an access key ID, a secret access key, and a security token. To learn more about Amazon Security Token Service (Amazon STS) API requests, see Actions in the Amazon Security Token Service API Reference.

To set up your SDK or tool to assume a role, you must first create or identify a specific *role* to assume. IAM roles are uniquely identified by a role Amazon Resource Name (<u>ARN</u>). Roles establish trust relationships with another entity. The trusted entity that uses the role might be a web identity provider or OpenID Connect(OIDC), or SAML federation. To learn more about IAM roles, see Methods to assume a role in the *IAM User Guide*.

After the IAM role is configured in your SDK, if that role is configured to trust your identity provider, you can further configure your SDK to assume that role in order to get temporary Amazon credentials.

Assume a role (web) 30



Note

It is an Amazon best practice to use Regional endpoints whenever possible and to configure your Amazon Web Services Region.

Federate with web identity or OpenID Connect

You can use the JSON Web Tokens (JWTs) from public identity providers, such as Login With Amazon, Facebook, Google to get temporary Amazon credentials using AssumeRoleWithWebIdentity. Depending on how they are used, these JWTs may be called ID tokens or access tokens. You may also use JWTs issued from identity providers (IdPs) that are compatible with OIDC's discovery protocol, such as Entrald or PingFederate.

If you are using Amazon Elastic Kubernetes Service, this feature provides the ability to specify different IAM roles for each one of your service accounts in an Amazon EKS cluster. This Kubernetes feature distributes JWTs to your pods which are then used by this credential provider to obtain temporary Amazon credentials. For more information on this Amazon EKS configuration, see IAM roles for service accounts in the Amazon EKS User Guide. However, for a simpler option, we recommend you use Amazon EKS Pod Identities instead if your SDK supports it.

Step 1: Set up an identity provider and IAM role

To configure federation with an external IdP, use an IAM identity provider to inform Amazon about the external IdP and its configuration. This establishes trust between your Amazon Web Services account and the external IdP. Before configuring the SDK to use the JSON Web Token (JWT) for authentication, you must first set up the identity provider (IdP) and the IAM role used to access it. To set these up, see Creating a role for web identity or OpenID Connect Federation (console) in the IAM User Guide.

Step 2: Configure the SDK or tool

Configure the SDK or tool to use a JSON Web Token (JWT) from Amazon STS for authentication.

When you specify this in a profile, the SDK or tool automatically makes the corresponding Amazon STS AssumeRoleWithWebIdentity API call for you. To retrieve and use temporary credentials using web identity federation, specify the following configuration values in the shared Amazon config file. For more details on each of these settings, see the Assume role credential provider settings section.

- role_arn From the IAM role you created in Step 1
- web_identity_token_file From the external IdP
- (Optional) duration_seconds
- (Optional) role_session_name

The following is an example of a shared config file configuration to assume a role with web identity:

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

Note

For mobile applications, consider using Amazon Cognito. Amazon Cognito acts as an identity broker and does much of the federation work for you. However, the Amazon Cognito identity provider isn't included in the SDKs and tools core libraries like other identity providers. To access the Amazon Cognito API, include the Amazon Cognito service client in the build or libraries for your SDK or tool. For usage with Amazon SDKs, see Code Examples in the Amazon Cognito Developer Guide.

For details on all assume role credential provider settings, see <u>Assume role credential provider</u> in this guide.

Using Amazon access keys to authenticate Amazon SDKs and tools

Using Amazon access keys is an option for authentication when using Amazon SDKs and tools.

Use short-term credentials

We recommend configuring your SDK or tool to use <u>Using IAM Identity Center to authenticate</u> Amazon SDK and tools to use extended session duration options.

However, to set up the SDK or tool's temporary credentials directly, see <u>Using short-term</u> credentials to authenticate Amazon SDKs and tools.

Use long-term credentials

Marning

To avoid security risks, don't use IAM users for authentication when developing purposebuilt software or working with real data. Instead, use federation with an identity provider such as Amazon IAM Identity Center.

Manage access across Amazon Web Services accounts

As a security best practice, we recommend using Amazon Organizations with IAM Identity Center to manage access across all your Amazon Web Services accounts. For more information, see Security best practices in IAM in the IAM User Guide.

You can create users in IAM Identity Center, use Microsoft Active Directory, use a SAML 2.0 identity provider (IdP), or individually federate your IdP to Amazon Web Services accounts. Using one of these approaches, you can provide a single sign-on experience for your users. You can also enforce multi-factor authentication (MFA) and use temporary credentials for Amazon Web Services account access. This differs from an IAM user, which is a long-term credential that can be shared and which might increase the security risk to your Amazon resources.

Create IAM users for sandbox environments only

If you're new to Amazon, you might create a test IAM user and then use it to run tutorials and explore what Amazon has to offer. It's okay to use this type of credential when you're learning, but we recommend that you avoid using it outside of a sandbox environment.

For the following use cases, it might make sense to get started with IAM users in Amazon:

- Getting started with your Amazon SDK or tool and exploring Amazon Web Services services in a sandbox environment.
- Running scheduled scripts, jobs, and other automated processes that don't support a humanattended sign-in process as part of your learning.

If you're using IAM users outside of these use cases, then transition to IAM Identity Center or federate your identity provider to Amazon Web Services accounts as soon as possible. For more information, see Identity federation in Amazon.

Use long-term credentials 33

Secure IAM user access keys

You should rotate IAM user access keys regularly. Follow the guidance in <u>Rotating access keys</u> in the *IAM User Guide*. If you believe that you have accidentally shared your IAM user access keys, then rotate your access keys.

IAM user access keys should be stored in the shared Amazon credentials file on the local machine. Don't store the IAM user access keys in your code. Don't include configuration files that contain your IAM user access keys inside of any source code management software. External tools, such as the open source project <u>git-secrets</u>, can help you from inadvertently committing sensitive information to a Git repository. For more information, see <u>IAM Identities (users, user groups, and roles)</u> in the *IAM User Guide*.

To set up an IAM user to get started, see <u>Using long-term credentials to authenticate Amazon</u> SDKs and tools.

Using short-term credentials to authenticate Amazon SDKs and tools

We recommend configuring your Amazon SDK or tool to use <u>Using IAM Identity Center to</u> <u>authenticate Amazon SDK and tools</u> with extended session duration options. However, you can copy and use temporary credentials that are available in the Amazon access portal. New credentials will need to be copied when these expire. You can use the temporary credentials in a profile or use them as values for system properties and environment variables.

Best practice: Instead of manually managing access keys and a token in the credentials file, we recommend your application uses temporary credentials delivered from:

- An Amazon compute service, such as running your application on Amazon Elastic Compute Cloud or in Amazon Lambda.
- Another option in the credential provider chain, such as <u>Using IAM Identity Center to</u> authenticate Amazon SDK and tools.
- Or use the Process credential provider to retrieve temporary credentials.

Set up a credentials file using short-term credentials retrieved from Amazon access portal

- 1. Create a shared credentials file.
- 2. In the credentials file, paste the following placeholder text until you paste in working temporary credentials.

Short-term credentials 34

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

3. Save the file. The file ~/.aws/credentials should now exist on your local development system. This file contains the [default] profile that the SDK or tool uses if a specific named profile is not specified.

- 4. Sign in to the Amazon access portal.
- 5. Follow these instructions for <u>Manual credential refresh</u> to copy IAM role credentials from the Amazon access portal.
 - a. For step 4 in the linked instructions, choose the IAM role name that grants access for your development needs. This role typically has a name like **PowerUserAccess** or **Developer**.
 - b. For step 7 in the linked instructions, select the **Manually add a profile to your Amazon credentials file** option and copy the contents.
- 6. Paste the copied credentials into your local credentials file. The generated profile name is not needed if you are using the default profile. Your file should resemble the following.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONG
```

7. Save the credentials file.

When the SDK creates a service client, it will access these temporary credentials and use them for each request. The settings for the IAM role chosen in step 5a determine how long the temporary credentials are valid. The maximum duration is twelve hours.

After the temporary credentials expire, repeat steps 4 through 7.

Short-term credentials 35

Using long-term credentials to authenticate Amazon SDKs and tools

Marning

To avoid security risks, don't use IAM users for authentication when developing purposebuilt software or working with real data. Instead, use federation with an identity provider such as Amazon IAM Identity Center.

If you use an IAM user to run your code, then the SDK or tool in your development environment authenticates by using long-term IAM user credentials in the shared Amazon credentials file. Review the Security best practices in IAM topic and transition to IAM Identity Center or other temporary credentials as soon as possible.

Important warnings and guidance for credentials

Warnings for credentials

- Do NOT use your account's root credentials to access Amazon resources. These credentials provide unrestricted account access and are difficult to revoke.
- **Do NOT** put literal access keys or credential information in your application files. If you do, you create a risk of accidentally exposing your credentials if, for example, you upload the project to a public repository.
- **Do NOT** include files that contain credentials in your project area.
- Be aware that any credentials stored in the shared Amazon credentials file are stored in plaintext.

Additional guidance for securely managing credentials

For a general discussion of how to securely manage Amazon credentials, see Best practices for managing Amazon access keys in the Amazon Web Services General Reference. In addition to that discussion, consider the following:

- Use IAM roles for tasks for Amazon Elastic Container Service (Amazon ECS) tasks.
- Use IAM roles for applications that are running on Amazon EC2 instances.

Long-term credentials

Prerequisites: Create an Amazon account

To use an IAM user to access Amazon services, you need an Amazon account and Amazon credentials.

Create an account.

To create an Amazon account, see <u>Getting started: Are you a first-time Amazon user?</u> in the *Amazon Account Management Reference Guide*.

2. Create an administrative user.

Avoid using your root user account (the initial account you create) to access the management console and services. Instead, create an administrative user account, as explained in <u>Create an</u> administrative user in the *IAM User Guide*.

After you create the administrative user account and record the login details, **be sure to sign out of your root user account** and sign back in using the administrative account.

Neither of these accounts are appropriate for doing development on Amazon or for running applications on Amazon. As a best practice, you need to create users, permission sets, or service roles that are appropriate for these tasks. For more information, see Apply least-privilege permissions in the IAM User Guide.

Step 1: Create your IAM user

- Create your IAM user by following the <u>Creating IAM users (console)</u> procedure in the *IAM User Guide*. When creating your IAM user:
 - We recommend you select Provide user access to the Amazon Web Services Management
 Console. This allows you to view Amazon Web Services services related to the code that you
 are running in a visual environment, such as checking Amazon CloudTrail diagnostic logs or
 uploading files to Amazon Simple Storage Service, which is helpful when debugging your
 code.
 - For **Set permissions Permission options**, select **Attach policies directly** for how you want to assign permissions to this user.
 - Most "Getting Started" SDK tutorials use the Amazon S3 service as an example. To provide your application with full access to Amazon S3, select the AmazonS3FullAccess policy to attach to this user.

Long-term credentials 37

• You can ignore the optional steps of that procedure regarding setting permission boundaries or tags.

Step 2: Get your access keys

- 1. In the navigation pane of the IAM console, select **Users** and then select the **User name** of the user that you created previously.
- On the user's page, select the Security credentials page. Then, under Access keys, select Create access key.
- 3. For **Create access key Step 1**, choose either **Command Line Interface (CLI)** or **Local code**. Both options generate the same type of key to use with both the Amazon CLI and the SDKs.
- 4. For Create access key Step 2, enter an optional tag and select Next.
- 5. For **Create access key Step 3**, select **Download .csv file** to save a .csv file with your IAM user's access key and secret access key. You need this information for later.

Marning

Use appropriate security measures to keep these credentials safe.

6. Select **Done**.

Step 3: Update the shared credentials file

- 1. Create or open the shared Amazon credentials file. This file is ~/.aws/credentials on Linux and macOS systems, and %USERPROFILE%\.aws\credentials on Windows. For more information, see Location of Credentials Files.
- 2. Add the following text to the shared credentials file. Replace the example ID value and example key value with the values in the .csv file that you downloaded earlier.

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

Save the file.

Long-term credentials 38

The shared credentials file is the most common way to store credentials. These can also be set as environment variables, see Amazon access keys for environment variable names. This is a way to get you started, but we recommend you transition to IAM Identity Center or other temporary credentials as soon as possible. After you transition away from using long-term credentials, remember to delete these credentials from the shared credentials file.

Using IAM roles to authenticate applications deployed to Amazon EC2

This example covers setting up an Amazon Identity and Access Management role with Amazon S3 access to use in your application deployed to an Amazon Elastic Compute Cloud instance.

In order to run your Amazon SDK application on an Amazon Elastic Compute Cloud instance, create an IAM role, and then give your Amazon EC2 instance access to that role. For more information, see IAM Roles for Amazon EC2 in the Amazon EC2 User Guide.

Create an IAM role

The Amazon SDK application that you develop likely accesses at least one Amazon Web Services service to perform actions. Create an IAM role that grants the required permissions necessary for your application to run.

This procedure creates a role that grants read-only access to Amazon S3 as an example. Many of the Amazon SDK guides have "getting started" tutorials that read from Amazon S3.

- 1. Sign in to the Amazon Web Services Management Console and open the IAM console at https://console.amazonaws.cn/iam/.
- 2. In the navigation pane, select **Roles**, then select **Create role**.
- 3. For Select trusted entity, under Trusted entity type, choose Amazon Web Services service.
- 4. Under **Use case**, choose **Amazon EC2**, then select **Next**.
- 5. For **Add permissions**, select the checkbox for **Amazon S3 Read Only Access** from the policy list, then select **Next**.
- 6. Enter a name for the role, then select **Create role**. Remember this name because you'll need it when you create your Amazon EC2 instance.

IAM roles for EC2 instances 39

Launch an Amazon EC2 instance and specify your IAM role

You can create and launch an Amazon EC2 instance using your IAM role by doing the following:

- Follow Quickly launch an instance in the Amazon EC2 User Guide. However, prior to the final submission step, also do the following:
 - Under Advanced details, for IAM Instance profile, choose the role that you created in the previous step.

With this IAM and Amazon EC2 setup, you can deploy your application to the Amazon EC2 instance and your application will have read access to the Amazon S3 service.

Connect to the EC2 instance

Connect to the Amazon EC2 instance so that you can transfer your application to it and then run the application. You'll need the file that contains the private portion of the key pair you used under **Key pair (login)** when you created your instance; that is, the PEM file.

You can do this by following the guidance for your instance type: Connect to your Linux instance or Connect to your Windows instance. When you connect, do so in such a way that you can transfer files from your development machine to your instance.



Note

On Linux or macOS terminal, you can use the secure copy command to copy your application. To use scp with a key pair, you can use the following command: scp -i path/to/key file/to/copy ec2-user@ec2-xx-xx-xxxxxx.compute.amazonaws.com:~.

For more information for Windows, see Transfer files to Windows instances.

If you're using an Amazon Toolkit, you can often also connect to the instance by using the Toolkit. For more information, see the specific user guide for the Toolkit you use.

Run your application on the EC2 instance

Copy your application files from your local drive to your Amazon EC2 instance.

2. Start the application and verify that it runs with the same results as on your development machine.

- 3. (Optional) Verify that the application uses the credentials provided by the IAM role.
 - a. Sign in to the Amazon Web Services Management Console and open the Amazon EC2 console at https://console.amazonaws.cn/ec2/.
 - b. Select the instance.
 - c. Choose Actions, Security, and then choose Modify IAM role.
 - d. For IAM role, detach the IAM role by choosing No IAM Role.
 - e. Choose **Update IAM role**.
 - f. Run the application again and confirm that it returns an authorization error.

Using the TIP plugin to access Amazon Web Services services

Trusted identity propagation (TIP) is a feature of Amazon IAM Identity Center that enables administrators of Amazon Web Services services to grant permissions based on user attributes such as group associations. With trusted identity propagation, identity context is added to an IAM role to identify the user requesting access to Amazon resources. This context is propagated to other Amazon Web Services services.

Identity context comprises information that Amazon Web Services services use to make authorization decisions when they receive access requests. This information includes metadata that identifies the requester (for example, an IAM Identity Center user), the Amazon Web Services service to which access is requested (for example, Amazon Redshift), and the scope of access (for example, read only access). The receiving Amazon Web Services service uses this context, and any permissions assigned to the user, to authorize access to its resources. For more information, see in the Trusted identity propagation overview in the Amazon IAM Identity Center User Guide.

Prerequisites for using the TIP plugin

The following resources are required in order for the plugin to work:

- 1. You must be using either the Amazon SDK for Java or the Amazon SDK for JavaScript.
- 2. Verify that the service you are using supports the trusted identity propagation.

Trusted identity propagation 41

See the **Enables trusted identity propagation through IAM Identity Center** column of the <u>Amazon managed applications that integrate with IAM Identity Center</u> table in the *Amazon IAM Identity Center User Guide*.

3. Enable IAM Identity Center and trusted identity propagation.

See TIP prerequisites and considerations in the Amazon IAM Identity Center User Guide.

4. You must have an Identity-Center-integrated application.

See <u>Amazon managed applications</u> or <u>Customer managed applications</u> in the *Amazon IAM Identity Center User Guide*.

5. You must set up a trusted token issuer (TTI) and connect your service to IAM Identity Center.

See <u>Prerequisites for trusted token issuers</u> and <u>Tasks for setting up a trusted token issuer</u> in the *Amazon IAM Identity Center User Guide*.

To use the TIP plugin in your code

- 1. Create an instance of the trusted identity propagation plugin.
- 2. Create a service client instance for interacting with your Amazon Web Services service and customize the service client by adding the trusted identity propagation plugin.

The TIP plugin takes the following input parameters:

- webTokenProvider: A function that the customer implements to obtain an OpenID token from their external identity provider.
- accessRoleArn: The IAM role ARN to be assumed by the plugin with the user's identity context to get the identity-enhanced credentials.
- applicationArn: The unique identifier string for the client or application. This value is an application ARN that has OAuth grants configured.
- applicationRoleArn: (Optional) The IAM role ARN to be assumed with
 AssumeRoleWithWebIdentity so that the OIDC and Amazon STS clients can be bootstrapped
 without a default credentials provider. If this is not provided, the value of the accessRoleArn
 parameter will be used.

sso0idcClient: (Optional) An SSO OIDC client, such as <u>Sso0idcClient</u> for Java or <u>client-sso-oidc</u> for Javascript, with customer-defined configurations. If not provided, an OIDC client using default configurations is instantiated and used.

• **stsClient**: (Optional) An Amazon STS client with customer-defined configurations, used to assume accessRoleArn with the user's identity context. If not provided, an Amazon STS client using default configurations is instantiated and used.

Java

To use the TIP plugin in your Amazon SDK for Java project, you need to declare it as a dependency in your project's pom.xml file.

In your source code, include the required package statement for software.amazon.awssdk.trustedidentitypropagation.

The following example code shows how to create an instance of the trusted identity propagation plugin and then add the plugin to a service client instance.

This example uses an S3Client as the chosen Amazon Web Services service client to show obtaining IAM Identity Center tokens. However, any other Amazon Web Services service that supports TIP would be similar.

```
StsClient client = StsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(AnonymousCredentialsProvider.create()).build();

TrustedIdentityPropagationPlugin trustedIdentityPropagationPlugin =
    TrustedIdentityPropagationPlugin.builder()
    .stsClient(client)
    .webTokenProvider(() -> idToken)
    .applicationArn(idcApplicationArn)
    .accessRoleArn(accessRoleArn)
    .ssoOidcClient(SsoOidcClient.builder().region(Region.US_EAST_1).build())
    .build();
```

For additional details and source, see <u>trusted-identity-propagation-java</u> on GitHub. Javascript

Run the following command to install the TIP authentication plugin package in your Amazon SDK for JavaScript project:

```
$ npm i @aws-sdk-extension/trusted-identity-propagation
```

The final package. json should include a dependency similar to the following:

```
"dependencies": {
"@aws-sdk-extension/trusted-identity-propagation": "^1.0.0"
},
```

In your source code, import the required TrustedIdentityPropagationExtension dependency.

The following example code shows how to create an instance of the trusted identity propagation plugin and then add the plugin to a service client instance.

This example uses an S3Client as the chosen Amazon Web Services service client to show obtaining IAM Identity Center tokens. However, any other Amazon Web Services service that supports TIP would be similar.

```
import { S3Client } from "@aws-sdk/client-s3";
import { TrustedIdentityPropagationExtension } from "@aws-sdk-extension/trusted-
identity-propagation";

// Plugin configurations, please refer to the documentation on each of these fields.
const applicationRoleArn = 'YOUR_APPLICATION_ROLE_ARN';
const accessRoleArn = 'YOUR_ACCESS_ROLE_ARN';
const applicationArn = 'YOUR_APPLICATION_ARN';

const s3Client = new S3Client({
    region,
```

```
extensions: [
  TrustedIdentityPropagationExtension.create({
   webTokenProvider: async () => {
     return 'ID_TOKEN_FROM_YOUR_IDENTITY_PROVIDER';
   },
   applicationRoleArn,
   accessRoleArn,
   applicationArn,
  }),
  ],
});
```

For additional details and source, see trusted-identity-propagation-js on GitHub.

Amazon SDKs and tools settings reference

SDKs provide language-specific APIs for Amazon Web Services services. They take care of some of the heavy lifting necessary in successfully making API calls, including authentication, retry behavior, and more. To do this, the SDKs have flexible strategies to obtain credentials to use for your requests, to maintain settings to use with each service, and to obtain values to use for global settings.

You can find detailed information about configuration settings in the following sections:

- <u>Amazon SDKs and Tools standardized credential providers</u> Common credential providers standardized across multiple SDKs.
- <u>Amazon SDKs and Tools standardized features</u> Common features standardized across multiple SDKs.

Creating service clients

To programmatically access Amazon Web Services services, SDKs use a client class/object for each Amazon Web Services service. For example, if your application needs to access Amazon EC2, your application creates an Amazon EC2 client object to interface with that service. You then use the service client to make requests to that Amazon Web Services service. In most SDKs, a service client object is immutable, so you must create a new client for each service to which you make requests and for making requests to the same service using a different configuration.

Precedence of settings

Global settings configure features, credential providers, and other functionality that are supported by most SDKs and have a broad impact across Amazon Web Services services. All SDKs have a series of places (or sources) that they check in order to find a value for global settings. The following is the setting lookup precedence:

- Any explicit setting set in the code or on a service client itself takes precedence over anything else.
 - Some settings can be set on a per-operation basis, and can be changed as needed for each operation that you invoke. For the Amazon CLI or Amazon Tools for PowerShell, these take the form of per-operation parameters that you enter on the command line. For an SDK, explicit

Creating service clients 46

assignments can take the form of a parameter that you set when you instantiate an Amazon Web Services service client or configuration object, or sometimes when you call an individual API.

- 2. Java/Kotlin only: The JVM system property for the setting is checked. If it's set, that value is used to configure the client.
- 3. The environment variable is checked. If it's set, that value is used to configure the client.
- 4. The SDK checks the shared credentials file for the setting. If it's set, the client uses it.
- 5. The shared config file for the setting. If the setting is present, the SDK uses it.
 - The AWS_PROFILE environment variable or the aws.profile JVM system property can be used to specify which profile that the SDK loads.
- 6. Any default value provided by the SDK source code itself is used last.



Note

Some SDKs and tools might check in a different order. Also, some SDKs and tools support other methods of storing and retrieving parameters. For example, the Amazon SDK for .NET supports an additional source called the SDK Store. For more information about providers that are unique to a SDK or tool, see the specific guide for the SDK or tool that you are using.

The order determines which methods take precedence and override others. For example, if you set up a profile in the shared config file, it's only found and used after the SDK or tool checks the other places first. This means that if you put a setting in the credentials file, it is used instead of one found in the config file. If you configure an environment variable with a setting and value, it would override that setting in both the credentials and config files. And finally, a setting on the individual operation (Amazon CLI command-line parameter or API parameter) or in code would override all other values for that one command.

Understanding the settings pages of this guide

The pages within the Settings reference section of this guide detail the available settings that can be set through various mechanisms. The tables that follow list the config and credential file settings, environment variables, and (for Java and Kotlin SDKs) the JVM settings that can be

used outside of your code to configure the feature. Each linked topic in each list takes you to the corresponding settings page.

- Config file settings list
- Credentials file settings list
- Environment variables list
- JVM system properties list

Each credential provider or feature has a page where the settings that are used to configure that functionality are listed. For each setting, you can often set the value either by adding the setting to a configuration file, or by setting an environment variable, or (for Java and Kotlin only) by setting a JVM system property. Each setting lists all supported methods of setting the value in a block above the details of the description. Although the precedence varies, the resulting functionality is the same regardless of how you set it.

The description will include the default value, if any, that takes effect if you do nothing. It also defines what a valid value is for that setting.

For example, let's look at a setting from the Request compression feature page.

The disable_request_compression example setting's information documents the following:

- There are three equivalent ways to control request compression outside of your codebase. You can either:
 - Set it in your config file using disable_request_compression
 - Set it as an environment variable using AWS_DISABLE_REQUEST_COMPRESSION
 - Or, if you are using the Java or Kotlin SDK, set it as a JVM system property using aws.disableRequestCompression

Note

There might also be a way to configure the same functionality directly in your code, but this Reference does not cover this since it is unique to each SDK. If you want to set your configuration in the code itself, see your specific SDK guide or API reference.

- If you do nothing, the value will default to false.
- The only valid values for this Boolean setting are true and false.

At the bottom of each feature page there is a **Support by Amazon SDKs and tools** table.

This table shows whether your SDK supports the settings that are listed on the page. The Supported column indicates the support level with the following values:

- Yes The settings are fully supported by the SDK as written.
- Partial Some of the settings are supported or the behavior deviates from the description. For Partial, an additional note indicates the deviation.
- No None of the settings are supported. This doesn't make claims as to whether the same functionality might be achieved in code; it only indicates that the listed external configuration settings are not supported.

Config file settings list

The settings listed in the following table can be assigned in the shared Amazon config file. They are global and affect all Amazon Web Services services. SDKs and tools may also support unique settings and environment variables. To see the settings and environment variables supported by only an individual SDK or tool, see that specific SDK or tool guide.

Setting name	Details	
account_i d_endpoin t_mode	Account-based endpoints	
api_versions	General configuration settings	
aws_acces s_key_id	Amazon access keys	
aws_account_id	Account-based endpoints	
aws_secre t_access_key	Amazon access keys	
aws_sessi on_token	Amazon access keys	

Setting name	Details
ca_bundle	General configuration settings
credentia l_process	Process credential provider
credentia l_source	Assume role credential provider
defaults_mode	Smart configuration defaults
<pre>disable_h ost_prefi x_injection</pre>	Host prefix injection
<pre>disable_r equest_co mpression</pre>	Request compression
duration_ seconds	Assume role credential provider
ec2_metad ata_servi ce_endpoint	IMDS credential provider
ec2_metad ata_servi ce_endpoi nt_mode	IMDS credential provider
ec2_metad ata_v1_di sabled	IMDS credential provider
<pre>endpoint_ discovery _enabled</pre>	Endpoint discovery

Setting name	Details
endpoint_url	Service-specific endpoints
external_id	Assume role credential provider
ignore_co nfigured_ endpoint_urls	Service-specific endpoints
max_attempts	Retry behavior
<pre>metadata_ service_n um_attempts</pre>	Amazon EC2 instance metadata
<pre>metadata_ service_t imeout</pre>	Amazon EC2 instance metadata
mfa_serial	Assume role credential provider
output	General configuration settings
<pre>parameter _validation</pre>	General configuration settings
region	Amazon Web Services Region
request_c hecksum_c alculation	Data Integrity Protections for Amazon S3
<pre>request_m in_compre ssion_siz e_bytes</pre>	Request compression

Setting name	Details
response_ checksum_ validation	Data Integrity Protections for Amazon S3
retry_mode	Retry behavior
role_arn	Assume role credential provider
role_sess ion_name	Assume role credential provider
<pre>s3_disabl e_multire gion_acce ss_points</pre>	Amazon S3 Multi-Region Access Points
s3_use_ar n_region	Amazon S3 access points
sdk_ua_app_id	Application ID
source_profile	Assume role credential provider
sso_account_id	IAM Identity Center credential provider
sso_region	IAM Identity Center credential provider
sso_regis tration_scopes	IAM Identity Center credential provider
sso_role_name	IAM Identity Center credential provider
sso_start_url	IAM Identity Center credential provider
sts_regio nal_endpoints	Amazon STS Regional endpoints

Setting name	Details
use_duals tack_endpoint	<u>Dual-stack and FIPS endpoints</u>
use_fips_ endpoint	Dual-stack and FIPS endpoints
<pre>web_ident ity_token_file</pre>	Assume role credential provider

Credentials file settings list

The settings listed in the following table can be assigned in the shared Amazon credentials file. They are global and affect all Amazon Web Services services. SDKs and tools may also support unique settings and environment variables. To see the settings and environment variables supported by only an individual SDK or tool, see that specific SDK or tool guide.

Setting name	Details	
aws_acces s_key_id	Amazon access keys	
aws_secre t_access_key	Amazon access keys	
aws_sessi on_token	Amazon access keys	

Environment variables list

Environment variables supported by most SDKs are listed in the following table. They are global and affect all Amazon Web Services services. SDKs and tools may also support unique settings and environment variables. To see the settings and environment variables supported by only an individual SDK or tool, see that specific SDK or tool guide.

Setting name	Details
AWS_ACCES S_KEY_ID	Amazon access keys
AWS_ACCOUNT_ID	Account-based endpoints
AWS_ACCOU NT_ID_END POINT_MODE	Account-based endpoints
AWS_CA_BUNDLE	General configuration settings
AWS_CONFIG_FILE	Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools
AWS_CONTA INER_AUTH ORIZATION _TOKEN	Container credential provider
AWS_CONTA INER_AUTH ORIZATION _TOKEN_FILE	Container credential provider
AWS_CONTA INER_CRED ENTIALS_F ULL_URI	Container credential provider
AWS_CONTA INER_CRED ENTIALS_R ELATIVE_URI	Container credential provider

Setting name	Details
AWS_DEFAU LTS_MODE	Smart configuration defaults
AWS_DISAB LE_HOST_P REFIX_INJ ECTION	Host prefix injection
AWS_DISAB LE_REQUES T_COMPRESSION	Request compression
AWS_EC2_M ETADATA_D ISABLED	IMDS credential provider
AWS_EC2_M ETADATA_S ERVICE_EN DPOINT	IMDS credential provider
AWS_EC2_M ETADATA_S ERVICE_EN DPOINT_MODE	IMDS credential provider
AWS_EC2_M ETADATA_V 1_DISABLED	IMDS credential provider
AWS_ENABL E_ENDPOIN T_DISCOVERY	Endpoint discovery
AWS_ENDPO INT_URL	Service-specific endpoints

Setting name	Details
AWS_ENDPO INT_URL_< SERVICE>	Service-specific endpoints
AWS_IGNOR E_CONFIGU RED_ENDPO INT_URLS	Service-specific endpoints
AWS_MAX_A TTEMPTS	Retry behavior
AWS_METAD ATA_SERVI CE_NUM_AT TEMPTS	Amazon EC2 instance metadata
AWS_METAD ATA_SERVI CE_TIMEOUT	Amazon EC2 instance metadata
AWS_PROFILE	Using shared config and credentials files to globally configure Amazon SDKs and tools
AWS_REGION	Amazon Web Services Region
AWS_REQUE ST_CHECKS UM_CALCULATION	Data Integrity Protections for Amazon S3
AWS_REQUE ST_MIN_CO MPRESSION _SIZE_BYTES	Request compression

Setting name	Details
AWS_RESPO NSE_CHECK SUM_VALIDATION	Data Integrity Protections for Amazon S3
AWS_RETRY_MODE	Retry behavior
AWS_ROLE_ARN	Assume role credential provider
AWS_ROLE_ SESSION_NAME	Assume role credential provider
AWS_S3_DI SABLE_MUL TIREGION_ ACCESS_POINTS	Amazon S3 Multi-Region Access Points
AWS_S3_US E_ARN_REGION	Amazon S3 access points
AWS_SDK_U A_APP_ID	Application ID
AWS_SECRE T_ACCESS_KEY	Amazon access keys
AWS_SESSI ON_TOKEN	Amazon access keys
AWS_SHARE D_CREDENT IALS_FILE	Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools
AWS_STS_R EGIONAL_E NDPOINTS	Amazon STS Regional endpoints

Setting name	Details
AWS_USE_D UALSTACK_ ENDPOINT	<u>Dual-stack and FIPS endpoints</u>
AWS_USE_F IPS_ENDPOINT	<u>Dual-stack and FIPS endpoints</u>
AWS_WEB_I DENTITY_T OKEN_FILE	Assume role credential provider

JVM system properties list

You can use the following JVM system properties for the Amazon SDK for Java and the Amazon SDK for Kotlin (targeting the JVM). See the section called "How to set JVM system properties" for instructions on how to set JVM system properties.

Setting name	Details
aws.accessKeyId	Amazon access keys
aws.accountId	Account-based endpoints
aws.accou ntIdEndpo intMode	Account-based endpoints
aws.configFile	Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools
aws.defau ltsMode	Smart configuration defaults

JVM system properties list 58

Setting name	Details			
aws.disab leEc2Meta dataV1	IMDS credential provider			
aws.disab leHostPre fixInjection	Host prefix injection			
aws.disab leRequest Compression	Request compression			
aws.ec2Me tadataSer viceEndpoint	IMDS credential provider			
aws.ec2Me tadataSer viceEndpo intMode	IMDS credential provider			
aws.endpo intDiscov eryEnabled	Endpoint discovery			
aws.endpointUrl	Service-specific endpoints			
<pre>aws.endpo intUrl<se rvicename=""></se></pre>	Service-specific endpoints			
aws.ignor eConfigur edEndpointUrls	Service-specific endpoints			
aws.maxAttempts	Retry behavior			

JVM system properties list 59

Setting name	Details			
aws.profile	Using shared config and credentials files to globally configure Amazon SDKs and tools			
aws.region	Amazon Web Services Region			
aws.reque stChecksu mCalculation	Data Integrity Protections for Amazon S3			
<pre>aws.reque stMinComp ressionSi zeBytes</pre>	Request compression			
aws.respo nseChecks umValidation	Data Integrity Protections for Amazon S3			
aws.retryMode	Retry behavior			
aws.roleArn	Assume role credential provider			
aws.roleS essionName	Assume role credential provider			
aws.s3Dis ableMulti RegionAcc essPoints	Amazon S3 Multi-Region Access Points			
aws.s3Use ArnRegion	Amazon S3 access points			
aws.secre tAccessKey	Amazon access keys			

JVM system properties list 60

Setting name	Details			
aws.sessi onToken	Amazon access keys			
aws.share dCredenti alsFile	Finding and changing the location of the shared config and credentials files of Amazon SDKs and tools			
aws.useDu alstackEn dpoint	Dual-stack and FIPS endpoints			
aws.useFi psEndpoint	Dual-stack and FIPS endpoints			
aws.userA gentAppId	Application ID			
<pre>aws.webId entityTok enFile</pre>	Assume role credential provider			

Amazon SDKs and Tools standardized credential providers

Many credential providers have been standardized to consistent defaults and to work the same way across many SDKs. This consistency increases productivity and clarity when coding across multiple SDKs. All settings can be overridden in code. For details, see your specific SDK API.



Important

Not all SDKs support all providers, or even all aspects within a provider.

Topics

- Understand the credential provider chain
- SDK-specific and tool-specific credential provider chains

- Amazon access keys
- Assume role credential provider
- Container credential provider
- IAM Identity Center credential provider
- IMDS credential provider
- Process credential provider

Understand the credential provider chain

All SDKs have a series of places (or sources) that they check in order to find valid credentials to use to make a request to an Amazon Web Services service. After valid credentials are found, the search is stopped. This systematic search is called the credential provider chain.

When using one of the standardized credential providers, the Amazon SDKs always attempt to renew credentials automatically when they expire. The built-in credential provider chain provides your application with the ability to refresh your credentials regardless of which provider you are using in the chain. No additional code is required for the SDK to do this.

Although the distinct chain used by each SDK varies, they most often include sources such as the following:

Credential provider	Description
Amazon access keys	Amazon access keys for an IAM user (such as AWS_ACCES S_KEY_ID , and AWS_SECRET_ACCESS_KEY).
<u>OpenID Connect</u> - Assume role credential provider	Sign in using a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Assume the permissions of an IAM role using a JSON Web Token (JWT) from Amazon Security Token Service (Amazon STS).
IAM Identity Center credential provider	Get credentials from Amazon IAM Identity Center.

Credential provider	Description
Assume role credential provider	Get access to other resources by assuming the permissions of an IAM role. (Retrieve and then use temporary credentials for a role).
Container credential provider	Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS) credentials. The container credential provider fetches credentials for the customer's containerized application.
Process credential provider	Custom credential provider. Get your credentials from an external source or process, including IAM Roles Anywhere.
IMDS credential provider	Amazon Elastic Compute Cloud (Amazon EC2) instance profile credentials. Associate an IAM role with each of your EC2 instances. Temporary credentials for that role are made available to code running in the instance. The credentials are delivered through the Amazon EC2 metadata service.

For each step in the chain, there are multiple ways to assign setting values. Setting values that are specified in code always take precedence. However, there are also <u>Environment variables</u> and the <u>Using shared config and credentials files to globally configure Amazon SDKs and tools</u>. For more information, see <u>Precedence of settings</u>.

SDK-specific and tool-specific credential provider chains

To go directly to your SDK's or tool's **specific** credential provider chain details, choose your SDK or tool from the following:

- Amazon CLI
- SDK for C++
- SDK for Go
- SDK for Java
- SDK for JavaScript

- SDK for Kotlin
- SDK for .NET
- SDK for PHP
- SDK for Python (Boto3)
- SDK for Ruby
- SDK for Rust
- SDK for Swift
- Tools for PowerShell

Amazon access keys



Marning

To avoid security risks, don't use IAM users for authentication when developing purposebuilt software or working with real data. Instead, use federation with an identity provider such as Amazon IAM Identity Center.

Amazon access keys for an IAM user can be used as your Amazon credentials. The Amazon SDK automatically uses these Amazon credentials to sign API requests to Amazon, so that your workloads can access your Amazon resources and data securely and conveniently. It is recommended to always use the aws_session_token so that the credentials are temporary and no longer valid after they expire. Using long-term credentials is not recommended.



Note

If Amazon becomes unable to refresh these temporary credentials, Amazon may extend the validity of the credentials so that your workloads are not impacted.

The shared Amazon credentials file is the recommended location for storing credentials information because it is safely outside of application source directories and separate from the SDK-specific settings of the shared config file.

To learn more about Amazon credentials and using access keys, see Amazon security credentials and Managing access keys for IAM users in the IAM User Guide.

Configure this functionality by using the following:

aws_access_key_id - shared Amazon config file setting, aws_access_key_id - shared Amazon credentials file setting (recommended method), AWS_ACCESS_KEY_ID - environment variable, aws.accessKeyId - JVM system property: Java/Kotlin only

Specifies the Amazon access key used as part of the credentials to authenticate the user.

aws_secret_access_key - shared Amazon config file setting, aws_secret_access_key - shared Amazon credentials file setting (recommended method), AWS_SECRET_ACCESS_KEY - environment variable, aws.secretAccessKey - JVM system property: Java/Kotlin only

Specifies the Amazon secret key used as part of the credentials to authenticate the user.

aws_session_token - shared Amazon config file setting, aws_session_token - shared Amazon credentials file setting (recommended method), AWS_SESSION_TOKEN - environment variable, aws.sessionToken - JVM system property: Java/Kotlin only

Specifies an Amazon session token used as part of the credentials to authenticate the user. You receive this value as part of the temporary credentials returned by successful requests to assume a role. A session token is required only if you manually specify temporary security credentials. However, we recommend you always use temporary security credentials instead of long-term credentials. For security recommendations, see Security best practices in IAM.

For instructions on how to obtain these values, see <u>Using short-term credentials to authenticate</u> Amazon SDKs and tools.

Example of setting these required values in the config or credentials file:

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
```

Linux/macOS example of setting environment variables via command line:

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export
AWS_SESSION_TOKEN=AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
```

Windows example of setting environment variables via command line:

```
setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
setx
AWS_SESSION_TOKEN AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
```

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Si Notes or more information
Amazon CLI v2	Yes
SDK for C++	Yes shared config file not supported.
SDK for Go V2 (1.x)	Yes
SDK for Go 1.x (V1)	Yes To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes
SDK for Java 1.x	Yes
SDK for JavaScript 3.x	Yes
SDK for JavaScript 2.x	Yes
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes

SDK	Sı	Notes or more information
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
SDK for Swift	Yes	
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	Environment variables not supported.

Assume role credential provider



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Assuming a role involves using a set of temporary security credentials to access Amazon resources that you might not have access to otherwise. These temporary credentials consist of an access key ID, a secret access key, and a security token.

To set up your SDK or tool to assume a role, you must first create or identify a specific role to assume. IAM roles are uniquely identified by a role Amazon Resource Name (ARN). Roles establish trust relationships with another entity. The trusted entity that uses the role might be an Amazon Web Services service, another Amazon Web Services account, a web identity provider or OIDC, or SAML federation.

After the IAM role is identified, if you are trusted by that role, you can configure your SDK or tool to use the permissions that are granted by the role. To do this, use the following settings.

For guidance on getting started using these settings, see Assuming a role with Amazon credentials to authenticate Amazon SDKs and tools in this guide.

Assume role provider

Assume role credential provider settings

Configure this functionality by using the following:

credential_source - shared Amazon config file setting

Used within Amazon EC2 instances or Amazon Elastic Container Service containers to specify where the SDK or tool can find credentials that have permission to assume the role that you specify with the role_arn parameter.

Default value: None

Valid values:

- **Environment** Specifies that the SDK or tool is to retrieve source credentials from the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.
- **Ec2InstanceMetadata** Specifies that the SDK or tool is to use the <u>IAM role attached to the</u> EC2 instance profile to get source credentials.
- **EcsContainer** Specifies that the SDK or tool is to use the <u>IAM role attached to the ECS</u> container to get source credentials.

You cannot specify both credential_source and source_profile in the same profile.

Example of setting this in a config file to indicate that credentials should be sourced from Amazon EC2:

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

duration_seconds - shared Amazon config file setting

Specifies the maximum duration of the role session, in seconds.

This setting applies only when the profile specifies to assume a role.

Default value: 3600 seconds (one hour)

Valid values: The value can range from 900 seconds (15 minutes) up to the maximum session duration setting configured for the role (which can be a maximum of 43200 seconds, or 12 hours). For more information, see <u>View the Maximum Session Duration Setting for a Role</u> in the *IAM User Guide*.

Example of setting this in a config file:

```
duration_seconds = 43200
```

external_id - shared Amazon config file setting

Specifies a unique identifier that is used by third parties to assume a role in their customers' accounts.

This setting applies only when the profile specifies to assume a role and the trust policy for the role requires a value for ExternalId. The value maps to the ExternalId parameter that is passed to the AssumeRole operation when the profile specifies a role.

Default value: None.

Valid values: See <u>How to use an External ID When Granting Access to Your Amazon Resources</u> to a Third Party in the *IAM User Guide*.

Example of setting this in a config file:

```
external_id = unique_value_assigned_by_3rd_party
```

mfa_serial - shared Amazon config file setting

Specifies the identification or serial number of a multi-factor authentication (MFA) device that the user must use when assuming a role.

Required when assuming a role where the trust policy for that role includes a condition that requires MFA authentication. For more information about MFA, see <u>Amazon Multi-factor</u> authentication in IAM in the IAM User Guide.

Default value: None.

Valid values: The value can be either a serial number for a hardware device (such as GAHT12345678), or an Amazon Resource Name (ARN) for a virtual MFA device. The format of the ARN is: arn:aws-cn:iam::account-id:mfa/mfa-device-name

Example of setting this in a config file:

This example assumes a virtual MFA device, called MyMFADevice, that has been created for the account and enabled for a user.

mfa_serial = arn:aws-cn:iam::123456789012:mfa/MyMFADevice

role_arn - shared Amazon config file setting, AWS_ROLE_ARN - environment variable, aws.roleArn - JVM system property: Java/Kotlin only

Specifies the Amazon Resource Name (ARN) of an IAM role that you want to use to perform operations requested using this profile.

Default value: None.

Valid values: The value must be the ARN of an IAM role, formatted as follows: arn:aws-cn:iam::account-id:role/role-name

In addition, you must also specify **one** of the following settings:

- source_profile To identify another profile to use to find credentials that have permission to assume the role in this profile.
- credential_source To use either credentials identified by the current environment variables or credentials attached to an Amazon EC2 instance profile, or an Amazon ECS container instance.
- web_identity_token_file To use public identity providers or any OpenID Connect
 (OIDC)-compatible identity provider for users who have been authenticated in a mobile or
 web application.

role_session_name - shared Amazon config file setting, AWS_ROLE_SESSION_NAME - environment variable, aws.roleSessionName - JVM system property: Java/Kotlin only

Specifies the name to attach to the role session. This name appears in Amazon CloudTrail logs for entries associated with this session, which can be useful when auditing. For details, see CloudTrail userIdentity element in the Amazon CloudTrail User Guide.

Default value: An optional parameter. If you don't provide this value, a session name is generated automatically if the profile assumes a role.

Valid values: Provided to the RoleSessionName parameter when the Amazon CLI or Amazon API calls the AssumeRole operation (or operations such as the AssumeRoleWithWebIdentity operation) on your behalf. The value becomes part of the assumed role user Amazon Resource Name (ARN) that you can query, and shows up as part of the CloudTrail log entries for operations invoked by this profile.

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

Example of setting this in a config file:

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

source_profile - shared Amazon config file setting

Specifies another profile whose credentials are used to assume the role specified by the role_arn setting in the original profile. To understand how profiles are used in the shared Amazon config and credentials files, see Shared config and credentials files.

If you specify a profile that is also an assume role profile, each role will be assumed in sequential order to fully resolve the credentials. This chain is stopped when the SDK encounters a profile with credentials. Role chaining limits your Amazon CLI or Amazon API role session to a maximum of one hour and can't be increased. For more information, see Roles terms and concepts in the *IAM User Guide*.

Default value: None.

Valid values: A text string that consists of the name of a profile defined in the config and credentials files. You must also specify a value for role_arn in the current profile.

You cannot specify both credential_source and source_profile in the same profile.

Example of setting this in a config file:

```
[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
```

In the previous example, the A profile tells the SDK or tool to automatically look up the credentials for the linked B profile. In this case, the B profile uses the credential helper

tool provided by <u>Using IAM Roles Anywhere to authenticate Amazon SDKs and tools</u> to get credentials for the Amazon SDK. Those temporary credentials are then used by your code to access Amazon resources. The specified role must have attached IAM permissions policies that allow the requested code to run, such as the command, Amazon Web Services service, or API method. Every action that is taken by profile A has the role session name included in CloudTrail logs.

For a second example of role chaining, the following configuration can be used if you have an application on an Amazon Elastic Compute Cloud instance, and you want to have that application assume another role.

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

[profile B]
credential_source=Ec2InstanceMetadata
```

Profile A will use the credentials from the Amazon EC2 instance to assume the specified role and will renew the credentials automatically.

web_identity_token_file - shared Amazon config file setting,
AWS_WEB_IDENTITY_TOKEN_FILE - environment variable, aws.webIdentityTokenFile JVM system property: Java/Kotlin only

Specifies the path to a file that contains an access token from a <u>supported OAuth 2.0 provider</u> or OpenID Connect ID identity provider.

This setting enables authentication by using web identity federation providers, such as <u>Google</u>, <u>Facebook</u>, and <u>Amazon</u>, among many others. The SDK or developer tool loads the contents of this file and passes it as the WebIdentityToken argument when it calls the AssumeRoleWithWebIdentity operation on your behalf.

Default value: None.

Valid values: This value must be a path and file name. The file must contain an OAuth 2.0 access token or an OpenID Connect token that was provided to you by an identity provider. Relative paths are treated as relative to the working directory of the process.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Parti	<pre>credential_source not supported. duration_ seconds not supported. mfa_serial not supported.</pre>
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Parti	<pre>mfa_serial not supported. duration_seconds not supported.</pre>
SDK for Java 1.x	Parti	<pre>credential_source not supported.mfa_serial not supported. JVM system properties not supported.</pre>
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Parti	credential_source not supported.
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	

SDK	Sı	Notes or more information
SDK for Swift	Yes	
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	

Container credential provider



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this quide.

The container credential provider fetches credentials for customer's containerized application. This credential provider is useful for Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS) customers. SDKs attempt to load credentials from the specified HTTP endpoint through a GET request.

If you use Amazon ECS, we recommend you use a task IAM Role for improved credential isolation, authorization, and auditability. When configured, Amazon ECS sets the AWS_CONTAINER_CREDENTIALS_RELATIVE_URI environment variable that the SDKs and tools use to obtain credentials. To configure Amazon ECS for this functionality, see Task IAM role in the Amazon Elastic Container Service Developer Guide.

If you use Amazon EKS, we recommend you use Amazon EKS Pod Identity for improved credential isolation, least privilege, auditability, independent operation, reusability, and scalability. Both your Pod and an IAM role are associated with a Kubernetes service account to manage credentials for your applications. To learn more on Amazon EKS Pod Identity, see Amazon EKS Pod Identities in the Amazon EKS User Guide. When configured, Amazon EKS sets the AWS_CONTAINER_CREDENTIALS_FULL_URI and AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE environment variables that the SDKs and tools use to obtain credentials. For setup information, see Setting up the Amazon EKS Pod Identity Agent in the Amazon EKS User Guide or Amazon EKS Pod Identity simplifies IAM permissions for applications on Amazon EKS clusters at the Amazon Blog website.

Configure this functionality by using the following:

AWS_CONTAINER_CREDENTIALS_FULL_URI - environment variable

Specifies the full HTTP URL endpoint for the SDK to use when making a request for credentials. This includes both the scheme and the host.

Default value: None.

Valid values: Valid URI.

Note: This setting is an alternative to AWS_CONTAINER_CREDENTIALS_RELATIVE_URI and will only be used if AWS_CONTAINER_CREDENTIALS_RELATIVE_URI is not set.

Linux/macOS example of setting environment variables via command line:

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

or

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

AWS_CONTAINER_CREDENTIALS_RELATIVE_URI - environment variable

Specifies the relative HTTP URL endpoint for the SDK to use when making a request for credentials. The value is appended to the default Amazon ECS hostname of 169.254.170.2.

Default value: None.

Valid values: Valid relative URI.

Linux/macOS example of setting environment variables via command line:

export AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=/get-credentials?a=1

AWS_CONTAINER_AUTHORIZATION_TOKEN - environment variable

Specifies an authorization token in plain text. If this variable is set, the SDK will set the Authorization header on the HTTP request with the environment variable's value.

Default value: None.

Valid values: String.

Note: This setting is an alternative to AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE and will only be used if AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE is not set.

Linux/macOS example of setting environment variables via command line:

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

AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE - environment variable

Specifies an absolute file path to a file that contains the authorization token in plain text.

Default value: None.

Valid values: String.

Linux/macOS example of setting environment variables via command line:

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

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	
SDK for Java 2.x	Yes	When <u>Lambda SnapStart</u> is activated, AWS_CONTA INER_CREDENTIALS_FULL_URI and AWS_CONTA

SDK	Sı	Notes or more information
		INER_AUTHORIZATION_TOKEN are automatically used for authentication.
SDK for Java 1.x	Yes	When Lambda SnapStart is activated, AWS_CONTA INER_CREDENTIALS_FULL_URI and AWS_CONTA INER_AUTHORIZATION_TOKEN are automatically used for authentication.
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	When Lambda SnapStart is activated, AWS_CONTA INER_CREDENTIALS_FULL_URI and AWS_CONTA INER_AUTHORIZATION_TOKEN are automatically used for authentication.
SDK for .NET 3.x	Yes	When Lambda SnapStart is activated, AWS_CONTA INER_CREDENTIALS_FULL_URI and AWS_CONTA INER_AUTHORIZATION_TOKEN are automatically used for authentication.
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	When Lambda SnapStart is activated, AWS_CONTA INER_CREDENTIALS_FULL_URI and AWS_CONTA INER_AUTHORIZATION_TOKEN are automatically used for authentication.
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
SDK for Swift	Yes	
Tools for PowerShell V5	Yes	

SDK	Sı	Notes or more information
Tools for PowerShell V4	Yes	

IAM Identity Center credential provider



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

This authentication mechanism uses Amazon IAM Identity Center to get single sign-on (SSO) access to Amazon Web Services services for your code.



Note

In the Amazon SDK API documentation, the IAM Identity Center credential provider is called the SSO credential provider.

After you enable IAM Identity Center, you define a profile for its settings in your shared Amazon config file. This profile is used to connect to the IAM Identity Center access portal. When a user successfully authenticates with IAM Identity Center, the portal returns short-term credentials for the IAM role associated with that user. To learn how the SDK gets temporary credentials from the configuration and uses them for Amazon Web Services service requests, see How IAM Identity Center authentication is resolved for Amazon SDKs and tools.

There are two ways to configure IAM Identity Center through the config file:

- (Recommended) SSO token provider configuration Extended session durations. Includes support for custom session durations.
- Legacy non-refreshable configuration Uses a fixed, eight-hour session.

In both configurations, you need to sign in again when your session expires.

The following two guides contain additional information about IAM Identity Center:

- Amazon IAM Identity Center User Guide
- Amazon IAM Identity Center Portal API Reference

For a deep dive on how the SDKs and tools use and refresh credentials using this configuration, see How IAM Identity Center authentication is resolved for Amazon SDKs and tools.

Prerequisites

You must first enable IAM Identity Center. For details about enabling IAM Identity Center authentication, see Enabling Amazon IAM Identity Center in the Amazon IAM Identity Center User Guide.



Note

Alternatively, for complete prerequisites and the necessary shared config file configuration that is detailed on this page, see the guided instructions for setting up Using IAM Identity Center to authenticate Amazon SDK and tools.

SSO token provider configuration

When you use the SSO token provider configuration, your Amazon SDK or tool automatically refreshes your session up to your extended session period. For more information on session duration and maximum duration, see Configure the session duration of the Amazon access portal and IAM Identity Center integrated applications in the Amazon IAM Identity Center User Guide.

The sso-session section of the config file is used to group configuration variables for acquiring SSO access tokens, which can then be used to acquire Amazon credentials. For more details on this section within a config file, see Format of the config file.

The following shared config file example configures the SDK or tool using a dev profile to request IAM Identity Center credentials.

```
[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
```

The previous examples shows that you define an sso-session section and associate it to a profile. Typically, sso_account_id and sso_role_name must be set in the profile section so that the SDK can request Amazon credentials. sso_region, sso_start_url, and sso_registration_scopes must be set within the sso-session section.

sso_account_id and sso_role_name aren't required for all scenarios of SSO token configuration. If your application only uses Amazon Web Services services that support bearer authentication, then traditional Amazon credentials are not needed. Bearer authentication is an HTTP authentication scheme that uses security tokens called bearer tokens. In this scenario, sso_account_id and sso_role_name aren't required. See the individual Amazon Web Services service guide to determine if the service supports bearer token authorization.

Registration scopes are configured as part of an sso-session. Scope is a mechanism in OAuth 2.0 to limit an application's access to a user's account. The previous example sets sso_registration_scopes to provide necessary access for listing accounts and roles.

The following example shows how you can reuse the same sso-session configuration across multiple profiles.

```
[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
```

The authentication token is cached to disk under the ~/.aws/sso/cache directory with a file name based on the session name.

Legacy non-refreshable configuration

Automated token refresh isn't supported using the legacy non-refreshable configuration. We recommend using the SSO token provider configuration instead.

To use the legacy non-refreshable configuration, you must specify the following settings within your profile:

- sso_start_url
- sso_region
- sso_account_id
- sso_role_name

You specify the user portal for a profile with the sso_start_url and sso_region settings. You specify permissions with the sso_account_id and sso_role_name settings.

The following example sets the four required values in the config file.

```
[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
```

The authentication token is cached to disk under the ~/.aws/sso/cache directory with a file name based on the sso_start_url.

IAM Identity Center credential provider settings

Configure this functionality by using the following:

sso_start_url - shared Amazon config file setting

The URL that points to your organization's IAM Identity Center issuer URL or access portal URL. For more information, see <u>Using the Amazon access portal</u> in the *Amazon IAM Identity Center User Guide*.

To find this value, open the <u>IAM Identity Center console</u>, view the **Dashboard**, find **Amazon** access portal URL.

• Alternatively, starting with version **2.22.0** of the Amazon CLI, you can instead use the value for **Amazon Issuer URL**.

sso_region - shared Amazon config file setting

The Amazon Web Services Region that contains your IAM Identity Center portal host; that is, the Region you selected before enabling IAM Identity Center. This is independent from your default Amazon Region, and can be different.

For a complete list of the Amazon Web Services Regions and their codes, see <u>Regional Endpoints</u> in the *Amazon Web Services General Reference*. To find this value, open the <u>IAM Identity Center console</u>, view the **Dashboard**, and find **Region**.

sso_account_id - shared Amazon config file setting

The numeric ID of the Amazon Web Services account that was added through the Amazon Organizations service to use for authentication.

To see the list of available accounts, go to the <u>IAM Identity Center console</u> and open the **Amazon Web Services accounts** page. You can also see the list of available accounts using the <u>ListAccounts</u> API method in the *Amazon IAM Identity Center Portal API Reference*. For example, you can call the Amazon CLI method <u>list-accounts</u>.

sso_role_name - shared Amazon config file setting

The name of a permission set provisioned as an IAM role that defines the user's resulting permissions. The role must exist in the Amazon Web Services account specified by sso_account_id. Use the role name, not the role Amazon Resource Name (ARN).

Permission sets have IAM policies and custom permissions policies attached to them and define the level of access that users have to their assigned Amazon Web Services accounts.

To see the list of available permission sets per Amazon Web Services account, go to the IAM
IAM
Identity Center console
Identity Center console
Identity Center Portal API Reference
Identity Identity Id

sso_registration_scopes - shared Amazon config file setting

A comma-delimited list of valid scope strings to be authorized for the sso-session. An application can request one or more scopes, and the access token issued to the application is limited to the scopes granted. A minimum scope of sso:account:access must be granted to get a refresh token back from the IAM Identity Center service. For the list of available access scope options, see Access scopes in the Amazon IAM Identity Center User Guide.

These scopes define the permissions requested to be authorized for the registered OIDC client and access tokens retrieved by the client. Scopes authorize access to IAM Identity Center bearer token authorized endpoints.

This setting doesn't apply to the legacy non-refreshable configuration. Tokens issued using the legacy configuration are limited to scope sso: account: access implicitly.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	Configuration values also supported in credentials file.
SDK for Java 1.x	No	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	

SDK	Si Notes or more information
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Parti Legacy non-refreshable configuration only.
SDK for Swift	Yes
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

IMDS credential provider



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Instance Metadata Service (IMDS) provides data about your instance that you can use to configure or manage the running instance. For more information about the data available, see Work with instance metadata in the Amazon EC2 User Guide. Amazon EC2 provides a local endpoint available to instances that can provide various bits of information to the instance. If the instance has a role attached, it can provide a set of credentials that are valid for that role. The SDKs can use that endpoint to resolve credentials as part of their default credential provider chain. Instance Metadata Service Version 2 (IMDSv2), a more secure version of IMDS that uses a session token, is used by

default. If that fails due to a non-retryable condition (HTTP error codes 403, 404, 405), IMDSv1 is used as a fallback.

Configure this functionality by using the following:

AWS_EC2_METADATA_DISABLED - environment variable

Whether or not to attempt to use Amazon EC2 Instance Metadata Service (IMDS) to obtain credentials.

Default value: false.

Valid values:

- true Do not use IMDS to obtain credentials.
- false Use IMDS to obtain credentials.

ec2_metadata_v1_disabled - shared Amazon config file setting, AWS_EC2_METADATA_V1_DISABLED - environment variable, aws.disableEc2MetadataV1 -JVM system property: Java/Kotlin only

Whether or not to use Instance Metadata Service Version 1 (IMDSv1) as a fallback if IMDSv2 fails.



Note

New SDKs don't support IMDSv1 and, thus, don't support this setting. For details, see table Support by Amazon SDKs and tools.

Default value: false.

Valid values:

- true Do not use IMDSv1 as a fallback.
- false Use IMDSv1 as a fallback.

ec2_metadata_service_endpoint - shared Amazon config file setting, AWS_EC2_METADATA_SERVICE_ENDPOINT - environment variable, aws.ec2MetadataServiceEndpoint - JVM system property: Java/Kotlin only

The endpoint of IMDS. This value overrides the default location that Amazon SDKs and tools will search for Amazon EC2 instance metadata.

Default value: If ec2 metadata service endpoint mode equals IPv4, then default endpoint is http://169.254.169.254. If ec2_metadata_service_endpoint_mode equals IPv6, then default endpoint is http://[fd00:ec2::254].

Valid values: Valid URI.

ec2_metadata_service_endpoint_mode - shared Amazon config file setting, AWS EC2 METADATA SERVICE ENDPOINT MODE - environment variable, aws.ec2MetadataServiceEndpointMode - JVM system property: Java/Kotlin only

The endpoint mode of IMDS.

Default value: IPv4.

Valid values: IPv4, IPv6.



The IMDS credential provider is a part of the Understand the credential provider chain. However, the IMDS credential provider is only checked after several other providers that are in this series. Therefore, if you want your program use this provider's credentials, you must remove other valid credential providers from your configuration or use a different profile. Alternatively, instead of relying on the credential provider chain to automatically discover which provider returns valid credentials, specify the use of the IMDS credential provider in code. You can specify credential sources directly when you create service clients.

Security for IMDS credentials

By default, when the Amazon SDK is not configured with valid credentials the SDK will attempt to use the Amazon EC2 Instance Metadata Service (IMDS) to retrieve credentials for an Amazon role. This behavior can be disabled by setting the AWS_EC2_METADATA_DISABLED environment variable to true. This prevents unnecessary network activity and enhances security on untrusted networks where the Amazon EC2 Instance Metadata Service may be impersonated.



Note

Amazon SDK clients configured with valid credentials will never use IMDS to retrieve credentials, regardless of any of these settings.

Disabling use of Amazon EC2 IMDS credentials

How you set this environment variable depends on what operating system is in use as well as whether or not you want the change to be persistent.

Linux and macOS

Customers using Linux or macOS can set this environment variable with the following command:

```
$ export AWS_EC2_METADATA_DISABLED=true
```

If you want this setting to be persistent across multiple shell sessions and system restarts, you can add the above command to your shell profile file, such as .bash_profile, .zsh_profile, or .profile.

Windows

Customers using Windows can set this environment variable with the following command:

```
$ set AWS_EC2_METADATA_DISABLED=true
```

If you want this setting to be persistent across multiple shell sessions and system restarts can use the following command instead:

```
$ setx AWS_EC2_METADATA_DISABLED=true
```



The **setx** command does not apply the value to the current shell session, so you will need to reload or reopen the shell for the change to take effect.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	
SDK for Java 1.x	Parti	JVM system properties: Use com.amazonaws.sdk. disableEc2MetadataV1 instead of aws.disab leEc2MetadataV1 ; aws.ec2MetadataSer viceEndpoint and aws.ec2MetadataSer viceEndpointMode not supported.
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	Yes	Does not use IMDSv1 fallback.
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	Does not use IMDSv1 fallback.
SDK for Swift	Yes	

SDK	Sı	Notes or more information
Tools for PowerShell V5	Yes	You can disable IMDSv1 fallback explicitly in code using [Amazon.Util.EC2InstanceMetadata]::E C2MetadataV1Disabled = \$true .
Tools for PowerShell V4	Yes	You can disable IMDSv1 fallback explicitly in code using [Amazon.Util.EC2InstanceMetadata]::E C2MetadataV1Disabled = \$true .

Process credential provider



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

SDKs provide a way to extend the credential provider chain for custom use cases. This provider can be used to provide custom implementations, such as retrieving credentials from an on-premises credentials store or integrating with your on-premises identify provider.

For example, IAM Roles Anywhere uses credential_process to get temporary credentials on behalf of your application. To configure credential_process for this use, see Using IAM Roles Anywhere to authenticate Amazon SDKs and tools.



Note

The following describes a method of sourcing credentials from an external process and might be used if you are running software outside of Amazon. If you are building on an Amazon compute resource, use other credential providers. If using this option, you should make sure that the config file is as locked down as possible using security best practices for your operating system. Confirm that your custom credential tool does not write any secret information to StdErr, because the SDKs and Amazon CLI can capture and log such information, potentially exposing it to unauthorized users.

Configure this functionality by using the following:

credential_process - shared Amazon config file setting

Specifies an external command that the SDK or tool runs on your behalf to generate or retrieve authentication credentials to use. The setting specifies the name of a program/command that the SDK will invoke. When the SDK invokes the process, it waits for the process to write JSON data to stdout. The custom provider must return information in a specific format. That information contains the credentials that the SDK or tool can use to authenticate you.

Note

The process credential provider is a part of the Understand the credential provider chain. However, the process credential provider is only checked after several other providers that are in this series. Therefore, if you want your program use this provider's credentials, you must remove other valid credential providers from your configuration or use a different profile. Alternatively, instead of relying on the credential provider chain to automatically discover which provider returns valid credentials, specify the use of the process credential provider in code. You can specify credential sources directly when you create service clients.

Specifying the path to the credentials program

The setting's value is a string that contains a path to a program that the SDK or development tool runs on your behalf:

- The path and file name can consist of only these characters: A-Z, a-z, 0-9, hyphen (), underscore (_), period (.), forward slash (/), backslash (\), and space.
- If the path or file name contains a space, surround the complete path and file name with doublequotation marks (" ").
- If a parameter name or a parameter value contains a space, surround that element with doublequotation marks (" "). Surround only the name or value, not the pair.
- Don't include any environment variables in the strings. For example, don't include \$HOME or %USERPROFILE%.
- Don't specify the home folder as ~. * You must specify either the full path or a base file name. If there is a base file name, the system attempts to find the program within folders specified by the PATH environment variable. The path varies depending on the operating system:

The following example shows setting credential_process in the shared config file on Linux/macOS.

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

The following example shows setting credential_process in the shared config file on Windows.

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

• Can be specified within a dedicated profile:

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

Valid output from the credentials program

The SDK runs the command as specified in the profile and then reads data from the standard output stream. The command you specify, whether a script or binary program, must generate JSON output on STDOUT that matches the following syntax.

```
"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

As of this writing, the Version key must be set to 1. This might increment over time as the structure evolves.

The Expiration key is an RFC3339 formatted timestamp. If the Expiration key isn't present in the tool's output, the SDK assumes that the credentials are long-term credentials that don't refresh. Otherwise, the credentials are considered temporary credentials, and they are automatically refreshed by rerunning the credential_process command before the credentials expire.



Note

The SDK does **not** cache external process credentials the way it does assume-role credentials. If caching is required, you must implement it in the external process.

The external process can return a non-zero return code to indicate that an error occurred while retrieving the credentials.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	

SDK	Si Notes or more information
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes
SDK for Swift	Yes
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

Amazon SDKs and Tools standardized features

Many features have been standardized to consistent defaults and to work the same way across many SDKs. This consistency increases productivity and clarity when coding across multiple SDKs. All settings can be overridden in code, see your specific SDK API for details.



Not all SDKs support all features, or even all aspects within a feature.

Topics

- Account-based endpoints
- Application ID
- Amazon EC2 instance metadata
- Amazon S3 access points

Standardized features

- Amazon S3 Multi-Region Access Points
- **Amazon Web Services Region**
- Amazon STS Regional endpoints
- Data Integrity Protections for Amazon S3
- **Dual-stack and FIPS endpoints**
- **Endpoint discovery**
- General configuration settings
- Host prefix injection
- IMDS client
- Retry behavior
- Request compression
- Service-specific endpoints
- Smart configuration defaults

Account-based endpoints



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Account-based endpoints help ensure high performance and scalability by using your Amazon Web Services account ID to route requests for services that support this feature. When you use an Amazon SDK and service that support account-based endpoints, the SDK client constructs and uses an account-based endpoint rather than a regional endpoint. If the account ID isn't visible to the SDK client, the client will use the regional endpoint. Account-based endpoints take the form of https://<account-id>.ddb.<region>.amazonaws.com, where <account-id> and <region> are your Amazon Web Services account ID and Amazon Web Services Region.

Configure this functionality by using the following:

Account-based endpoints

aws_account_id - shared Amazon config file setting, AWS_ACCOUNT_ID - environment variable, aws.accountId - JVM system property: Java/Kotlin only

The Amazon Web Services account ID. Used for account-based endpoint routing. An Amazon Web Services account ID has a format like 111122223333.

Account-based endpoint routing provides better request performance for some services.

account_id_endpoint_mode - shared Amazon config file setting,
AWS_ACCOUNT_ID_ENDPOINT_MODE - environment variable, aws.accountIdEndpointMode JVM system property: Java/Kotlin only

This setting is used to turn off account-based endpoint routing if necessary, and bypass account-based rules.

Default value: preferred

Valid values:

- **preferred** The endpoint should include account ID if available.
- disabled A resolved endpoint doesn't include account ID.
- required The endpoint must include account ID. If the account ID isn't available, the SDK throws an error.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Supp	Released in SDK version	Notes or more information
Amazon CLI v2	Yes	2.25.0	
Amazon CLI v1	Yes	1.38.0	
SDK for C++	No		
SDK for Go V2 (1.x)	Yes	v1.35.0	

Account-based endpoints 95

SDK	Supp	Released in SDK version	Notes or more information
SDK for Go 1.x (V1)	No		
SDK for Java 2.x	Yes	v2.28.4	
SDK for Java 1.x	Yes	v1.12.771	
SDK for JavaScript 3.x	Yes	v3.656.0	
SDK for JavaScript 2.x	No		
SDK for Kotlin	Yes	v1.3.37	
SDK for .NET 4.x	Yes	4.0.0	
SDK for .NET 3.x	No		
SDK for PHP 3.x	Yes	v3.318.0	
SDK for Python (Boto3)	Yes	1.37.0	
SDK for Ruby 3.x	Yes	v1.123.0	
SDK for Rust	No		
SDK for Swift	Yes	1.2.0	
Tools for PowerShel l V5	No		
Tools for PowerShel	No		

Account-based endpoints 96

Application ID



(i) Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

A single Amazon Web Services account can be used by multiple customer applications to make calls to Amazon Web Services services. Application ID provides a way for customers to identify which source application made a set of calls using an Amazon Web Services account. Amazon SDKs and services don't use or interpret this value other than to surface it back in customer communications. For example, this value can be included in operational emails or in the Amazon Health Dashboard to uniquely identify which of your applications is associated with the notification.

Configure this functionality by using the following:

sdk_ua_app_id - shared Amazon config file setting, AWS_SDK_UA_APP_ID - environment variable, aws.userAgentAppId - JVM system property: Java/Kotlin only

This setting is a unique string you assign to your application to identify which of your applications within a particular Amazon Web Services account makes calls to Amazon.

Default value: None

Valid values: String with maximum length of 50. Letters, numbers and the following special characters are allowed: !,\$,%,&,*,+,-,.,,^,_,`,|,~.

Example of setting this value in the config file:

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

Linux/macOS example of setting environment variables via command line:

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

Application ID 97

Windows example of setting environment variables via command line:

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

If you include symbols that have a special meaning to the shell being used, escape the value as appropriate.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	shared config file not supported.
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	Parti	Shared config file setting not supported; environment variable not supported.
SDK for Java 1.x	No	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	No	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	

Application ID 98

SDK	Sı	Notes or more information
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
SDK for Swift	No	
Tools for PowerShell V5	No	
Tools for PowerShell V4	No	

Amazon EC2 instance metadata



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Amazon EC2 provides a service on instances called the Instance Metadata Service (IMDS). To learn more about this service, see Work with instance metadata in the Amazon EC2 User Guide. When attempting to retrieve credentials on an Amazon EC2 instance that has been configured with an IAM role, the connection to the instance metadata service is adjustable.

Configure this functionality by using the following:

metadata_service_num_attempts - shared Amazon config file setting, AWS_METADATA_SERVICE_NUM_ATTEMPTS - environment variable

This setting specifies the number of total attempts to make before giving up when attempting to retrieve data from the instance metadata service.

Default value: 1

Valid values: Number greater than or equal to 1.

Amazon EC2 instance metadata

metadata_service_timeout - shared Amazon config file setting, AWS_METADATA_SERVICE_TIMEOUT - environment variable

Specifies the number of seconds before timing out when attempting to retrieve data from the instance metadata service.

Default value: 1

Valid values: Number greater than or equal to 1.

Example of setting these values in the config file:

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

Linux/macOS example of setting environment variables via command line:

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

Windows example of setting environment variables via command line:

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

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	No	
SDK for Go V2 (1.x)	No	

Amazon EC2 instance metadata 100

SDK	Sı	Notes or more information	
SDK for Go 1.x (V1)	No		
SDK for Java 2.x	Parti	Only AWS_METADATA_SERVICE_TIMEOUT	is supported.
SDK for Java 1.x	Parti	Only AWS_METADATA_SERVICE_TIMEOUT	is supported.
SDK for JavaScript 3.x	No		
SDK for JavaScript 2.x	No		
SDK for Kotlin	No		
SDK for .NET 4.x	No		
SDK for .NET 3.x	No		
SDK for PHP 3.x	Yes		
SDK for Python (Boto3)	Yes		
SDK for Ruby 3.x	No		
SDK for Rust	No		
SDK for Swift	No		
Tools for PowerShell V5	No		
Tools for PowerShell V4	No		

Amazon S3 access points



guide.

For help in understanding the layout of settings pages, or in interpreting the **Support by Amazon SDKs and tools** table that follows, see <u>Understanding the settings pages of this</u>

Amazon S3 access points 101

The Amazon S3 service provides access points as an alternative way to interact with Amazon S3 buckets. Access points have unique policies and configurations that can be applied to them instead of directly to the bucket. With Amazon SDKs, you can use access point Amazon Resource Names (ARNs) in the bucket field for API operations instead of specifying the bucket name explicitly. They are used for specific operations such as using an access point ARN with GetObject to fetch an object from a bucket, or using an access point ARN with PutObject to add an object to a bucket.

To learn more about Amazon S3 access points and ARNs, see <u>Using access points</u> in the *Amazon S3* User Guide.

Configure this functionality by using the following:

s3_use_arn_region - shared Amazon config file setting, AWS_S3_USE_ARN_REGION - environment variable, aws.s3UseArnRegion - JVM system property: Java/Kotlin only, To configure value directly in code, consult your specific SDK directly.

This setting controls whether the SDK uses the access point ARN Amazon Web Services Region to construct the Regional endpoint for the request. The SDK validates that the ARN Amazon Web Services Region is served by the same Amazon partition as the client's configured Amazon Web Services Region to prevent cross-partition calls that most likely will fail. If multiply defined, the code-configured setting takes precedence, followed by the environment variable setting.

Default value: false

Valid values:

- **true** The SDK uses the ARN's Amazon Web Services Region when constructing the endpoint instead of the client's configured Amazon Web Services Region. Exception: If the client's configured Amazon Web Services Region is a FIPS Amazon Web Services Region, then it must match the ARN's Amazon Web Services Region. Otherwise, an error will result.
- **false** The SDK uses the client's configured Amazon Web Services Region when constructing the endpoint.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

Amazon S3 access points 102

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	JVM system property not supported.
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	Doesn't follow standard precedence; shared config file value takes precedence over environment variable.
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	No	
SDK for Swift	No	
Tools for PowerShell V5	Yes	Doesn't follow standard precedence; shared config file value takes precedence over environment variable.
Tools for PowerShell V4	Yes	Doesn't follow standard precedence; shared config file value takes precedence over environment variable.

Amazon S3 access points 103

Amazon S3 Multi-Region Access Points



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from Amazon S3 buckets located in multiple Amazon Web Services Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture used in a single Region, and then run those applications anywhere in the world.

To learn more about Multi-Region Access Points, see Multi-Region Access Points in Amazon S3 in the Amazon S3 User Guide.

To learn more about Multi-Region Access Point Amazon Resource Names (ARNs), see Making requests using a Multi-Region Access Point in the Amazon S3 User Guide.

To learn more about creating Multi-Region Access Points, see Managing Multi-Region Access Points in the Amazon S3 User Guide.

The SigV4A algorithm is the signing implementation used to sign the global Region requests. This algorithm is obtained by the SDK through a dependency on the Amazon Common Runtime (CRT) libraries.

Configure this functionality by using the following:

s3_disable_multiregion_access_points - shared Amazon config file setting, AWS_S3_DISABLE_MULTIREGION_ACCESS_POINTS - environment variable, aws.s3DisableMultiRegionAccessPoints - JVM system property: Java/Kotlin only, To configure value directly in code, consult your specific SDK directly.

This setting controls whether the SDK potentially attempts cross-Region requests. If multiply defined, the code-configured setting takes precedence, followed by the environment variable setting.

Default value: false

Valid values:

- **true** Stops the use of cross-Region requests.
- false Enables cross-Region requests using Multi-Region Access Points.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Si Notes or more information
Amazon CLI v2	Yes
SDK for C++	Yes
SDK for Go V2 (1.x)	Yes
SDK for Go 1.x (V1)	No
SDK for Java 2.x	Yes
SDK for Java 1.x	No
SDK for JavaScript 3.x	Yes
SDK for JavaScript 2.x	No
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes

SDK	Sı	Notes or more information
SDK for Swift	No	
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	

Amazon Web Services Region



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Amazon Web Services Regions are an important concept to understand when working with Amazon Web Services services.

With Amazon Web Services Regions, you can access Amazon Web Services services that physically reside in a specific geographic area. This can be useful to keep your data and applications running close to where you and your users will access them. Regions provide fault tolerance, stability, and resilience, and can also reduce latency. With Regions, you can create redundant resources that remain available and unaffected by a Regional outage.

Most Amazon Web Services service requests are associated with a particular geographic region. The resources that you create in one Region do not exist in any other Region unless you explicitly use a replication feature offered by an Amazon Web Services service. For example, Amazon S3 and Amazon EC2 support cross-Region replication. Some services, such as IAM, do not have Regional resources.

The Amazon Web Services General Reference contains information on the following:

- To understand the relationship between Regions and endpoints, and to view a list of existing Regional endpoints, see Amazon service endpoints.
- To view the current list of all supported Regions and endpoints for each Amazon Web Services service, see Service endpoints and quotas.

Amazon Web Services Region 106

Creating service clients

To programmatically access Amazon Web Services services, SDKs use a client class/object for each Amazon Web Services service. If your application needs to access Amazon EC2, for example, your application would create an Amazon EC2 client object to interface with that service.

If no Region is explicitly specified for the client in the code itself, the client defaults to using the Region that is set through the following region setting. However, the active Region for a client can be explicitly set for any individual client object. Setting the Region in this way takes precedence over any global setting for that particular service client. The alternative Region is specified during instantiation of that client, specific to your SDK (check your specific SDK Guide or your SDK's code base).

Configure this functionality by using the following:

region - shared Amazon config file setting, AWS_REGION - environment variable, aws.region - JVM system property: Java/Kotlin only

Specifies the default Amazon Web Services Region to use for Amazon requests. This Region is used for SDK service requests that aren't provided with a specific Region to use.

Default value: None. You must specify this value explicitly.

Valid values:

- Any of the Region codes available for the chosen service, as listed in <u>Amazon service</u> <u>endpoints</u> in the <u>Amazon General Reference</u>. For example, the value us-east-1 sets the endpoint to the Amazon Web Services Region US East (N. Virginia).
- aws-global specifies the global endpoint for services that support a separate global endpoint in addition to Regional endpoints, such as Amazon Security Token Service (Amazon STS) and Amazon Simple Storage Service (Amazon S3).

Example of setting this value in the config file:

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

Linux/macOS example of setting environment variables via command line:

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

Amazon Web Services Region 107

Windows example of setting environment variables via command line:

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

Most SDKs have a "configuration" object that is available for setting the default Region from within the application code. For details, see your specific Amazon SDK developer guide.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	Amazon CLI v2 uses any value in AWS_REGION before any value in AWS_DEFAULT_REGION (both variables are checked).
Amazon CLI v1	Yes	Amazon CLI v1 uses environment variable named AWS_DEFAULT_REGION for this purpose.
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	

Amazon Web Services Region 108

SDK	Su Notes or more information
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes This SDK uses environment variable named AWS_DEFAU LT_REGION for this purpose.
SDK for Ruby 3.x	Yes
SDK for Rust	Yes
SDK for Swift	Yes
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

Amazon STS Regional endpoints



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Amazon Security Token Service (Amazon STS) is available both as a global and Regional service. Some of Amazon SDKs and CLIs use the global service endpoint (https:// sts.amazonaws.com) by default, while some use the Regional service endpoints (https:// sts. {region_identifier}. {partition_domain}). In Regions that are enabled by default, requests to the Amazon STS global endpoint are automatically served in the same Region where the request originates. In opt-in Regions, requests to the Amazon STS global endpoint are served by a single Amazon Web Services Region, US East (N. Virginia). For more information on Amazon STS endpoints, see Endpoints in the Amazon Security Token Service API Reference or Manage Amazon STS in an Amazon Web Services Region in the Amazon Identity and Access Management User Guide.

It is an Amazon best practice to use Regional endpoints whenever possible and to configure your Amazon Web Services Region. Customers in partitions other than commercial must use Regional endpoints. Not all SDKs and tools support this setting, but all have defined behavior around global and Regional endpoints. See the following section for more information.



Amazon has made changes to the Amazon Security Token Service (Amazon STS) global endpoint (https://sts.amazonaws.com) in Regions enabled by default to enhance its resiliency and performance. Amazon STS requests to the global endpoint are automatically served in the same Amazon Web Services Region as your workloads. These changes will not be deployed to opt-in Regions. We recommend that you use the appropriate Amazon STS regional endpoints. For more information, see Amazon STS global endpoint changes in the Amazon Identity and Access Management User Guide.

For SDKs and tools that support this setting, customers can configure the functionality by using the following:

sts_regional_endpoints - shared Amazon config file setting, AWS_STS_REGIONAL_ENDPOINTS - environment variable

This setting specifies how the SDK or tool determines the Amazon Web Services service endpoint that it uses to talk to the Amazon Security Token Service (Amazon STS).

Default value: legacy



Note

All new SDK major versions releasing after July 2022 will default to regional. New SDK major versions might remove this setting and use regional behavior. To reduce future impact regarding this change, we recommend you start using regional in your application when possible.

Valid values: (Recommended value: regional)

• legacy – Uses the global Amazon STS endpoint, sts.amazonaws.com.

• regional – The SDK or tool always uses the Amazon STS endpoint for the currently configured Region. For example, if the client is configured to use us-west-2, all calls to Amazon STS are made to the Regional endpoint sts.us-west-2.amazonaws.com, instead of the global sts.amazonaws.com endpoint. To send a request to the global endpoint while this setting is enabled, you can set the Region to aws-global.

Example of setting these values in the config file:

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

Linux/macOS example of setting environment variables via command line:

```
export AWS_STS_REGIONAL_ENDPOINTS=regional
```

Windows example of setting environment variables via command line:

```
setx AWS_STS_REGIONAL_ENDPOINTS regional
```

Support by Amazon SDKs and tools



Note

It is an Amazon best practice to use Regional endpoints whenever possible and to configure your Amazon Web Services Region.

The table that follows summarizes, for your SDK or tool:

- Supports setting: Whether the shared config file variable and environment variable for STS Regional endpoints are supported.
- **Default setting value**: The default value of the setting if it is supported.
- Default service client target STS Endpoint: What default endpoint is used by the client even if the setting to change it is not available.
- Service client fallback behavior: What the SDK does when it is supposed to use a Regional endpoint but no Region has been configured. This is the behavior regardless of if it is using a Regional endpoint because of a default or because regional has been selected by the setting.

The table also uses the following values:

- **Global endpoint**: https://sts.amazonaws.com.
- **Regional endpoint**: Based on the configured <u>Amazon Web Services Region</u> used by your application.

• us-east-1 (Regional): Uses the us-east-1 Region endpoint but with longer session tokens than typical global requests.

SDK	Defau settin value		Service client fallback behavior	Notes or more information
Amazon CLI v2	N N/A	Regional endpoint	Global endpoint	
Amazon CLI v1	Ye legac	Global endpoint	Global endpoint	
SDK for C++	N N/A	Regional endpoint	us-east-1 (Regional)	
SDK for Go V2 (1.x)	N N/A	Regional endpoint	Request failure	
SDK for Go 1.x (V1)	Y∈ legac	gy Global endpoint	Global endpoint	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	N N/A	Regional endpoint	Request failure	If no Region is configure d, the AssumeRole and AssumeRoleWithWebI dentity will use the global STS endpoint.

SDK	!	Default setting value	Default service client target STS Endpoint	Service client fallback behavior	Notes or more information
SDK for Java 1.x	Υє	legacy	Global endpoint	Global endpoint	
SDK for JavaScript 3.x	N	N/A	Regional endpoint	Request failure	
SDK for JavaScript 2.x	Υє	legacy	Global endpoint	Global endpoint	
SDK for Kotlin	N	N/A	Regional endpoint	Global endpoint	
SDK for .NET 4.x	N	N/A	Regional endpoint	us-east-1 (Regional)	
SDK for .NET 3.x	Υє	legacy	Global endpoint	Global endpoint	
SDK for PHP 3.x	Υє	legacy	Global endpoint	Request failure	
SDK for Python (Boto3)	Υє	legacy	Global endpoint	Global endpoint	
SDK for Ruby 3.x	Υє	regional	Regional endpoint	Request failure	
SDK for Rust	N	N/A	Regional endpoint	Request failure	

SDK	:	Default setting value	Default service client target STS Endpoint	Service client fallback behavior	Notes or more information
SDK for Swift	N	N/A	Regional endpoint	Request failure	
Tools for PowerShell V5	Υє	legacy	Global endpoint	Global endpoint	
Tools for PowerShell V4	Υє	legacy	Global endpoint	Global endpoint	

Data Integrity Protections for Amazon S3



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

For some time, Amazon SDKs have supported data integrity checks when uploading data to or downloading data from Amazon Simple Storage Service. Previously, these checks were optin. Now, we've enabled these checks by default, using CRC-based algorithms such as CRC32 or CRC64NVME. Although each SDK or tool has a default algorithm, you can choose a different algorithm. You can also continue to still manually supply a pre-calculated checksum for uploads if you want. Consistent behavior across uploads, multipart uploads, downloads, and encryption modes simplifies client-side integrity checks.

The latest versions of our Amazon SDKs and Amazon CLI automatically calculate a cyclic redundancy check (CRC)-based checksum for each upload and sends it to Amazon S3. Amazon S3 independently calculates a checksum on the server side and validates it against the provided

value before durably storing the object and its checksum in the object's metadata. By storing the checksum in the metadata alongside the object, when the object is downloaded, the same checksum can be automatically returned and used to validate downloads as well. You can also verify the checksum stored in the object's metadata at any time.

To learn more about checksum operations, multipart uploads, or the list of supported checksum algorithms, see Checking object integrity in Amazon S3 in the Amazon Simple Storage Service User Guide.

Multipart uploads:

Amazon S3 also provides developers with consistent full object checksums across single part and multipart uploads.

When uploading files in multiple parts, the SDKs calculate checksums for each part. Amazon S3 uses these checksums to verify the integrity of each part through the UploadPart API. Additionally, Amazon S3 validates the entire file's size and checksum when you call the CompleteMultipartUpload API.

If your SDK has an Amazon S3 Transfer Manager to assist with multipart uploads, the checksums are validated for the parts using the SDK-specific default algorithm found in the <u>Support by Amazon SDKs and tools</u> table. You can opt-in to a full object checksum by setting setting checksum_type to FULL_OBJECT or by choosing to use the CRC64NVME algorithm.

If you are using an older version of SDK or Amazon CLI:

If your application uses a version prior to December 2024 of the SDK or tool, Amazon S3 still computes a CRC64NVME checksum on new objects and stores it in the object metadata for future reference. You can later compare the stored CRC with a CRC computed on your side and verify the network transmission was correct. Also, you can still manually extend the integrity protection by providing your own precomputed checksums with your Put0bject or UploadPart requests, which is the standard technique for addressing this in older versions.

Configure this functionality by using the following:

request_checksum_calculation - shared Amazon config file
setting, AWS_REQUEST_CHECKSUM_CALCULATION - environment variable,
aws.requestChecksumCalculation - JVM system property: Java/Kotlin only

By default, users are opted-in to calculating a request checksum when sending a request. The user can choose any of the <u>available checksum algorithms</u> as a part of building the request. Otherwise, an SDK-specific default algorithm is used. See the <u>Support by Amazon SDKs and tools</u> table for the default algorithm for each SDK or tool.

Default value: WHEN_SUPPORTED

Valid values:

- WHEN_SUPPORTED Checksum validation is performed on all response payloads when supported by the API operation, such as data transfers to Amazon S3.
- WHEN_REQUIRED Checksum validation is performed only when required by the API operation.

response_checksum_validation - shared Amazon config file setting, AWS_RESPONSE_CHECKSUM_VALIDATION - environment variable, aws.responseChecksumValidation - JVM system property: Java/Kotlin only

By default, users are opted-in to a response checksum validation when sending a request. A checksum is calculated for the response payload and compared against the checksum response header. If checksum validation fails, an error is raised to the user when the payload is read.

The checksum response header also indicates the algorithm for the checksum. The Amazon S3 client attempts to validate response checksums for all Amazon S3 API operations that support checksums. However, if the SDK has not implemented the specified checksum algorithm then this validation is skipped.

Default value: WHEN_SUPPORTED

Valid values:

- WHEN_SUPPORTED Checksum validation is performed on all response payloads when supported by the API operation, such as data transfers to Amazon S3.
- WHEN_REQUIRED Checksum validation is performed only when supported by the API
 operation and the caller has explicitly enabled checksum for the operation. For example,
 when the Amazon S3 GetObject API is called and the ChecksumMode parameter is set to
 enabled.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.



Note

In the following table, 'CRT' refers to the Amazon Common Runtime (CRT) libraries and might require adding an additional dependency to your project.

SDK	Supp	Default checksum algorithm	Supported checksum algorithms	Notes or more informati on
Amazon CLI v2	Yes	CRC64NVME	CRC64NVME, CRC32, CRC32C, SHA1, SHA256	For Amazon CLI v1, the default algorithm and the supported algorithms will be identical to Python (Boto3).
SDK for C++	Yes	CRC64NVME	CRC64NVME, CRC32, CRC32C, SHA1, SHA256	
SDK for Go V2 (1.x)	Yes	CRC32	CRC64NVME, CRC32, CRC32C, SHA1, SHA256	
SDK for Go 1.x (V1)	No			
SDK for Java 2.x	Yes	CRC32	CRC64NVME (via CRT only), CRC32, CRC32C, SHA1, SHA256	

SDK	Supp	Default checksum algorithm	Supported checksum algorithms	Notes or more informati on
SDK for Java 1.x	No			
SDK for JavaScript 3.x	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	
SDK for JavaScript 2.x	No			
SDK for Kotlin	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	
SDK for .NET 4.x	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	
SDK for .NET 3.x	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	
SDK for PHP 3.x	Yes	CRC32	CRC32, CRC32C (via CRT only), SHA1, SHA256	awscrt extension is required in order to use CRC32C.
SDK for Python (Boto3)	Yes	CRC32	CRC64NVME (via CRT only), CRC32, CRC32C (via CRT only), SHA1, SHA256	
SDK for Ruby 3.x	Yes	CRC32	CRC64NVME (via CRT only), CRC32, CRC32C (via CRT only), SHA1, SHA256	
SDK for Rust	Yes	CRC32	CRC64NVME, CRC32, CRC32C, SHA1, SHA256	

SDK	Supp	Default checksum algorithm	Supported checksum algorithms	Notes or more informati on
SDK for Swift	Yes	CRC32	CRC64NVME, CRC32, CRC32C, SHA1, SHA256	CRT dependency required for all algorithms.
Tools for PowerShell V5	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	
Tools for PowerShell V4	Yes	CRC32	CRC32, CRC32C, SHA1, SHA256	

Dual-stack and FIPS endpoints



Note

For help in understanding the layout of settings pages, or in interpreting the Support by Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Configure this functionality by using the following:

use_dualstack_endpoint - shared Amazon config file setting, AWS_USE_DUALSTACK_ENDPOINT - environment variable, aws.useDualstackEndpoint - JVM system property: Java/Kotlin only

Turns on or off whether the SDK will send requests to dual-stack endpoints. To learn more about dual-stack endpoints, which support both IPv4 and IPv6 traffic, see Using Amazon S3 dual-stack endpoints in the Amazon Simple Storage Service User Guide. Dual-stack endpoints are available for some services in some regions.

Default value: false

Valid values:

• **true** – The SDK or tool will attempt to use dual-stack endpoints to make network requests. If a dual-stack endpoint does not exist for the service and/or Amazon Web Services Region, the request will fail.

• false – The SDK or tool will not use dual-stack endpoints to make network requests.

use_fips_endpoint - shared Amazon config file setting, AWS_USE_FIPS_ENDPOINT - environment variable, aws.useFipsEndpoint - JVM system property: Java/Kotlin only

Turns on or off whether the SDK or tool will send requests to FIPS-compliant endpoints. The Federal Information Processing Standards (FIPS) are a set of US Government security requirements for data and its encryption. Government agencies, partners, and those wanting to do business with the federal government are required to adhere to FIPS guidelines. Unlike standard Amazon endpoints, FIPS endpoints use a TLS software library that complies with FIPS 140-2. If this setting is enabled and a FIPS endpoint does not exist for the service in your Amazon Web Services Region, the Amazon call may fail. Service-specific endpoints and the --endpoint-url option for the Amazon Command Line Interface override this setting.

To learn more about other ways to specify FIPS endpoints by Amazon Web Services Region, see <u>FIPS Endpoints by Service</u>. For more information on Amazon Elastic Compute Cloud service endpoints, see <u>Dual-stack</u> (IPv4 and IPv6) endpoints in the *Amazon EC2 API Reference*.

Default value: false

Valid values:

- true The SDK or tool will send requests to FIPS-compliant endpoints.
- false The SDK or tool will not send requests to FIPS-compliant endpoints.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	

SDK	Si Notes or more information
SDK for Go V2 (1.x)	Yes
SDK for Go 1.x (V1)	Yes To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes
SDK for Java 1.x	No
SDK for JavaScript 3.x	Yes
SDK for JavaScript 2.x	Yes
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes
SDK for Swift	Yes
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

Endpoint discovery



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

SDKs use endpoint discovery to access service endpoints (URLs to access various resources), while still maintaining flexibility for Amazon to alter URLs as needed. This way, your code can automatically detect new endpoints. There are no fixed endpoints for some services. Instead, you get the available endpoints during runtime by making a request to get the endpoints first. After retrieving the available endpoints, the code then uses the endpoint to access other operations. For example, for Amazon Timestream, the SDK makes a DescribeEndpoints request to retrieve the available endpoints, and then uses those endpoints to complete specific operations such as CreateDatabase or CreateTable.

Configure this functionality by using the following:

endpoint_discovery_enabled - shared Amazon config file setting, AWS_ENABLE_ENDPOINT_DISCOVERY - environment variable, aws.endpointDiscoveryEnabled - JVM system property: Java/Kotlin only, To configure value directly in code, consult your specific SDK directly.

Turns on or off endpoint discovery for DynamoDB.

Endpoint discovery is required in Timestream and optional in Amazon DynamoDB. This setting defaults to either true or false depending on whether the service requires endpoint discovery. Timestream requests default to true, and Amazon DynamoDB requests default to false.

Valid values:

- true The SDK should automatically attempt to discover an endpoint for services where endpoint discovery is optional.
- false The SDK should not automatically attempt to discover an endpoint for services where endpoint discovery is optional.

Endpoint discovery 122

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	Yes	To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	Yes	The SDK for Java 2.x uses AWS_ENDPOINT_DISCO VERY_ENABLED for the environment variable name.
SDK for Java 1.x	Parti	JVM system property not supported.
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Parti	Supported for Timestream only.
SDK for Swift	No	

Endpoint discovery 123

SDK	Sı	Notes or more information
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	

General configuration settings



Note

For help in understanding the layout of settings pages, or in interpreting the Support by Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

SDKs support some general settings that configure overall SDK behaviors.

Configure this functionality by using the following:

api_versions - shared Amazon config file setting

Some Amazon services maintain multiple API versions to support backward compatibility. By default, SDK and Amazon CLI operations use the latest available API version. To require a specific API version to use for your requests, include the api_versions setting in your profile.

Default value: None. (Latest API version is used by the SDK.)

Valid values: This is a nested setting that's followed by one or more indented lines that each identify one Amazon service and the API version to use. See the documentation for the Amazon service to understand which API versions are available.

The example sets a specific API version for two Amazon services in the config file. These API versions are used only for commands that run under the profile that contains these settings. Commands for any other service use the latest version of that service's API.

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

ca_bundle - shared Amazon config file setting, AWS_CA_BUNDLE - environment variable

Specifies the path to a custom certificate bundle (a file with a . pem extension) to use when establishing SSL/TLS connections.

Default value: none

Valid values: Specify either the full path or a base file name. If there is a base file name, the system attempts to find the program within folders specified by the PATH environment variable.

Example of setting this value in the config file:

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

Due to differences in how operating systems handle paths and escaping of path characters, the following is an example of setting this value in the config file on Windows:

```
[default]
ca_bundle = C:\\Users\\username\\.aws\\aws-custom-bundle.pem
```

Linux/macOS example of setting environment variables via command line:

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

Windows example of setting environment variables via command line:

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

output - shared Amazon config file setting

Specifies how results are formatted in the Amazon CLI and other Amazon SDKs and tools.

Default value: json

Valid values:

- json The output is formatted as a JSON string.
- yaml The output is formatted as a YAML string.
- yaml-stream The output is streamed and formatted as a YAML string. Streaming allows for faster handling of large data types.

• <u>text</u> – The output is formatted as multiple lines of tab-separated string values. This can be useful to pass the output to a text processor, like grep, sed, or awk.

• <u>table</u> – The output is formatted as a table using the characters +|- to form the cell borders. It typically presents the information in a "human-friendly" format that is much easier to read than the others, but not as programmatically useful.

parameter_validation - shared Amazon config file setting

Specifies whether the SDK or tool attempts to validate command line parameters before sending them to the Amazon service endpoint.

Default value: true

Valid values:

- true The default. The SDK or tool performs client-side validation of command line
 parameters. This helps the SDK or tool confirm that parameters are valid, and catches some
 errors. The SDK or tool can reject requests that aren't valid before sending requests to the
 Amazon service endpoint.
- **false** The SDK or tool doesn't validate command line parameters before sending them to the Amazon service endpoint. The Amazon service endpoint is responsible for validating all requests and rejecting requests that aren't valid.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Parti	api_versions not supported.
SDK for C++	Yes	
SDK for Go V2 (1.x)	Parti	api_versions and parameter_validation not supported.

SDK	Sı	Notes or more information
SDK for Go 1.x (V1)	Parti	api_versions and parameter_validation not supported. To use shared config file settings, you must turn on loading from the config file; see <u>Sessions</u> .
SDK for Java 2.x	No	
SDK for Java 1.x	No	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	Yes	
SDK for Kotlin	No	
SDK for .NET 4.x	No	
SDK for .NET 3.x	No	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	No	
SDK for Swift	No	
Tools for PowerShell V5	No	
Tools for PowerShell V4	No	

Host prefix injection



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Host prefix injection is a feature where Amazon SDKs automatically prepend a prefix to the hostname of service endpoints for certain API operations. This prefix can be either a static string or a dynamic value that includes data from your request parameters.

For example, when using Amazon Simple Storage Service to perform actions on Amazon S3 objects or buckets, the SDK replaces your bucket name and Amazon Web Services account ID in the final API endpoint.

While this behavior is required for normal Amazon service endpoints, it can cause problems when using custom endpoints such as VPC endpoints or local testing tools. In these cases, you might need to disable host prefix injection.

Configure this functionality by using the following:

disable_host_prefix_injection - shared Amazon config file setting, AWS_DISABLE_HOST_PREFIX_INJECTION - environment variable, aws.disableHostPrefixInjection - JVM system property: Java/Kotlin only

This setting controls whether the SDK or tool will modify the endpoint hostname by prepending a host prefix as defined in your SDK's client object or variable.

Default value: false

Valid values:

- true Disable host prefix injection. The SDK will not modify the endpoint hostname.
- false Enable host prefix injection. The SDK will prepend the host prefix to the endpoint hostname.

Example of setting this value in the config file:

```
[default]
disable_host_prefix_injection = true
```

Linux/macOS example of setting environment variables via command line:

```
export AWS_DISABLE_HOST_PREFIX_INJECTION=true
```

Windows example of setting environment variables via command line:

```
setx AWS_DISABLE_HOST_PREFIX_INJECTION true
```

Examples of host prefix injection

The following table of examples show how SDKs modify the final endpoint when host prefix injection is enabled and disabled.

- Host prefix: The template of the host prefix property string set on the SDK's client object or variable in code.
- Inputs: Additional inputs set on the SDK's client object or variable in code.
- Client endpoint: The client's derived endpoint.
- Setting value: Resolved value for the previous setting.
- Resulting endpoint: The resulting endpoint the SDK client uses to make the API call.

Host prefix	Inputs	Client endpoint	Setting value	Resulting endpoint
"data."	8	"https:// service.u s-west-2. amazonaws .com"	false	"https:// data.service.us- west-2.amaz onaws.com"
"{Bucket}- {AccountId}."	Bucket: "amzn- s3-demo-buck et1", AccountId	"https:// service.u s-west-2.	false	"https://amzn- s3-demo-bucke t1-123456 789012.se

Host prefix	Inputs	Client endpoint	Setting value	Resulting endpoint
	:"1234567 89012"	amazonaws .com"		rvice.us- west-2.am azonaws.com"
"data."	0	"https:// override. us-west-2 .amazonaw s.com" (as an override endpoint)	true	"https:// override. us-west-2 .amazonaw s.com"

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	No	Setting not supported, but can be configured in code on the client using: enableHostPrefixInjection .
SDK for Go V2 (1.x)	No	Can be disabled <u>using middleware</u> .
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	No	Setting not supported, but can be configured in code on the client using: SdkAdvancedClientOption.DIS ABLE_HOST_PREFIX_INJECTION .
SDK for Java 1.x	No	Setting not supported, but can be configured in code on the client using: withDisableHostPrefixInjection .

SDK	Sı	Notes or more information
SDK for JavaScript 3.x	No	Setting not supported, but can be configured in code on the client using: disableHostPrefix .
SDK for JavaScript 2.x	No	Setting not supported, but can be configured in code on the client using: hostPrefixEnabled .
SDK for Kotlin	No	
SDK for .NET 4.x	No	Setting not supported, but can be configured in code on the client using: DisableHostPrefixInjection .
SDK for .NET 3.x	No	Setting not supported, but can be configured in code on the client using: DisableHostPrefixInjection .
SDK for PHP 3.x	No	Setting not supported, but can be configured in code on the client using: disable_host_prefix_injection .
SDK for Python (Boto3)	Yes	Can be configured in code on the client using: inject_ho st_prefix .
SDK for Ruby 3.x	No	Setting not supported, but can be configured in code on the client using: disable_host_prefix_injection .
SDK for Rust	No	
SDK for Swift	No	
Tools for PowerShell V5	No	Setting not supported, but can be included in specific cmdlets using parameter -ClientConfig @{Disable HostPrefixInjection = \$true} .
Tools for PowerShell V4	No	Setting not supported, but can be included in specific cmdlets using parameter -ClientConfig @{Disable HostPrefixInjection = \$true} .

IMDS client



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

SDKs implement an Instance Metadata Service Version 2 (IMDSv2) client using session-oriented requests. For more information on IMDSv2, see Use IMDSv2 in the Amazon EC2 User Guide. The IMDS client is configurable via a client configuration object available in the SDK code base.

Configure this functionality by using the following:

retries - client configuration object member

The number of additional retry attempts for any failed request.

Default value: 3

Valid values: Number greater than 0.

port - client configuration object member

The port for the endpoint.

Default value: 80

Valid values: Number.

token_ttl - client configuration object member

The TTL of the token.

Default value: 21,600 seconds (6 hours, the maximum time allotted).

Valid values: Number.

endpoint - client configuration object member

The endpoint of IMDS.

IMDS client 132

Default value: If endpoint_mode equals IPv4, then default endpoint is

http://169.254.169.254. If endpoint_mode equals IPv6, then default endpoint is

http://[fd00:ec2::254].

Valid values: Valid URI.

The following options are supported by most SDKs. See your specific SDK code base for details.

endpoint_mode - client configuration object member

The endpoint mode of IMDS.

Default value: IPv4

Valid values: IPv4, IPv6

http_open_timeout - client configuration object member (name may vary)

The number of seconds to wait for the connection to open.

Default value: 1 second.

Valid values: Number greater than 0.

http_read_timeout - client configuration object member (name may vary)

The number of seconds for one chunk of data to be read.

Default value: 1 second.

Valid values: Number greater than 0.

http_debug_output - client configuration object member (name may vary)

Sets an output stream for debugging.

Default value: None.

Valid values: A valid I/O stream, like STDOUT.

backoff - client configuration object member (name may vary)

The number of seconds to sleep in between retries or a customer provided backoff function to call. This overrides the default exponential backoff strategy.

IMDS client 133

Default value: Varies by SDK.

Valid values: Varies by SDK. Can be either a numeric value or a call out to a custom function.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Si Notes or more information
Amazon CLI v2	Yes
SDK for C++	No
SDK for Go V2 (1.x)	Yes
SDK for Go 1.x (V1)	Yes
SDK for Java 2.x	Yes
SDK for Java 1.x	Yes
SDK for JavaScript 3.x	Yes
SDK for JavaScript 2.x	Yes
SDK for Kotlin	No
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes

IMDS client 134

SDK	Sı	Notes or more information
SDK for Swift	Yes	
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	

Retry behavior



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Retry behavior includes settings regarding how the SDKs attempt to recover from failures resulting from requests made to Amazon Web Services services.

Configure this functionality by using the following:

retry_mode - shared Amazon config file setting, AWS_RETRY_MODE - environment variable, aws.retryMode - JVM system property: Java/Kotlin only

Specifies how the SDK or developer tool attempts retries.

Default value: This value is specific to your SDK. Check your specific SDK guide or your SDK's code base for its default retry_mode.

Valid values:

- standard (Recommended) The recommended set of retry rules across Amazon SDKs. This mode includes a standard set of errors that are retried, and automatically adjusts the number of retries to maximize availability and stability. This mode is safe for use in multi-tenant applications. The default maximum number of attempts with this mode is three, unless max_attempts is explicitly configured.
- adaptive A retry mode, appropriate only for specialized use-cases, that includes the functionality of standard mode as well as automatic client-side rate limiting. This retry mode

is not recommended for multi-tenant applications, unless you take care to isolate application tenants. See Choosing between standard and adaptive retry modes for more information. This mode is experimental and it might change behavior in the future.

 legacy – (Not Recommended) Specific to your SDK (check your specific SDK guide or your SDK's code base).

max_attempts - shared Amazon config file setting, AWS_MAX_ATTEMPTS - environment variable, aws.maxAttempts - JVM system property: Java/Kotlin only

Specifies the maximum number attempts to make on a request.

Default value: If this value is not specified, its default depends on the value of the retry mode setting:

- If retry_mode is legacy Uses a default value specific to your SDK (check your specific SDK) guide or your SDK's code base for max_attempts default).
- If retry_mode is standard Makes three attempts.
- If retry_mode is adaptive Makes three attempts.

Valid values: Number greater than 0.

Choosing between standard and adaptive retry modes

We recommend you use the standard retry mode unless you are certain that your usage is better suited for adaptive.



Note

The adaptive mode assumes that you are pooling clients based on the scope at which the backend service may throttle requests. If you don't do this, throttles in one resource could delay requests for an unrelated resource if you are using the same client for both resources.

Standard	Adaptive
Application use-cases: All.	Application use-cases:
	1. Not sensitive to latency.

Standard	Adaptive
	Client only accesses a single resource, or, you are providing logic to pool your clients separately by the service resource that is being accessed.
Supports circuit-breaking to prevent the SDK from retrying during outages.	Supports circuit-breaking to prevent the SDK from retrying during outages.
Uses jittered exponential backoff in the event of failures.	Uses dynamic backoff durations to attempt to minimize the number of failed requests, in exchange for the potential for increased latency.
Never delays the first request attempt, only the retries.	Can throttle or delay the initial request attempt.

If you choose to use adaptive mode, your application must construct clients that are designed around each resource that might be throttled. A resource, in this case, is finer-tuned than just thinking of each Amazon Web Services service. Amazon Web Services services can have additional dimensions that they use to throttle requests. Let's use the Amazon DynamoDB service as an example. DynamoDB uses Amazon Web Services Region plus the table being accessed to throttle requests. This means that one table that your code is accessing might be throttled more than others. If your code used the same client to access all the tables, and requests to one of those tables is throttled, then adaptive retry mode will reduce the request rate for all tables. Your code should be designed to have one client per Region-and-table pair. If you experience unexpected latency when using adaptive mode, see the specific Amazon documentation guide for the service you are using.

Retry mode implementation details

The Amazon SDKs make use of <u>token buckets</u> to decide whether a request should be retried and (in the case of the adaptive retry mode) how quickly requests should be sent. Two token buckets are used by the SDK: a retry token bucket and a request rate token bucket.

• The retry token bucket is used to determine whether the SDK should temporarily disable retries in order to protect the upstream and downstream services during outages. Tokens are acquired

from the bucket before retries are attempted, and tokens are returned to the bucket when requests succeed. If the bucket is empty when a retry is attempted, the SDK will not retry the request.

The request rate token bucket is used only in the adaptive retry mode to determine the rate at
which to send requests. Tokens are acquired from the bucket before a request is sent, and tokens
are returned to the bucket at a dynamically-determined rate based on throttling responses
returned by the service.

Following is the high-level pseudocode for both the standard and adaptive retry modes:

```
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)
  }
}
```

Following are more details about the components used in the pseudocode:

GetSendToken:

This step is only used in adaptive retry mode. This step acquires a token from the request rate token bucket. If a token is not available, it will wait for one to become available. Your SDK might have configuration options available to fail the request instead of wait. Tokens in the bucket are refilled at a rate that is determined dynamically, based on the number of throttling responses received by the client.

SendHTTPRequest:

This step sends the request to Amazon. Most Amazon SDKs use an HTTP library that uses connection pools to reuse an existing connection when making an HTTP request. Generally, connections are reused if a request failed due to throttling errors but not if a request fails due to a transient error.

RequestBookkeeping:

Tokens are added to the token bucket if the request is successful. For adaptive retry mode only, the fill rate of the request rate token bucket is updated based on the type of response received.

Retryable:

This step determines whether a response can be retried based on the following:

- The HTTP status code.
- The error code returned from the service.
- Connection errors, defined as any error received by the SDK in which an HTTP response from the service is not received.

Transient errors (HTTP status codes 400, 408, 500, 502, 503, and 504) and throttling errors (HTTP status codes 400, 403, 429, 502, 503, and 509) can all potentially be retried. SDK retry behavior is determined in combination with error codes or other data from the service.

MAX_ATTEMPTS:

The default number of maximum attempts is set by the retry_mode setting, unless overridden by the max_attempts setting.

HasRetryQuota

This step acquires a token from the retry token bucket. If the retry token bucket is empty, the request will not be retried.

ExponentialBackoff

For an error that can be retried, the retry delay is calculated using truncated exponential backoff. The SDKs use truncated binary exponential backoff with jitter. The following algorithm shows how the amount of time to sleep, in seconds, is defined for a response for request i:

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

In the preceding algorithm, the following values apply:

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

r = 2

MAX_BACKOFF = 20 seconds for most SDKs. See your specific SDK guide or source code for confirmation.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	Yes	
SDK for Java 1.x	Yes	JVM system properties: use com.amazonaws.sdk. maxAttempts instead of aws.maxAttempts; use com.amazonaws.sdk.retryMode instead of aws.retryMode.
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	No	Supports a maximum number of retries, exponential backoff with jitter, and an option for a custom method for retry backoff.
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	

Retry behavior 140

SDK	Si Notes or more information
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes
SDK for Swift	Yes
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

Request compression



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Amazon SDKs and tools can automatically compress payloads when sending requests to Amazon Web Services services that support receiving compressed payloads. Compressing the payload on the client prior to sending it to a service may reduce the overall number of requests and bandwidth required to send data to the service, as well as reduce unsuccessful requests due to service limitations on the payload size. For compression, the SDK or tool selects an encoding algorithm that is supported by both the service and the SDK. However, the current list of possible encodings consists only of gzip, but it may expand in the future.

Request compression can be especially useful if your application is using Amazon CloudWatch. CloudWatch is a monitoring and observability service that collects monitoring and operational data in the form of logs, metrics, and events. One example of a service operation that supports compression is CloudWatch's PutMetricDataAPI method.

Request compression 141

Configure this functionality by using the following:

disable_request_compression - shared Amazon config file
setting, AWS_DISABLE_REQUEST_COMPRESSION - environment variable,
aws.disableRequestCompression - JVM system property: Java/Kotlin only

Turns on or off whether the SDK or tool will compress a payload prior to sending a request.

Default value: false

Valid values:

- true Turn off request compression.
- **false** Use request compression when possible.

request_min_compression_size_bytes - shared Amazon config file
setting, AWS_REQUEST_MIN_COMPRESSION_SIZE_BYTES - environment variable,
aws.requestMinCompressionSizeBytes - JVM system property: Java/Kotlin only

Sets the minimum size in bytes of the request body that the SDK or tool should compress. Small payloads may become longer when compressed, thus, there is a lower limit where it makes sense to perform compression. This value is inclusive, a request size greater than or equal to the value is compressed.

Default value: 10240 bytes

Valid values: Integer value between 0 and 10485760 bytes inclusive.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Sı	Notes or more information
Amazon CLI v2	Yes	
SDK for C++	Yes	
SDK for Go V2 (1.x)	Yes	

Request compression 142

SDK	Si Notes or more information
SDK for Go 1.x (V1)	No
SDK for Java 2.x	Yes
SDK for Java 1.x	No
SDK for JavaScript 3.x	Yes
SDK for JavaScript 2.x	No
SDK for Kotlin	Yes
SDK for .NET 4.x	Yes
SDK for .NET 3.x	Yes
SDK for PHP 3.x	Yes
SDK for Python (Boto3)	Yes
SDK for Ruby 3.x	Yes
SDK for Rust	Yes
SDK for Swift	No
Tools for PowerShell V5	Yes
Tools for PowerShell V4	Yes

Service-specific endpoints



For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

Service-specific endpoint configuration provides the option to use an endpoint of your choosing for API requests and to have that choice persist. These settings provide flexibility to support local endpoints, VPC endpoints, and third-party local Amazon development environments. Different endpoints can be used for testing and production environments. You can specify an endpoint URL for individual Amazon Web Services services.

Configure this functionality by using the following:

endpoint_url - shared Amazon config file setting, AWS_ENDPOINT_URL - environment variable, aws.endpointUrl - JVM system property: Java/Kotlin only

When specified directly within a profile or as an environment variable, this setting specifies the endpoint that is used for all service requests. This endpoint is overridden by any configured service-specific endpoint.

You can also use this setting within a services section of a shared Amazon config file to set a custom endpoint for a specific service. For a list of all service identifier keys to use for subsections within the services section, see Identifiers for service-specific endpoints.

Default value: none

Valid values: A URL including the scheme and host for the endpoint. The URL can optionally contain a path component that contains one or more path segments.

AWS_ENDPOINT_URL_<SERVICE> - environment variable, aws.endpointUrl<ServiceName> - JVM system property: Java/Kotlin only

AWS_ENDPOINT_URL_<SERVICE>, where <SERVICE> is the Amazon Web Services service identifier, sets a custom endpoint for a specific service. For a list of all service-specific environment variables, see Identifiers for service-specific endpoints.

This service-specific endpoint overrides any global endpoint set in AWS_ENDPOINT_URL.

Default value: none

Valid values: A URL including the scheme and host for the endpoint. The URL can optionally contain a path component that contains one or more path segments.

ignore_configured_endpoint_urls - shared Amazon config file
setting, AWS_IGNORE_CONFIGURED_ENDPOINT_URLS - environment variable,
aws.ignoreConfiguredEndpointUrls - JVM system property: Java/Kotlin only

This setting is used to ignore all custom endpoints configurations.

Note that any explicit endpoint set in the code or on a service client itself is used regardless of this setting. For example, including the --endpoint-url command line parameter with an Amazon CLI command or passing an endpoint URL into a client constructor will always take effect.

Default value: false

Valid values:

- **true** The SDK or tool does not read any custom configuration options from the shared config file or from environment variables for setting an endpoint URL.
- **false** The SDK or tool uses any available user-provided endpoints from the shared config file or from environment variables.

Configure endpoints using environment variables

To route requests for all services to a custom endpoint URL, set the AWS_ENDPOINT_URL global environment variable.

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

To route requests for a specific Amazon Web Services service to a custom endpoint URL, use the AWS_ENDPOINT_URL_<SERVICE> environment variable. Amazon DynamoDB has a serviceId of DynamoDB. For this service, the endpoint URL environment variable is AWS_ENDPOINT_URL_DYNAMODB. This endpoint takes precedence over the global endpoint set in AWS_ENDPOINT_URL for this service.

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

As another example, Amazon Elastic Beanstalk has a serviceId of <u>Elastic Beanstalk</u>. The Amazon Web Services service identifier is based on the API model's serviceId by replacing all spaces with underscores and uppercasing all letters. To set the endpoint for this service, the corresponding environment variable is AWS_ENDPOINT_URL_ELASTIC_BEANSTALK. For a list of all service-specific environment variables, see <u>Identifiers</u> for service-specific endpoints.

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

Configure endpoints using the shared config file

In the shared config file, endpoint_url is used in different places for different functionality.

- endpoint_url specified directly within a profile makes that endpoint the global endpoint.
- endpoint_url nested under a service identifier key within a services section makes that endpoint apply to requests made only to that service. For details on defining a services section in your shared config file, see Format of the config file.

The following example uses a services definition to configure a service-specific endpoint URL to be used for Amazon S3 and a custom global endpoint to be used for all other services:

```
[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
```

A single profile can configure endpoints for multiple services. This example shows how to set the service-specific endpoint URLs for Amazon S3 and Amazon Elastic Beanstalk in the same profile. Amazon Elastic Beanstalk has a serviceId of Elastic Beanstalk. The Amazon Web Services service identifier is based on the API model's serviceId by replacing all spaces with underscores and lowercasing all letters. Thus, the service identifier key becomes elastic_beanstalk and settings for this service begin on the line elastic_beanstalk = . For a list of all service identifier keys to use in the services section, see Identifiers for service-specific endpoints.

```
[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
```

The service configuration section can be used from multiple profiles. For example, two profiles can use the same services definition while altering other profile properties:

```
[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
```

Configure endpoints in profiles using role-based credentials

If your profile has role-based credentials configured through a source_profile parameter for IAM assume role functionality, the SDK only uses service configurations for the specified profile. It does not use profiles that are role chained to it. For example, using the following shared config file:

```
[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
```

If you use profile B and make a call in your code to Amazon EC2, the endpoint resolves as https://profile-b-ec2-endpoint.aws. If your code makes a request to any other service, the endpoint resolution will not follow any custom logic. The endpoint does not resolve to the global endpoint defined in profile A. For a global endpoint to take effect for profile B, you would need to set endpoint_url directly within profile B. For more information on the source_profile setting, see Assume role credential provider.

Precedence of settings

The settings for this feature can be used at the same time but only one value will take priority per service. For API calls made to a given Amazon Web Services service, the following order is used to select a value:

- Any explicit setting set in the code or on a service client itself takes precedence over anything else.
 - For the Amazon CLI, this is the value provided by the --endpoint-url command line parameter. For an SDK, explicit assignments can take the form of a parameter that you set when you instantiate an Amazon Web Services service client or configuration object.
- 2. The value provided by a service-specific environment variable such as AWS_ENDPOINT_URL_DYNAMODB.
- 3. The value provided by the AWS_ENDPOINT_URL global endpoint environment variable.
- 4. The value provided by the endpoint_url setting nested under a service identifier key within a services section of the shared config file.
- 5. The value provided by the endpoint_url setting specified directly within a profile of the shared config file.
- 6. Any default endpoint URL for the respective Amazon Web Services service is used last.

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Si Notes or more information
Amazon CLI v2	Yes
SDK for C++	No
SDK for Go V2 (1.x)	Yes
SDK for Go 1.x (V1)	No
SDK for Java 2.x	Yes

SDK	Sı	Notes or more information
SDK for Java 1.x	No	
SDK for JavaScript 3.x	Yes	
SDK for JavaScript 2.x	No	
SDK for Kotlin	Yes	
SDK for .NET 4.x	Yes	
SDK for .NET 3.x	Yes	
SDK for PHP 3.x	Yes	
SDK for Python (Boto3)	Yes	
SDK for Ruby 3.x	Yes	
SDK for Rust	Yes	
SDK for Swift	Yes	
Tools for PowerShell V5	Yes	
Tools for PowerShell V4	Yes	

Identifiers for service-specific endpoints

For information on how and where to use the identifiers in the following table, see <u>Service-specific</u> endpoints.

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
AccessAnalyzer	ac AWS_ENDPOINT_URL_ACCESSANALYZER 1)
Account	a AWS_ENDPOINT_URL_ACCOUNT
ACM	a AWS_ENDPOINT_URL_ACM
ACM PCA	a AWS_ENDPOINT_URL_ACM_PCA
Alexa For Business	a: AWS_ENDPOINT_URL_ALEXA_FOR_BUSINESS _!
amp	ar AWS_ENDPOINT_URL_AMP
Amplify	ar AWS_ENDPOINT_URL_AMPLIFY
AmplifyBackend	ar AWS_ENDPOINT_URL_AMPLIFYBACKEND
AmplifyUIBuilder	ar AWS_ENDPOINT_URL_AMPLIFYUIBUILDER
API Gateway	ar AWS_ENDPOINT_URL_API_GATEWAY ar
ApiGatewayManageme ntApi	a; AWS_ENDPOINT_URL_APIGATEWAYMANAGEMENTAPI yr n1

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Ai ct fil</service>
ApiGatewayV2	a; AWS_ENDPOINT_URL_APIGATEWAYV2 y
AppConfig	a; AWS_ENDPOINT_URL_APPCONFIG
AppConfigData	a; AWS_ENDPOINT_URL_APPCONFIGDATA d;
AppFabric	a; AWS_ENDPOINT_URL_APPFABRIC
Appflow	a; AWS_ENDPOINT_URL_APPFLOW
AppIntegrations	a; AWS_ENDPOINT_URL_APPINTEGRATIONS at
Application Auto Scaling	a; AWS_ENDPOINT_URL_APPLICATION_AUTO_SCALING or ca
Application Insights	a; AWS_ENDPOINT_URL_APPLICATION_INSIGHTS or t:
ApplicationCostPro filer	a; AWS_ENDPOINT_URL_APPLICATIONCOSTPROFILER or f:

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
App Mesh	a; AWS_ENDPOINT_URL_APP_MESH
AppRunner	a; AWS_ENDPOINT_URL_APPRUNNER
AppStream	a¡ AWS_ENDPOINT_URL_APPSTREAM
AppSync	a; AWS_ENDPOINT_URL_APPSYNC
ARC Zonal Shift	a: AWS_ENDPOINT_URL_ARC_ZONAL_SHIFT _:
Artifact	a: AWS_ENDPOINT_URL_ARTIFACT
Athena	at AWS_ENDPOINT_URL_ATHENA
AuditManager	at AWS_ENDPOINT_URL_AUDITMANAGER ge
Auto Scaling	at AWS_ENDPOINT_URL_AUTO_SCALING
Auto Scaling Plans	at AWS_ENDPOINT_URL_AUTO_SCALING_PLANS in
b2bi	b2 AWS_ENDPOINT_URL_B2BI

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
Backup	b: AWS_ENDPOINT_URL_BACKUP
Backup Gateway	b; AWS_ENDPOINT_URL_BACKUP_GATEWAY
BackupStorage	b: AWS_ENDPOINT_URL_BACKUPSTORAGE r:
Batch	b: AWS_ENDPOINT_URL_BATCH
BCM Data Exports	<pre>bc AWS_ENDPOINT_URL_BCM_DATA_EXPORTS e;</pre>
Bedrock	b AWS_ENDPOINT_URL_BEDROCK
Bedrock Agent	be AWS_ENDPOINT_URL_BEDROCK_AGENT ge
Bedrock Agent Runtime	<pre>be AWS_ENDPOINT_URL_BEDROCK_AGENT_RUNTIME ge ir</pre>
Bedrock Runtime	be AWS_ENDPOINT_URL_BEDROCK_RUNTIME ur
billingconductor	b: AWS_ENDPOINT_URL_BILLINGCONDUCTOR
Braket	b: AWS_ENDPOINT_URL_BRAKET

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Budgets	bi AWS_ENDPOINT_URL_BUDGETS
Cost Explorer	cc AWS_ENDPOINT_URL_COST_EXPLORER o:
chatbot	cl AWS_ENDPOINT_URL_CHATBOT
Chime	cl AWS_ENDPOINT_URL_CHIME
Chime SDK Identity	cl AWS_ENDPOINT_URL_CHIME_SDK_IDENTITY _:
Chime SDK Media Pipelines	cl AWS_ENDPOINT_URL_CHIME_SDK_MEDIA_PIPELINES _r pt
Chime SDK Meetings	cl AWS_ENDPOINT_URL_CHIME_SDK_MEETINGS _r
Chime SDK Messaging	cl AWS_ENDPOINT_URL_CHIME_SDK_MESSAGING _r g
Chime SDK Voice	<pre>c! AWS_ENDPOINT_URL_CHIME_SDK_VOICE _'</pre>

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
CleanRooms	c: AWS_ENDPOINT_URL_CLEANROOMS s
CleanRoomsML	c: AWS_ENDPOINT_URL_CLEANROOMSML
Cloud9	c: AWS_ENDPOINT_URL_CLOUD9
CloudControl	c: AWS_ENDPOINT_URL_CLOUDCONTROL
CloudDirectory	c: AWS_ENDPOINT_URL_CLOUDDIRECTORY
CloudFormation	c: AWS_ENDPOINT_URL_CLOUDFORMATION
CloudFront	c: AWS_ENDPOINT_URL_CLOUDFRONT
CloudFront KeyValueS tore	c: AWS_ENDPOINT_URL_CLOUDFRONT_KEYVALUESTORE t_ e:
CloudHSM	c: AWS_ENDPOINT_URL_CLOUDHSM
CloudHSM V2	c: AWS_ENDPOINT_URL_CLOUDHSM_V2

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
CloudSearch	c: AWS_ENDPOINT_URL_CLOUDSEARCH
CloudSearch Domain	c: AWS_ENDPOINT_URL_CLOUDSEARCH_DOMAIN
CloudTrail	c: AWS_ENDPOINT_URL_CLOUDTRAIL
CloudTrail Data	c: AWS_ENDPOINT_URL_CLOUDTRAIL_DATA 1_
CloudWatch	c: AWS_ENDPOINT_URL_CLOUDWATCH
codeartifact	cc AWS_ENDPOINT_URL_CODEARTIFACT
CodeBuild	cc AWS_ENDPOINT_URL_CODEBUILD
CodeCatalyst	<pre>cc AWS_ENDPOINT_URL_CODECATALYST y:</pre>
CodeCommit	cc AWS_ENDPOINT_URL_CODECOMMIT

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
CodeDeploy	cc AWS_ENDPOINT_URL_CODEDEPLOY y
CodeGuru Reviewer	cc AWS_ENDPOINT_URL_CODEGURU_REVIEWER
CodeGuru Security	<pre>cc AWS_ENDPOINT_URL_CODEGURU_SECURITY se</pre>
CodeGuruProfiler	cc AWS_ENDPOINT_URL_CODEGURUPROFILER
CodePipeline	cc AWS_ENDPOINT_URL_CODEPIPELINE
CodeStar	cc AWS_ENDPOINT_URL_CODESTAR
CodeStar connections	<pre>cc AWS_ENDPOINT_URL_CODESTAR_CONNECTIONS cc</pre>
codestar notificat ions	cc AWS_ENDPOINT_URL_CODESTAR_NOTIFICATIONS nc ic
Cognito Identity	<pre>c AWS_ENDPOINT_URL_COGNITO_IDENTITY d:</pre>

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Cognito Identity Provider	<pre>cc AWS_ENDPOINT_URL_COGNITO_IDENTITY_PROVIDER dc rc</pre>
Cognito Sync	cc AWS_ENDPOINT_URL_COGNITO_SYNC
Comprehend	cc AWS_ENDPOINT_URL_COMPREHEND d
ComprehendMedical	cc AWS_ENDPOINT_URL_COMPREHENDMEDICAL dr
Compute Optimizer	cc AWS_ENDPOINT_URL_COMPUTE_OPTIMIZER p1
Config Service	cc AWS_ENDPOINT_URL_CONFIG_SERVICE
Connect	cc AWS_ENDPOINT_URL_CONNECT
Connect Contact Lens	cc AWS_ENDPOINT_URL_CONNECT_CONTACT_LENS or ns
ConnectCampaigns	cc AWS_ENDPOINT_URL_CONNECTCAMPAIGNS m;

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al co fil</service>
ConnectCases	<pre>c(AWS_ENDPOINT_URL_CONNECTCASES s(</pre>
ConnectParticipant	cc AWS_ENDPOINT_URL_CONNECTPARTICIPANT r1
ControlTower	C: AWS_ENDPOINT_URL_CONTROLTOWER
Cost Optimization Hub	cc AWS_ENDPOINT_URL_COST_OPTIMIZATION_HUB m: hu
Cost and Usage Report Service	<pre>cc AWS_ENDPOINT_URL_COST_AND_USAGE_REPO us RT_SERVICE o: ce</pre>
Customer Profiles	cı AWS_ENDPOINT_URL_CUSTOMER_PROFILES p:
DataBrew	d: AWS_ENDPOINT_URL_DATABREW
DataExchange	da AWS_ENDPOINT_URL_DATAEXCHANGE

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
Data Pipeline	<pre>d: AWS_ENDPOINT_URL_DATA_PIPELINE 1:</pre>
DataSync	d: AWS_ENDPOINT_URL_DATASYNC
DataZone	d: AWS_ENDPOINT_URL_DATAZONE
DAX	d: AWS_ENDPOINT_URL_DAX
Detective	d: AWS_ENDPOINT_URL_DETECTIVE
Device Farm	d AWS_ENDPOINT_URL_DEVICE_FARM
DevOps Guru	d: AWS_ENDPOINT_URL_DEVOPS_GURU
Direct Connect	d: AWS_ENDPOINT_URL_DIRECT_CONNECT
Application Discovery Service	a; AWS_ENDPOINT_URL_APPLICATION_DISCOVE or RY_SERVICE e: c:
DLM	d: AWS_ENDPOINT_URL_DLM

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
Database Migration Service	d; AWS_ENDPOINT_URL_DATABASE_MIGRATION_ m: SERVICE!
DocDB	dc AWS_ENDPOINT_URL_DOCDB
DocDB Elastic	<pre>dc AWS_ENDPOINT_URL_DOCDB_ELASTIC s1</pre>
drs	d: AWS_ENDPOINT_URL_DRS
Directory Service	d: AWS_ENDPOINT_URL_DIRECTORY_SERVICE _:
DynamoDB	d: AWS_ENDPOINT_URL_DYNAMODB
DynamoDB Streams	d: AWS_ENDPOINT_URL_DYNAMODB_STREAMS
EBS	el AWS_ENDPOINT_URL_EBS
EC2	e AWS_ENDPOINT_URL_EC2
EC2 Instance Connect	ec AWS_ENDPOINT_URL_EC2_INSTANCE_CONNECT nc cd
ECR	e AWS_ENDPOINT_URL_ECR

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
ECR PUBLIC	e AWS_ENDPOINT_URL_ECR_PUBLIC
ECS	e AWS_ENDPOINT_URL_ECS
EFS	e AWS_ENDPOINT_URL_EFS
EKS	el AWS_ENDPOINT_URL_EKS
EKS Auth	el AWS_ENDPOINT_URL_EKS_AUTH
Elastic Inference	e: AWS_ENDPOINT_URL_ELASTIC_INFERENCE
ElastiCache	e: AWS_ENDPOINT_URL_ELASTICACHE
Elastic Beanstalk	e: AWS_ENDPOINT_URL_ELASTIC_BEANSTALK e;
Elastic Transcoder	e: AWS_ENDPOINT_URL_ELASTIC_TRANSCODER r;
Elastic Load Balancing	e: AWS_ENDPOINT_URL_ELASTIC_LOAD_BALANCING o; c:

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
Elastic Load Balancing v2	e: AWS_ENDPOINT_URL_ELASTIC_LOAD_BALANCING_V2 o; c:
EMR	er AWS_ENDPOINT_URL_EMR
EMR containers	er AWS_ENDPOINT_URL_EMR_CONTAINERS in
EMR Serverless	er AWS_ENDPOINT_URL_EMR_SERVERLESS r:
EntityResolution	er AWS_ENDPOINT_URL_ENTITYRESOLUTION ol
Elasticsearch Service	e: AWS_ENDPOINT_URL_ELASTICSEARCH_SERVICE a: i(
EventBridge	e AWS_ENDPOINT_URL_EVENTBRIDGE
Evidently	e AWS_ENDPOINT_URL_EVIDENTLY
finspace	f: AWS_ENDPOINT_URL_FINSPACE
finspace data	f: AWS_ENDPOINT_URL_FINSPACE_DATA d:

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
Firehose	f: AWS_ENDPOINT_URL_FIREHOSE
fis	f: AWS_ENDPOINT_URL_FIS
FMS	fr AWS_ENDPOINT_URL_FMS
forecast	fc AWS_ENDPOINT_URL_FORECAST
forecastquery	fc AWS_ENDPOINT_URL_FORECASTQUERY
FraudDetector	f: AWS_ENDPOINT_URL_FRAUDDETECTOR
FreeTier	f: AWS_ENDPOINT_URL_FREETIER
FSx	f: AWS_ENDPOINT_URL_FSX
GameLift	ga AWS_ENDPOINT_URL_GAMELIFT
Glacier	g: AWS_ENDPOINT_URL_GLACIER
Global Accelerator	g: AWS_ENDPOINT_URL_GLOBAL_ACCELERATOR
Glue	g: AWS_ENDPOINT_URL_GLUE
grafana	g: AWS_ENDPOINT_URL_GRAFANA

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Greengrass	g: AWS_ENDPOINT_URL_GREENGRASS s
GreengrassV2	g: AWS_ENDPOINT_URL_GREENGRASSV2
GroundStation	g: AWS_ENDPOINT_URL_GROUNDSTATION t:
GuardDuty	gi AWS_ENDPOINT_URL_GUARDDUTY
Health	h AWS_ENDPOINT_URL_HEALTH
HealthLake	he AWS_ENDPOINT_URL_HEALTHLAKE e
Honeycode	hc AWS_ENDPOINT_URL_HONEYCODE
IAM	i; AWS_ENDPOINT_URL_IAM
identitystore	ic AWS_ENDPOINT_URL_IDENTITYSTORE
imagebuilder	ir AWS_ENDPOINT_URL_IMAGEBUILDER de

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
ImportExport	<pre>ir AWS_ENDPOINT_URL_IMPORTEXPORT o:</pre>
Inspector	in AWS_ENDPOINT_URL_INSPECTOR
Inspector Scan	ir AWS_ENDPOINT_URL_INSPECTOR_SCAN
Inspector2	ir AWS_ENDPOINT_URL_INSPECTOR2 2
InternetMonitor	ir AWS_ENDPOINT_URL_INTERNETMONITOR
IoT	ic AWS_ENDPOINT_URL_IOT
IoT Data Plane	ic AWS_ENDPOINT_URL_IOT_DATA_PLANE p:
IoT Jobs Data Plane	ic AWS_ENDPOINT_URL_IOT_JOBS_DATA_PLANE date
IoT 1Click Devices Service	<pre>ic AWS_ENDPOINT_URL_IOT_1CLICK_DEVICES_ k_ SERVICE _!</pre>

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
IoT 1Click Projects	ic AWS_ENDPOINT_URL_IOT_1CLICK_PROJECTS k_ s
IoTAnalytics	ic AWS_ENDPOINT_URL_IOTANALYTICS
IotDeviceAdvisor	ic AWS_ENDPOINT_URL_IOTDEVICEADVISOR
IoT Events	ic AWS_ENDPOINT_URL_IOT_EVENTS s
IoT Events Data	ic AWS_ENDPOINT_URL_IOT_EVENTS_DATA s_
IoTFleetHub	ic AWS_ENDPOINT_URL_IOTFLEETHUB
IoTFleetWise	ic AWS_ENDPOINT_URL_IOTFLEETWISE is
IoTSecureTunneling	ic AWS_ENDPOINT_URL_IOTSECURETUNNELING
IoTSiteWise	ic AWS_ENDPOINT_URL_IOTSITEWISE

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
IoTThingsGraph	ic AWS_ENDPOINT_URL_IOTTHINGSGRAPH g:
IoTTwinMaker	ic AWS_ENDPOINT_URL_IOTTWINMAKER
IoT Wireless	ic AWS_ENDPOINT_URL_IOT_WIRELESS es
ivs	iv AWS_ENDPOINT_URL_IVS
IVS RealTime	iv AWS_ENDPOINT_URL_IVS_REALTIME ir
ivschat	iv AWS_ENDPOINT_URL_IVSCHAT
Kafka	k; AWS_ENDPOINT_URL_KAFKA
KafkaConnect	k; AWS_ENDPOINT_URL_KAFKACONNECT
kendra	k← AWS_ENDPOINT_URL_KENDRA
Kendra Ranking	k AWS_ENDPOINT_URL_KENDRA_RANKING
Keyspaces	k AWS_ENDPOINT_URL_KEYSPACES

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
Kinesis	k: AWS_ENDPOINT_URL_KINESIS
Kinesis Video Archived Media	k: AWS_ENDPOINT_URL_KINESIS_VIDEO_ARCHI i VED_MEDIA i a
Kinesis Video Media	k: AWS_ENDPOINT_URL_KINESIS_VIDEO_MEDIA ic a
Kinesis Video Signaling	k: AWS_ENDPOINT_URL_KINESIS_VIDEO_SIGNALING ic a:
Kinesis Video WebRTC Storage	k: AWS_ENDPOINT_URL_KINESIS_VIDEO_WEBRT ic C_STORAGE tc e
Kinesis Analytics	k: AWS_ENDPOINT_URL_KINESIS_ANALYTICS
Kinesis Analytics V2	k: AWS_ENDPOINT_URL_KINESIS_ANALYTICS_V2 n; v2

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Kinesis Video	k: AWS_ENDPOINT_URL_KINESIS_VIDEO
KMS	kr AWS_ENDPOINT_URL_KMS
LakeFormation	1: AWS_ENDPOINT_URL_LAKEFORMATION t:
Lambda	1; AWS_ENDPOINT_URL_LAMBDA
Launch Wizard	1; AWS_ENDPOINT_URL_LAUNCH_WIZARD
Lex Model Building Service	<pre>1 AWS_ENDPOINT_URL_LEX_MODEL_BUILDING! SERVICE _!</pre>
Lex Runtime Service	<pre>1 AWS_ENDPOINT_URL_LEX_RUNTIME_SERVICE m(e</pre>
Lex Models V2	1 AWS_ENDPOINT_URL_LEX_MODELS_V2 s_
Lex Runtime V2	1: AWS_ENDPOINT_URL_LEX_RUNTIME_V2
License Manager	1: AWS_ENDPOINT_URL_LICENSE_MANAGER ar

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
License Manager Linux Subscriptions	1: AWS_ENDPOINT_URL_LICENSE_MANAGER_LIN ar UX_SUBSCRIPTIONS nr r:
License Manager User Subscriptions	1: AWS_ENDPOINT_URL_LICENSE_MANAGER_USE ar R_SUBSCRIPTIONS e: ir
Lightsail	1: AWS_ENDPOINT_URL_LIGHTSAIL
Location	1 AWS_ENDPOINT_URL_LOCATION
CloudWatch Logs	c: AWS_ENDPOINT_URL_CLOUDWATCH_LOGS h_
LookoutEquipment	1 AWS_ENDPOINT_URL_LOOKOUTEQUIPMENT u:
LookoutMetrics	1 AWS_ENDPOINT_URL_LOOKOUTMETRICS
LookoutVision	1 AWS_ENDPOINT_URL_LOOKOUTVISION s:
m2	m2 AWS_ENDPOINT_URL_M2

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Machine Learning	machine_Learning
Macie2	m: AWS_ENDPOINT_URL_MACIE2
ManagedBlockchain	ma AWS_ENDPOINT_URL_MANAGEDBLOCKCHAIN
ManagedBlockchain Query	makaws_endpoint_url_managedblockchain_query or qr
Marketplace Agreement	mac AWS_ENDPOINT_URL_MARKETPLACE_AGREEMENT ce ei
Marketplace Catalog	maketplace_catalog
Marketplace Deploymen t	ma AWS_ENDPOINT_URL_MARKETPLACE_DEPLOYMENT Comme
Marketplace Entitleme nt Service	mac AWS_ENDPOINT_URL_MARKETPLACE_ENTITLE complete MENT_SERVICE er v:

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Marketplace Commerce Analytics	m: AWS_ENDPOINT_URL_MARKETPLACE_COMMERC c: E_ANALYTICS c: i:
MediaConnect	me AWS_ENDPOINT_URL_MEDIACONNECT
MediaConvert	mc AWS_ENDPOINT_URL_MEDIACONVERT e:
MediaLive	m@ AWS_ENDPOINT_URL_MEDIALIVE
MediaPackage	ma AWS_ENDPOINT_URL_MEDIAPACKAGE aq
MediaPackage Vod	me AWS_ENDPOINT_URL_MEDIAPACKAGE_VOD
MediaPackageV2	me AWS_ENDPOINT_URL_MEDIAPACKAGEV2
MediaStore	me AWS_ENDPOINT_URL_MEDIASTORE e
MediaStore Data	mc AWS_ENDPOINT_URL_MEDIASTORE_DATA e_

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
MediaTailor	me AWS_ENDPOINT_URL_MEDIATAILOR o:
Medical Imaging	ma AWS_ENDPOINT_URL_MEDICAL_IMAGING ma
MemoryDB	me AWS_ENDPOINT_URL_MEMORYDB
Marketplace Metering	ma AWS_ENDPOINT_URL_MARKETPLACE_METERING Company note not
Migration Hub	m: AWS_ENDPOINT_URL_MIGRATION_HUB _
mgn	mc AWS_ENDPOINT_URL_MGN
Migration Hub Refactor Spaces	m: AWS_ENDPOINT_URL_MIGRATION_HUB_REFAC _I TOR_SPACES c1 e:
MigrationHub Config	m: AWS_ENDPOINT_URL_MIGRATIONHUB_CONFIG hu g

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
MigrationHubOrches trator	m: AWS_ENDPOINT_URL_MIGRATIONHUBORCHESTRATOR hu t:
MigrationHubStrategy	m: AWS_ENDPOINT_URL_MIGRATIONHUBSTRATEGY hu g:
Mobile	mc AWS_ENDPOINT_URL_MOBILE
mq	mc AWS_ENDPOINT_URL_MQ
MTurk	m¹ AWS_ENDPOINT_URL_MTURK
MWAA	m\ AWS_ENDPOINT_URL_MWAA
Neptune	n AWS_ENDPOINT_URL_NEPTUNE
Neptune Graph	ne AWS_ENDPOINT_URL_NEPTUNE_GRAPH ra
neptunedata	n: AWS_ENDPOINT_URL_NEPTUNEDATA t:
Network Firewall	ne AWS_ENDPOINT_URL_NETWORK_FIREWALL i:
NetworkManager	ne AWS_ENDPOINT_URL_NETWORKMANAGER

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
NetworkMonitor	ne AWS_ENDPOINT_URL_NETWORKMONITOR n:
nimble	n: AWS_ENDPOINT_URL_NIMBLE
OAM	o: AWS_ENDPOINT_URL_OAM
Omics	or AWS_ENDPOINT_URL_OMICS
OpenSearch	o; AWS_ENDPOINT_URL_OPENSEARCH h
OpenSearchServerless	or AWS_ENDPOINT_URL_OPENSEARCHSERVERLESS hs
0psWorks	o; AWS_ENDPOINT_URL_OPSWORKS
0psWorksCM	o; AWS_ENDPOINT_URL_OPSWORKSCM
Organizations	o: AWS_ENDPOINT_URL_ORGANIZATIONS ic
OSIS	o: AWS_ENDPOINT_URL_OSIS
Outposts	OI AWS_ENDPOINT_URL_OUTPOSTS
p8data	p{ AWS_ENDPOINT_URL_P8DATA

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fil</service>
p8data	p{ AWS_ENDPOINT_URL_P8DATA
Panorama	p: AWS_ENDPOINT_URL_PANORAMA
Payment Cryptography	p: AWS_ENDPOINT_URL_PAYMENT_CRYPTOGRAPHY ry hy
Payment Cryptography Data	p: AWS_ENDPOINT_URL_PAYMENT_CRYPTOGRAPHY_DATA r: h:
Pca Connector Ad	pc AWS_ENDPOINT_URL_PCA_CONNECTOR_AD
Personalize	<pre>pe AWS_ENDPOINT_URL_PERSONALIZE ze</pre>
Personalize Events	<pre>pe AWS_ENDPOINT_URL_PERSONALIZE_EVENTS ze</pre>
Personalize Runtime	<pre>pe AWS_ENDPOINT_URL_PERSONALIZE_RUNTIME ze e</pre>
PI	p: AWS_ENDPOINT_URL_PI
Pinpoint	p: AWS_ENDPOINT_URL_PINPOINT

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
Pinpoint Email	p: AWS_ENDPOINT_URL_PINPOINT_EMAIL er
Pinpoint SMS Voice	p: AWS_ENDPOINT_URL_PINPOINT_SMS_VOICE sr
Pinpoint SMS Voice V2	<pre>p: AWS_ENDPOINT_URL_PINPOINT_SMS_VOICE_V2 sr _'</pre>
Pipes	p: AWS_ENDPOINT_URL_PIPES
Polly	pc AWS_ENDPOINT_URL_POLLY
Pricing	p: AWS_ENDPOINT_URL_PRICING
PrivateNetworks	p: AWS_ENDPOINT_URL_PRIVATENETWORKS to
Proton	p: AWS_ENDPOINT_URL_PROTON
QBusiness	ql AWS_ENDPOINT_URL_QBUSINESS
QConnect	q AWS_ENDPOINT_URL_QCONNECT
QLDB	q: AWS_ENDPOINT_URL_QLDB

serviceId	Se AWS_ENDPOINT_URL_ <service> environment variable id r ke fo st AI Co fil</service>
QLDB Session	q: AWS_ENDPOINT_URL_QLDB_SESSION
QuickSight	qı AWS_ENDPOINT_URL_QUICKSIGHT
RAM	r: AWS_ENDPOINT_URL_RAM
rbin	rl AWS_ENDPOINT_URL_RBIN
RDS	rc AWS_ENDPOINT_URL_RDS
RDS Data	rc AWS_ENDPOINT_URL_RDS_DATA
Redshift	re AWS_ENDPOINT_URL_REDSHIFT
Redshift Data	r AWS_ENDPOINT_URL_REDSHIFT_DATA
Redshift Serverless	<pre>re AWS_ENDPOINT_URL_REDSHIFT_SERVERLESS se</pre>
Rekognition	re AWS_ENDPOINT_URL_REKOGNITION or
repostspace	re AWS_ENDPOINT_URL_REPOSTSPACE

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
resiliencehub	r AWS_ENDPOINT_URL_RESILIENCEHUB
Resource Explorer 2	<pre>re AWS_ENDPOINT_URL_RESOURCE_EXPLORER_2 e; 2</pre>
Resource Groups	re AWS_ENDPOINT_URL_RESOURCE_GROUPS g:
Resource Groups Tagging API	<pre>re AWS_ENDPOINT_URL_RESOURCE_GROUPS_TAG g: GING_API g!</pre>
RoboMaker	rc AWS_ENDPOINT_URL_ROBOMAKER
RolesAnywhere	rc AWS_ENDPOINT_URL_ROLESANYWHERE
Route 53	rc AWS_ENDPOINT_URL_ROUTE_53
Route53 Recovery Cluster	r AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CLUSTER e 1

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
Route53 Recovery Control Config	rc AWS_ENDPOINT_URL_ROUTE53_RECOVERY_CO ec NTROL_CONFIG oc nt
Route53 Recovery Readiness	rc AWS_ENDPOINT_URL_ROUTE53_RECOVERY_RE ec ADINESS ea
Route 53 Domains	rc AWS_ENDPOINT_URL_ROUTE_53_DOMAINS dc
Route53Resolver	r AWS_ENDPOINT_URL_ROUTE53RESOLVER
RUM	rı AWS_ENDPOINT_URL_RUM
S3	s: AWS_ENDPOINT_URL_S3
S3 Control	s: AWS_ENDPOINT_URL_S3_CONTROL
S30utposts	s: AWS_ENDPOINT_URL_S30UTPOSTS s
SageMaker	s: AWS_ENDPOINT_URL_SAGEMAKER

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st AI Ct fil</service>
SageMaker A2I Runtime	s: AWS_ENDPOINT_URL_SAGEMAKER_A2I_RUNTIME _: ir
Sagemaker Edge	s: AWS_ENDPOINT_URL_SAGEMAKER_EDGE
SageMaker FeatureSt ore Runtime	s: AWS_ENDPOINT_URL_SAGEMAKER_FEATUREST _1 ORE_RUNTIME t: ir
SageMaker Geospatial	s: AWS_ENDPOINT_URL_SAGEMAKER_GEOSPATIAL _! a:
SageMaker Metrics	s: AWS_ENDPOINT_URL_SAGEMAKER_METRICS _r
SageMaker Runtime	s: AWS_ENDPOINT_URL_SAGEMAKER_RUNTIME _:
savingsplans	s: AWS_ENDPOINT_URL_SAVINGSPLANS
Scheduler	s AWS_ENDPOINT_URL_SCHEDULER

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
schemas	s AWS_ENDPOINT_URL_SCHEMAS
SimpleDB	s: AWS_ENDPOINT_URL_SIMPLEDB
Secrets Manager	s AWS_ENDPOINT_URL_SECRETS_MANAGER
SecurityHub	s. AWS_ENDPOINT_URL_SECURITYHUB
SecurityLake	s AWS_ENDPOINT_URL_SECURITYLAKE
ServerlessApplicat ionRepository	st AWS_ENDPOINT_URL_SERVERLESSAPPLICATI st ONREPOSITORY ic tc
Service Quotas	s AWS_ENDPOINT_URL_SERVICE_QUOTAS
Service Catalog	se AWS_ENDPOINT_URL_SERVICE_CATALOG ad
Service Catalog AppRegistry	st AWS_ENDPOINT_URL_SERVICE_CATALOG_APP at REGISTRY p:

serviceId	Se AWS_ENDPOINT_URL_ <service> environment variable id r ke fo st AI Co fil</service>
ServiceDiscovery	se AWS_ENDPOINT_URL_SERVICEDISCOVERY
SES	s AWS_ENDPOINT_URL_SES
SESv2	se AWS_ENDPOINT_URL_SESV2
Shield	sl AWS_ENDPOINT_URL_SHIELD
signer	s: AWS_ENDPOINT_URL_SIGNER
SimSpaceWeaver	s: AWS_ENDPOINT_URL_SIMSPACEWEAVER e;
SMS	sr AWS_ENDPOINT_URL_SMS
Snow Device Managemen t	SI AWS_ENDPOINT_URL_SNOW_DEVICE_MANAGEMENT CO
Snowball	sr AWS_ENDPOINT_URL_SNOWBALL
SNS	sr AWS_ENDPOINT_URL_SNS
SQS	sc AWS_ENDPOINT_URL_SQS
SSM	s: AWS_ENDPOINT_URL_SSM
SSM Contacts	s: AWS_ENDPOINT_URL_SSM_CONTACTS

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
SSM Incidents	s: AWS_ENDPOINT_URL_SSM_INCIDENTS er
Ssm Sap	s: AWS_ENDPOINT_URL_SSM_SAP
SS0	s: AWS_ENDPOINT_URL_SSO
SSO Admin	s: AWS_ENDPOINT_URL_SSO_ADMIN
SSO OIDC	s: AWS_ENDPOINT_URL_SSO_OIDC
SFN	s AWS_ENDPOINT_URL_SFN
Storage Gateway	st AWS_ENDPOINT_URL_STORAGE_GATEWAY
STS	st AWS_ENDPOINT_URL_STS
SupplyChain	sı AWS_ENDPOINT_URL_SUPPLYCHAIN
Support	si AWS_ENDPOINT_URL_SUPPORT
Support App	si AWS_ENDPOINT_URL_SUPPORT_APP
SWF	sv AWS_ENDPOINT_URL_SWF

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st All Ct fill</service>
synthetics	s: AWS_ENDPOINT_URL_SYNTHETICS s
Textract	t AWS_ENDPOINT_URL_TEXTRACT
Timestream InfluxDB	t: AWS_ENDPOINT_URL_TIMESTREAM_INFLUXDB m_ b
Timestream Query	t: AWS_ENDPOINT_URL_TIMESTREAM_QUERY m_
Timestream Write	t: AWS_ENDPOINT_URL_TIMESTREAM_WRITE m_
tnb	tr AWS_ENDPOINT_URL_TNB
Transcribe	t: AWS_ENDPOINT_URL_TRANSCRIBE e
Transfer	t: AWS_ENDPOINT_URL_TRANSFER
Translate	t: AWS_ENDPOINT_URL_TRANSLATE
TrustedAdvisor	t: AWS_ENDPOINT_URL_TRUSTEDADVISOR v:

serviceId	St AWS_ENDPOINT_URL_ <service> environment variable id r kt fo st Al Ct fil</service>
VerifiedPermissions	<pre>v AWS_ENDPOINT_URL_VERIFIEDPERMISSIONS e: s</pre>
Voice ID	vc AWS_ENDPOINT_URL_VOICE_ID
VPC Lattice	V AWS_ENDPOINT_URL_VPC_LATTICE
WAF	wa AWS_ENDPOINT_URL_WAF
WAF Regional	w: AWS_ENDPOINT_URL_WAF_REGIONAL
WAFV2	wa AWS_ENDPOINT_URL_WAFV2
WellArchitected	<pre>we AWS_ENDPOINT_URL_WELLARCHITECTED te</pre>
Wisdom	w: AWS_ENDPOINT_URL_WISDOM
WorkDocs	wc AWS_ENDPOINT_URL_WORKDOCS
WorkLink	wc AWS_ENDPOINT_URL_WORKLINK
WorkMail	wc AWS_ENDPOINT_URL_WORKMAIL

serviceId	Se AWS_ENDPOINT_URL_ <service> environment variable id r ke fo sh Ai Co fil</service>
WorkMailMessageFlow	wc AWS_ENDPOINT_URL_WORKMAILMESSAGEFLOW e: w
WorkSpaces	w AWS_ENDPOINT_URL_WORKSPACES
WorkSpaces Thin Client	wc AWS_ENDPOINT_URL_WORKSPACES_THIN_CLIENT s_ ic
WorkSpaces Web	wc AWS_ENDPOINT_URL_WORKSPACES_WEBs_
XRay	x: AWS_ENDPOINT_URL_XRAY

Smart configuration defaults



Note

For help in understanding the layout of settings pages, or in interpreting the **Support by** Amazon SDKs and tools table that follows, see Understanding the settings pages of this guide.

With the smart configuration defaults feature, Amazon SDKs can provide predefined, optimized default values for other configuration settings.

Configure this functionality by using the following:

defaults_mode - shared Amazon config file setting, AWS_DEFAULTS_MODE - environment variable, aws.defaultsMode - JVM system property: Java/Kotlin only

With this setting, you can choose a mode that aligns with your application architecture, which then provides optimized default values for your application. If an Amazon SDK setting has a value explicitly set, then that value always takes precedence. If an Amazon SDK setting does not have a value explicitly set, and defaults_mode is not equal to legacy, then this feature can provide different default values for various settings optimized for your application. Settings may include the following: HTTP communication settings, retry behavior, service Regional endpoint settings, and, potentially, any SDK-related configuration. Customers who use this feature can get new configuration defaults tailored to common usage scenarios. If your defaults_mode is not equal to legacy, we recommend performing tests of your application when you upgrade the SDK, because the default values provided might change as best practices evolve.

Default value: legacy

Note: New major versions of SDKs will default to standard.

Valid values:

- legacy Provides default settings that vary by SDK and existed before establishment of defaults mode.
- standard Provides the latest recommended default values that should be safe to run in most scenarios.
- in-region Builds on the standard mode and includes optimization tailored for applications that call Amazon Web Services services from within the same Amazon Web Services Region.
- cross-region Builds on the standard mode and includes optimization tailored for applications that call Amazon Web Services services in a different Region.
- mobile Builds on the standard mode and includes optimization tailored for mobile applications.
- auto Builds on the standard mode and includes experimental features. The SDK attempts
 to discover the runtime environment to determine the appropriate settings automatically.
 The auto detection is heuristics-based and does not provide 100% accuracy. If the runtime
 environment can't be determined, standard mode is used. The auto detection might

query <u>instance metadata</u>, which might introduce latency. If startup latency is critical to your application, we recommend choosing an explicit defaults_mode instead.

Example of setting this value in the config file:

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

The following parameters might be optimized based on the selection of defaults_mode:

- retryMode Specifies how the SDK attempts retries. See <u>Retry behavior</u>.
- stsRegionalEndpoints Specifies how the SDK determines the Amazon Web Services service endpoint that it uses to talk to the Amazon Security Token Service (Amazon STS). See Amazon STS Regional endpoints.
- s3UsEast1RegionalEndpoints Specifies how the SDK determines the Amazon service endpoint that it uses to talk to the Amazon S3 for the us-east-1 Region.
- connectTimeoutInMillis After making an initial connection attempt on a socket, the amount of time before timing out. If the client does not receive a completion of the connect handshake, the client gives up and fails the operation.
- tlsNegotiationTimeoutInMillis The maximum amount of time that a TLS handshake can take from the time the CLIENT HELLO message is sent to the time the client and server have fully negotiated ciphers and exchanged keys.

The default value for each setting changes depending on the defaults_mode selected for your application. These values are currently set as follows (subject to change):

Parameter	standard mode	in-region mode	cross-reg ion mode	mobile mode
retryMode	standard	standard	standard	standard
stsRegion alEndpoin ts	regional	regional	regional	regional

Parameter	standard mode	in-region mode	cross-reg ion mode	mobile mode
s3UsEast1 RegionalE ndpoints	regional	regional	regional	regional
connectTi meoutInMi llis	3100	1100	3100	30000
tlsNegoti ationTime outInMill is	3100	1100	3100	30000

For example, if the defaults_mode you selected was standard, then the value of standard would be assigned for retry_mode (from the valid retry_mode options) and the value of regional would be assigned for stsRegionalEndpoints (from the valid stsRegionalEndpoints options).

Support by Amazon SDKs and tools

The following SDKs support the features and settings described in this topic. Any partial exceptions are noted. Any JVM system property settings are supported by the Amazon SDK for Java and the Amazon SDK for Kotlin only.

SDK	Supported	Notes or more information
Amazon CLI v2	No	
SDK for C++	Yes	Parameters not optimized: stsRegionalEndpoints , s3UsEast1RegionalE ndpoints ,tlsNegoti ationTimeoutInMill is .

SDK	Supported	Notes or more information
SDK for Go V2 (1.x)	Yes	Parameters not optimized: retryMode , stsRegion alEndpoints , s3UsEast1RegionalE ndpoints .
SDK for Go 1.x (V1)	No	
SDK for Java 2.x	Yes	Parameters not optimized: stsRegionalEndpoints .
SDK for Java 1.x	No	
SDK for JavaScript 3.x	Yes	Parameters not optimized: stsRegionalEndpoints , s3UsEast1RegionalE ndpoints ,tlsNegoti ationTimeoutInMill is .connectTi meoutInMillis is called connectionTimeout .
SDK for JavaScript 2.x	No	
SDK for Kotlin	No	
SDK for .NET 4.x	Yes	Parameters not optimized :connectTimeoutInMi llis ,tlsNegoti ationTimeoutInMill is .

SDK	Supported	Notes or more information
SDK for .NET 3.x	Yes	Parameters not optimized : connectTimeoutInMi llis ,tlsNegoti ationTimeoutInMill is .
SDK for PHP 3.x	Yes	Parameters not optimized : tlsNegotiationTime outInMillis .
SDK for Python (Boto3)	Yes	Parameters not optimized : tlsNegotiationTime outInMillis .
SDK for Ruby 3.x	Yes	
SDK for Rust	No	
SDK for Swift	No	
Tools for PowerShell V5	Yes	Parameters not optimized : connectTimeoutInMi llis ,tlsNegoti ationTimeoutInMill is .
Tools for PowerShell V4	Yes	Parameters not optimized : connectTimeoutInMi llis ,tlsNegoti ationTimeoutInMill is .

Amazon Common Runtime (CRT) libraries

The Amazon Common Runtime (CRT) libraries are a base library of the SDKs. The CRT is a modular family of independent packages, written in C. Each package provides good performance and minimal footprint for different required functionalities. These functionalities are common and shared across all SDKs providing better code reuse, optimization, and accuracy. The packages are:

- <u>awslabs/aws-c-auth</u>: Amazon client-side authentication (standard credential providers and signing (sigv4))
- <u>awslabs/aws-c-cal</u>: Cryptographic primitive types, hashes (MD5, SHA256, SHA256 HMAC), signers, AES
- <u>awslabs/aws-c-common</u>: Basic data structures, threading/synchronization primitive types, buffer management, stdlib-related functions
- awslabs/aws-c-compression: Compression algorithms (Huffman encoding/decoding)
- <u>awslabs/aws-c-event-stream</u>: Event stream message processing (headers, prelude, payload, crc/trailer), remote procedure call (RPC) implementation over event streams
- awslabs/aws-c-http: C99 implementation of the HTTP/1.1 and HTTP/2 specifications
- <u>awslabs/aws-c-io</u>: Sockets (TCP, UDP), DNS, pipes, event loops, channels, SSL/TLS
- <u>awslabs/aws-c-iot</u>: C99 implementation of Amazon IoT cloud services integration with devices
- <u>awslabs/aws-c-mqtt</u>: Standard, lightweight messaging protocol for the Internet of Things (IoT)
- <u>awslabs/aws-c-s3</u>: C99 library implementation for communicating with the Amazon S3 service, designed for maximizing throughput on high bandwidth Amazon EC2 instances
- <u>awslabs/aws-c-sdkutils</u>: A utilities library for parsing and managing Amazon profiles
- <u>awslabs/aws-checksums</u>: Cross-platform hardware-accelerated CRC32c and CRC32 with fallback to efficient software implementations
- <u>awslabs/aws-lc</u>: General-purpose cryptographic library maintained by the Amazon
 Cryptography team for Amazon and their customers, based on code from the Google BoringSSL
 project and the OpenSSL project
- awslabs/s2n: C99 implementation of the TLS/SSL protocols, designed to be small and fast with security as a priority

The CRT is available through all SDKs except Go and Rust.

CRT dependencies

The CRT libraries form a complex net of relationships and dependencies. Knowing these relationships is helpful if you need to build the CRT directly from source. However, most users access CRT functionality through their language SDK (such as Amazon SDK for C++ or Amazon SDK for Java) or their language IoT device SDK (such as Amazon IoT SDK for C++ or Amazon IoT SDK for Java). In the following diagram, the Language CRT Bindings box refers to the package wrapping the CRT libraries for a specific language SDK. This is a collection of packages of the form aws-crt-*, where '*' is an SDK language (such as aws-crt-cpp or aws-crt-java).

The following is an illustration of the hierarchical dependencies of the CRT libraries.

CRT dependencies 195

Amazon SDKs and Tools maintenance policy

Overview

This document outlines the maintenance policy for Amazon Software Development Kits (SDKs) and Tools, including Mobile and IoT SDKs, and their underlying dependencies. Amazon regularly provides the Amazon SDKs and Tools with updates that may contain support for new or updated Amazon APIs, new features, enhancements, bug fixes, security patches, or documentation updates. Updates may also address changes with dependencies, language runtimes, and operating systems. Amazon SDK releases are published to package managers (e.g. Maven, NuGet, PyPI), and are available as source code on GitHub.

We recommend users to stay up-to-date with SDK releases to keep up with the latest features, security updates, and underlying dependencies. Continued use of an unsupported SDK version is not recommended and is done at the user's discretion.

Versioning

The Amazon SDK release versions are in the form of X.Y.Z where X represents the major version. Increasing the major version of an SDK indicates that this SDK underwent significant and substantial changes to support new idioms and patterns in the language. Major versions are introduced when public interfaces (e.g. classes, methods, types, etc.), behaviors, or semantics have changed. Applications need to be updated in order for them to work with the newest SDK version. It is important to update major versions carefully and in accordance with the upgrade guidelines provided by Amazon.

SDK major version lifecycle

The lifecycle for major SDKs and Tools versions consists of 5 phases, which are outlined below.

Developer Preview (Phase 0) - During this phase, SDKs are not supported, should not be used
in production environments, and are meant for early access and feedback purposes only. It is
possible for future releases to introduce breaking changes. Once Amazon identifies a release to
be a stable product, it may mark it as a Release Candidate. Release Candidates are ready for GA
release unless significant bugs emerge, and will receive full Amazon support.

Overview 196

• General Availability (GA) (Phase 1) - During this phase, SDKs are fully supported. Amazon will provide regular SDK releases that include support for new services, API updates for existing services, as well as bug and security fixes. For Tools, Amazon will provide regular releases that include new feature updates and bug fixes. Amazon will support the GA version of an SDK for at least 24 months.

- Maintenance Announcement (Phase 2) Amazon will make a public announcement at least 6 months before an SDK enters maintenance mode. During this period, the SDK will continue to be fully supported. Typically, maintenance mode is announced at the same time as the next major version is transitioned to GA.
- Maintenance (Phase 3) During the maintenance mode, Amazon limits SDK releases to address
 critical bug fixes and security issues only. An SDK will not receive API updates for new or existing
 services, or be updated to support new regions. Maintenance mode has a default duration of 12
 months, unless otherwise specified.
- End-of-Support (Phase 4) When an SDK reaches end-of support, it will no longer receive updates or releases. Previously published releases will continue to be available via public package managers and the code will remain on GitHub. The GitHub repository may be archived. Use of an SDK which has reached end-of-support is done at the user's discretion. We recommend users upgrade to the new major version.

The following is a visual illustration of the SDK major version lifecycle. Please note that the timelines shown below are illustrative and not binding.

Dependency lifecycle

Most Amazon SDKs have underlying dependencies, such as language runtimes, operating systems, or third party libraries and frameworks. These dependencies are typically tied to the language community or the vendor who owns that particular component. Each community or vendor publishes their own end-of-support schedule for their product.

The following terms are used to classify underlying third party dependencies:

- Operating System (OS): Examples include Amazon Linux AMI, Amazon Linux 2, Windows 2008, Windows 2012, Windows 2016, etc.
- Language Runtime: Examples include Java 7, Java 8, Java 11, .NET Core, .NET Standard, .NET PCL, etc.
- Third party Library / Framework: Examples include OpenSSL, .NET Framework 4.5, Java EE, etc.

Dependency lifecycle 197

Our policy is to continue supporting SDK dependencies for at least 6 months after the community or vendor ends support for the dependency. This policy, however, could vary depending on the specific dependency.



Note

Amazon reserves the right to stop support for an underlying dependency without increasing the major SDK version

Communication methods

Maintenance announcements are communicated in several ways:

- An email announcement is sent to affected accounts, announcing our plans to end support for the specific SDK version. The email will outline the path to end-of-support, specify the campaign timelines, and provide upgrade guidance.
- Amazon SDK documentation, such as API reference documentation, user guides, SDK product marketing pages, and GitHub readme(s) are updated to indicate the campaign timeline and provide guidance on upgrading affected applications.
- An Amazon blog post is published that outlines the path to end-of-support, as well as reiterates the campaign timelines.
- Deprecation warnings are added to the SDKs, outlining the path to end-of-support and linking to the SDK documentation.

To see the list of available major versions of Amazon SDKs and Tools and where they are in their maintenance lifecycle, see Version lifecycle.

Communication methods 198

Amazon SDKs and Tools version lifecycle

The table below shows the list of available Amazon Software Development Kit (SDK) major versions and where they are in the maintenance lifecycle with associated timelines. For detailed information on the lifecycle for the major versions of Amazon SDKs and Tools and their underlying dependencies, see *Maintenance policy*.

SDK	Major version	Current Phase	General Availability Date	Notes
Amazon CLI	1.x	General Availabil ity	9/2/2013	
Amazon CLI	2.x	General Availabil ity	2/10/2020	
SDK for C++	1.x	General Availabil ity	9/2/2015	
SDK for Go V2	V2 1.x	General Availabil ity	1/19/2021	
SDK for Go	1.x	Maintenance	11/19/2015	See <u>announcem</u> <u>ent</u> for details and dates
SDK for Java	1.x	Maintenance	3/25/2010	See <u>announcem</u> <u>ent</u> for details and dates
SDK for Java	2.x	General Availabil ity	11/20/2018	
SDK for JavaScript	1.x	End-of-Support	5/6/2013	

SDK	Major version	Current Phase	General Availability Date	Notes
SDK for JavaScript	2.x	Maintenance	6/19/2014	See <u>announcem</u> <u>ent</u> for details and dates
SDK for JavaScript	3.x	General Availabil ity	12/15/2020	
SDK for Kotlin	1.x	General Availabil ity	11/27/2023	
SDK for .NET	1.x	End-of-Support	11/2009	
SDK for .NET	2.x	End-of-Support	11/8/2013	
SDK for .NET	3.x	General Availabil ity	7/28/2015	
SDK for .NET	4.x	General Availabil ity	4/28/2025	
SDK for PHP	2.x	End-of-Support	11/2/2012	
SDK for PHP	3.x	General Availabil ity	5/27/2015	
SDK for Python (Boto2)	1.x	End-of-Support	7/13/2011	
SDK for Python (Boto3)	1.x	General Availabil ity	6/22/2015	
SDK for Python (Botocore)	1.x	General Availabil ity	6/22/2015	
SDK for Ruby	1.x	End-of-Support	7/14/2011	

SDK	Major version	Current Phase	General Availability Date	Notes
SDK for Ruby	2.x	End-of-Support	2/15/2015	
SDK for Ruby	3.x	General Availabil ity	8/29/2017	
SDK for Rust	1.x	General Availabil ity	11/27/2023	
SDK for Swift	1.x	General Availabil ity	9/17/2024	
Tools for PowerShell	2.x	End-of-Support	11/8/2013	
Tools for PowerShell	3.x	End-of-Support	7/29/2015	
Tools for PowerShell	4.x	General Availabil ity	11/21/2019	
Tools for PowerShell	5.x	General Availabil ity	6/23/2025	

Searching for an SDK or tool not mentioned? Encryption SDKs, IoT Device SDKs, and Mobile SDKs, for example, are not included in this guide. To find documentation on these other tools, see <u>Tools</u> to Build on Amazon.

Document history for Amazon SDKs and Tools Reference Guide

The following table describes important additions and updates to the *Amazon SDKs and Tools Reference Guide*. For notification about updates to this documentation, you can subscribe to the RSS feed.

Change	Description	Date
Adding new version of Tools for PowerShell	Adding latest version of Tools for PowerShell support to all Setting reference Compatibility with Amazon SDKs tables. Added Host prefix injection feature.	June 23, 2025
Page title updates	More titles, table titles, abstracts, and SEO updates.	March 5, 2025
Page title updates	Updating content to use more descriptive titles.	February 24, 2025
Adding Swift SDK to Settings reference	Adding Swift SDK support to all Setting reference Compatibility with Amazon SDKs tables.	September 17, 2024
SDK for Java 1.x system properties	Add details about supported JVM system configuration settings by the Amazon SDK for Java 1.x.	May 30, 2024
Settings updates	Add JVM system configura tion settings.	March 27, 2024

Compatibility table updates	Updates to compatibility for SDK support, updates to IAM Identity Center procedures.	February 20, 2024
Container credential update. IMDS update.	Adding support for Amazon EKS. Adding setting to disable IMDSv1 fallback.	December 29, 2023
Request compression	Adding settings for request compression feature.	December 27, 2023
Compatibility tables	Compatibility tables for SDK and tool features updated to include SDK for Kotlin, SDK for Rust, and Amazon Tools for PowerShell.	December 10, 2023
Authentication updates	Updates to supported methods of authentication for SDKs and tools.	July 1, 2023
IAM best practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	February 27, 2023
SSO updates	Updates to SSO credentia ls for the new SSO token configuration.	November 19, 2022
Settings updates	Updates to support table for General configuration and for Amazon S3 Multi-Region Access Points.	November 17, 2022

Settings updates	Updates to clarity of IMDS client and IMDS credentia ls. Updates to Environment variables.	November 4, 2022
Updating welcome page	Announcing Amazon CodeWhisperer.	September 22, 2022
Service name change for single sign-on	Updates to reflect that Amazon SSO is now referred to as Amazon IAM Identity Center.	July 26, 2022
Settings update	Minor updates to config file details and to supported settings.	June 15, 2022
<u>Update</u>	Massive update of almost all parts of this guide.	February 1, 2022
<u>Initial release</u>	The first release of this guide is released to the public.	March 13, 2020