

Developer Guide

AWS SDK for C++



AWS SDK for C++: Developer Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is the AWS SDK for C++ Developer Guide?	1
Additional documentation and resources	1
Maintenance and support for SDK major versions	2
Get started	3
SDK authentication with AWS	3
Start an AWS access portal session	4
More authentication information	5
Get the SDK from source	6
Build on Windows	6
Build on Linux/macOS	11
"Hello S3" app	14
Get the SDK from a package manager	20
Prerequisites	7
Get the SDK using vcpkg	21
Troubleshoot build issues	22
CMake Error: Could not find a package configuration file provided by "AWSSDK"	22
CMake Error: Could not find load file (and you're on SDK version 1.8)	23
CMake Error: Could not find load file	24
Runtime Error: cannot proceed because aws-*.dll was not found	24
Configure	26
CMake parameters	26
General CMake Variables and Options	26
Android CMake Variables and Options	41
Override your HTTP client	44
Control iostreams used by the HttpClient and the AWSClient	44
Use the SDK	45
Basic Use	45
Initializing and Shutting Down the SDK	45
Setting SDK Options	46
Credential provider chain	48
Default AWS Region	48
AWS Client configuration	48
Configuration Variables	50
Service Client Classes	52

Utility Modules	53
HTTP Stack	53
String Utils	53
Hashing Utils	53
JSON Parser	53
XML Parser	54
Memory Management	54
Allocating and Deallocating Memory	54
STL and AWS Strings and Vectors	55
Remaining Issues	57
Native SDK Developers and Memory Controls	57
Logging	58
Error Handling	60
Using a custom libcrypto	61
How to build a custom libcrypto into the SDK for C++	61
Bringing it all together in a docker image	63
Work with AWS services	65
Get started on code examples	65
Structure of the code examples	65
Using AWS for troubleshooting and diagnostics	66
Building and Debugging Code Examples in Visual Studio	70
Asynchronous methods	71
Asynchronous SDK methods	71
Calling SDK asynchronous methods	72
Notification of the Completion of an Asynchronous Operation	73
Code examples with guidance	76
Amazon CloudWatch examples	77
Amazon DynamoDB examples	93
Amazon EC2 examples	108
Amazon S3 examples	133
Amazon SQS examples	168
Code examples	184
Actions and scenarios	184
ACM	185
Aurora	215
Auto Scaling	258

CloudTrail	288
CloudWatch	293
CloudWatch Logs	304
CodeBuild	308
Amazon Cognito Identity Provider	313
DynamoDB	336
Amazon EC2	407
EventBridge	434
AWS Glue	438
HealthImaging	476
IAM	511
AWS IoT	552
AWS IoT data	593
Lambda	596
MediaConvert	620
Amazon RDS	629
Amazon Rekognition	665
Amazon S3	670
Secrets Manager	714
Amazon SES	716
Amazon SNS	738
Amazon SQS	779
AWS STS	815
Amazon Transcribe	817
Cross-service examples	821
Create a serverless application to manage photos	821
Create an Aurora Serverless work item tracker	822
Security	823
Data Protection	823
Identity and Access Management	824
Audience	825
Authenticating with identities	825
Managing access using policies	829
How AWS services work with IAM	831
Troubleshooting AWS identity and access	831
Compliance Validation	833

Resilience	834
Infrastructure Security	835
Enforcing a minimum TLS version	835
Enforce specific TLS version with libcurl on all platforms	836
Enforce specific TLS version on Windows	837
Amazon S3 Encryption Client Migration	840
Migration Overview	840
Update Existing Clients to Read New Formats	841
Migrate Encryption and Decryption Clients to V2	842
Additional Examples	844
Document history	848

What is the AWS SDK for C++ Developer Guide?

Welcome to the AWS SDK for C++ Developer Guide.

The AWS SDK for C++ provides a modern C++ (version C++ 11 or later) interface for Amazon Web Services (AWS). It provides both high-level and low-level APIs for nearly all AWS features, minimizing dependencies and providing platform portability on Windows, macOS, Linux, and mobile.

[Get started with the AWS SDK for C++](#)

Note

The AWS IoT SDKs and the `aws-iot-device-sdk-cpp` are separate from this SDK. The AWS IoT Device SDK for C++ v2 is available at [aws-iot-device-sdk-cpp-v2](#) on GitHub. For more information about AWS IoT, see [What is AWS IoT](#) in the AWS IoT Developer Guide.

Additional documentation and resources

In addition to this guide, the following are valuable online resources for AWS SDK for C++ developers:

- [AWS SDKs and Tools Reference Guide](#): Contains settings, features, and other foundational concepts common amongst AWS SDKs.
- GitHub:
 - [SDK source](#)
 - [SDK issues](#)
- [AWS SDK for C++ API Reference](#)
- [AWS C++ Developer Blog](#)
- The [AWS Code Sample Catalog](#)
- [SDK License](#)
- *Video:* [Introducing the AWS SDK for C++ from AWS re:invent 2015](#)

Maintenance and support for SDK major versions

For information about maintenance and support for SDK major versions and their underlying dependencies, see the following in the [AWS SDKs and Tools Reference Guide](#):

- [AWS SDKs and tools maintenance policy](#)
- [AWS SDKs and tools version support matrix](#)

Get started with the AWS SDK for C++

AWS SDK for C++ is a modularized, cross-platform, open-source library you can use to connect to Amazon Web Services.

The AWS SDK for C++ uses [CMake](#) to support multiple platforms over multiple domains, including video games, systems, mobile, and embedded devices. CMake is a build tool that you can use to manage your application's dependencies and to create makefiles suitable for the platform you're building on. CMake removes the parts of the build that are not used for your platform or application.

Before you run code to access AWS resources, you must establish how your code authenticates with AWS.

- [SDK authentication with AWS](#)

To use the AWS SDK for C++ in your code, obtain the SDK executables by building the SDK source directly or by using a package manager.

- [Get the AWS SDK for C++ from source code](#)
- [Get the AWS SDK for C++ from a package manager](#)

If you run into build issues regarding CMake, see [Troubleshoot build issues](#).

SDK authentication with AWS

You must establish how your code authenticates with AWS when developing with AWS services. There are different ways in which you can configure programmatic access to AWS resources, depending on the environment and the AWS access available to you.

To choose your method of authentication and configure it for the SDK, see [Authentication and access](#) in the *AWS SDKs and Tools Reference Guide*.

We recommend that new users who are developing locally, and are not given a method of authentication by their employer, should set up AWS IAM Identity Center. This method includes installing the AWS CLI for ease of configuration and for regularly signing in to the AWS access portal. If you choose this method, your environment should contain the following elements after

you complete the procedure for [IAM Identity Center authentication](#) in the *AWS SDKs and Tools Reference Guide*:

- The AWS CLI, which you use to start an AWS access portal session before you run your application.
- A [shared AWSconfig file](#) having a [default] profile with a set of configuration values that can be referenced from the SDK. To find the location of this file, see [Location of the shared files](#) in the *AWS SDKs and Tools Reference Guide*.
- The shared config file sets the [region](#) setting. This sets the default AWS Region that the SDK uses for AWS requests. This Region is used for SDK service requests that aren't specified with a Region to use.
- The SDK uses the profile's [SSO token provider configuration](#) to acquire credentials before sending requests to AWS. The `sso_role_name` value, which is an IAM role connected to an IAM Identity Center permission set, should allow access to the AWS services used in your application.

The following sample config file shows a default profile set up with SSO token provider configuration. The profile's `sso_session` setting refers to the named [sso-session section](#). The `sso-session` section contains settings to initiate an AWS access portal session.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

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

The AWS SDK for C++ does not need additional packages (such as SS0 and SS00IDC) to be added to your application to use IAM Identity Center authentication.

Start an AWS access portal session

Before running an application that accesses AWS services, you need an active AWS access portal session for the SDK to use IAM Identity Center authentication to resolve credentials. Depending on

your configured session lengths, your access will eventually expire and the SDK will encounter an authentication error. To sign in to the AWS access portal, run the following command in the AWS CLI.

```
aws sso login
```

Since you have a default profile setup, you do not need to call the command with a `--profile` option. If your SSO token provider configuration is using a named profile, the command is `aws sso login --profile named-profile`.

To test if you already have an active session, run the following AWS CLI command.

```
aws sts get-caller-identity
```

The response to this command should report the IAM Identity Center account and permission set configured in the shared `config` file.

Note

If you already have an active AWS access portal session and run `aws sso login`, you will not be required to provide credentials.

The sign-in process might prompt you to allow the AWS CLI access to your data. Because the AWS CLI is built on top of the SDK for Python, permission messages may contain variations of the `botocore` name.

More authentication information

Human users, also known as *human identities*, are the people, administrators, developers, operators, and consumers of your applications. They must have an identity to access your AWS environments and applications. Human users that are members of your organization are also known as *workforce identities*, that means you, the developer. Use temporary credentials when accessing AWS. You can use an identity provider for your human users to provide federated access to AWS accounts by assuming roles, which provide temporary credentials. For centralized access management, we recommend that you use AWS IAM Identity Center (IAM Identity Center) to manage access to your accounts and permissions within those accounts. For more alternatives, see the following:

- To learn more about best practices, see [Security best practices in IAM](#) in the *IAM User Guide*.
- To create short-term AWS credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*.
- To learn about other AWS SDK for C++ credential providers, see [Standardized credential providers](#) in the *AWS SDKs and Tools Reference Guide*.

Get the AWS SDK for C++ from source code

You can use the AWS SDK for C++ from your code by first building the SDK from source and then installing it locally.

Process overview

General process	Detailed process
<p>Build and install the SDK source</p> <ol style="list-style-type: none"> 1. Use CMake to generate build files for the SDK. 2. Build the SDK. 3. Install the SDK. 	<p>First build the SDK from source and install it.</p> <ul style="list-style-type: none"> • Build on Windows • Build on Linux/macOS
<p>Build your application using the SDK</p> <ol style="list-style-type: none"> 1. Write your own code to use the SDK or use a sample application, and add the AWSSDK package to your cmake file. 2. Use CMake to generate build files for your application. 3. Build your application. 4. Run your application. 	<p>Then develop your own application using the SDK.</p> <ul style="list-style-type: none"> • "Hello, S3" starter application

Build the AWS SDK for C++ on Windows

To set up the AWS SDK for C++, you can either build the SDK yourself directly from the source or download the libraries using a package manager.

The SDK source is separated into individual packages by service. Installing the entire SDK can take up to an hour. Installing only the specific subset of services that your program uses decreases installation time and also reduces size on disk. To choose which services to install, you need to know the package name of each service your program uses. You can see the list of package directories at [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) on GitHub. The package name is the suffix of the directory name for the service.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName> # Repo directory name and packageName
aws-sdk-cpp\aws-cpp-sdk-s3           # Example: Package name is s3
```

Prerequisites

You need a minimum of 4 GB of RAM to build some of the larger AWS clients. The SDK might fail to build on Amazon EC2 instance types *t2.micro*, *t2.small*, and other small instance types due to insufficient memory.

To use the AWS SDK for C++, you need one of the following:

- Microsoft Visual Studio 2015 or later,
- GNU Compiler Collection (GCC) 4.9 or later, or
- Clang 3.3 or later.

Building the SDK for Windows with curl

On Windows, the SDK is built with [WinHTTP](#) as the default HTTP client. However, WinHTTP 1.0 does not support HTTP/2 bidirectional streaming, which is required for some AWS services such as Amazon Transcribe and Amazon Lex. Thus, it is sometimes necessary to build curl support with the SDK. To view all available curl download options, see [curl Releases and Downloads](#). One method for building the SDK with curl support is the following:

To build the SDK with curl library support included

1. Navigate to [curl for Windows](#) and download the curl binary package for Microsoft Windows.
2. Unpack the package to a folder on your computer, for example, `C:\curl`.
3. Navigate to [CA certificates extracted from Mozilla](#) and download the `cacert.pem` file. This Privacy Enhanced Mail (PEM) file contains a bundle of valid digital certificates that are used to verify the authenticity of secure websites. The certificates are distributed by certificate authority (CA) companies such as GlobalSign and Verisign.

4. Move the `cacert.pem` file to the `bin` subfolder that you unpacked in a previous step, for example, `C:\curl\bin`. Rename the file as `curl-ca-bundle.crt`.

Also, the Microsoft Build Engine (MSBuild) must be able to locate the `curl.dll` in the procedure that follows. Therefore, you should add the `curl bin` folder path to your Windows `PATH` environment variable, for example, set `PATH=%PATH%;C:\curl\bin`. You must add this each time you open a new command prompt to build the SDK. Alternatively, you can set the environment variable globally in your Windows system settings so that the setting is remembered.

When *Building the SDK from source* in the procedure that follows, see Step 5 (Generate build files) for required command syntax to build `curl` into your SDK.

When writing your code, you must set `caFile` in the [AWS Client configuration](#) to the location of your certificate file. For an example using Amazon Transcribe, see [transcribe](#) in the *AWS Code Examples Repository* on GitHub.

Building the SDK from source

You can build the SDK from source using command-line tools. Using this method, you can customize your SDK build. For information about available options, see [CMake Parameters](#). There are three main steps. First, you build the files using CMake. Second, you use MSBuild to build the SDK binaries that work with your operating system and build toolchain. Third, you install or copy the binaries into the correct location on the development machine.

To build the SDK from source

1. Install [CMake](#) (minimum version 3.13) and the relevant build tools for your platform. It is recommended to add `cmake` to your `PATH`. To check your version of CMake, open a command prompt and run command **`cmake --version`**
2. In a command prompt, navigate to a folder where you want to store the SDK.
3. Get the latest source code.

Version 1.11 uses git submodules to wrap external dependencies. This includes the [CRT libraries](#) described in the *AWS SDKs and Tools Reference Guide*.

Download or clone the SDK source from [aws/aws-sdk-cpp](#) on GitHub:

- Clone with Git: HTTPS

```
git clone --recurse-submodules https://github.com/aws/aws-sdk-cpp
```

- Clone with Git: SSH

```
git clone --recurse-submodules git@github.com:aws/aws-sdk-cpp.git
```

4. We recommend you store the generated build files outside of the SDK source directory. Create a new directory to store the build files in and navigate to that folder.

```
mkdir sdk_build  
cd sdk_build
```

5. Generate the build files by running cmake. Specify on the cmake command line whether to build a *Debug* or *Release* version. Choose Debug throughout this procedure to run a debug configuration of your application code. Choose Release throughout this procedure to run a release configuration of your application code. For Windows, the SDK install location is typically `\Program Files (x86)\aws-cpp-sdk-all\`. Command syntax:

```
{path to cmake if not in PATH} {path to source location of aws-sdk-cpp} -DCMAKE_BUILD_TYPE=[Debug | Release] -DCMAKE_PREFIX_PATH={path to install destination}
```

For more ways to modify the build output, see [CMake Parameters](#).

To generate the build files, do one of the following:

- **Generate build files (all AWS services):** To build the entire SDK, run cmake, specifying whether to build a *Debug* or *Release* version. For example:

```
cmake "..\aws-sdk-cpp" -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH="C:\Program Files (x86)\aws-cpp-sdk-all"
```

- **Generate build files (subset AWS services):** To build only a particular service or services package(s) for the SDK, add the CMake [BUILD_ONLY](#) parameter. The following example builds only the Amazon S3 service package:

```
cmake ..\aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DBUILD_ONLY="s3" -DCMAKE_PREFIX_PATH="C:\Program Files (x86)\aws-cpp-sdk-all"
```

- **Generate build files (with curl):** After completing the curl prerequisites, three additional cmake command line options are required to include curl support in the SDK: [FORCE_CURL](#), [CURL_INCLUDE_DIR](#), and [CURL_LIBRARY](#). For example:

```
cmake ..\aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DFORCE_CURL=ON -  
DCURL_INCLUDE_DIR='C:/curl/include'  
-DCURL_LIBRARY='C:/curl/lib/libcurl.dll.a' -DCMAKE_PREFIX_PATH="C:\Program  
Files (x86)\aws-cpp-sdk-all"
```

Note

If you get an error Failed to build third-party libraries, check your version of CMake by running **cmake --version**. You must use CMake minimum version 3.13.

6. Build the SDK binaries. If you're building the entire SDK, this step can take one hour or longer. Command syntax:

```
{path to cmake if not in PATH} --build . --config=[Debug | Release]
```

```
cmake --build . --config=Debug
```

Note

If you encounter the error The code execution cannot proceed ... dll not found. Reinstalling the program may fix this problem.", retry the cmake command again.

7. Open a command prompt with **administrator** privileges to install the SDK in the location specified earlier using the CMAKE_PREFIX_PATH parameter. Command syntax:

```
{path to cmake if not in PATH} --install . --config=[Debug | Release]
```

```
cmake --install . --config=Debug
```


Building for Android on Windows

To build for Android, add `-DTARGET_ARCH=ANDROID` to your cmake command line. The AWS SDK for C++ includes a CMake toolchain file that includes what you need by referencing the appropriate environment variables (`ANDROID_NDK`).

To build the SDK for Android on Windows, you need to run cmake from a Visual Studio (2015 or later) developer command prompt. You'll also need NMAKE [NMAKE](#) installed and the commands `git` and `patch` in your path. If you have git installed on a Windows system, you'll most likely find `patch` in a sibling directory (`../Git/usr/bin/`). Once you've verified these requirements, your cmake command line will change slightly to use NMAKE.

```
cmake -G "NMake Makefiles" ` -DTARGET_ARCH=ANDROID ` <other options> ..
```

NMAKE builds serially. To build more quickly, we recommend you install JOM as an alternative to NMAKE, and then change the cmake invocation as follows:

```
cmake -G "NMake Makefiles JOM" ` -DTARGET_ARCH=ANDROID ` <other options> ..
```

For an example application, see [Setting up an Android application with AWS SDK for C++](#)

Build the AWS SDK for C++ on Linux/macOS

To set up the AWS SDK for C++, you can either build the SDK yourself directly from the source or download the libraries using a package manager.

The SDK source is separated into individual packages by service. Installing the entire SDK can take up to an hour. Installing only the specific subset of services that your program uses decreases installation time and also reduces size on disk. To choose which services to install, you need to know the package name of each service your program uses. You can see the list of package directories at [aws/aws-sdk-cpp](#) on GitHub. The package name is the suffix of the directory name for the service.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName> # Repo directory name and packageName  
aws-sdk-cpp\aws-cpp-sdk-s3 # Example: Package name is s3
```

Prerequisites

You need a minimum of 4 GB of RAM to build some of the larger AWS clients. The SDK might fail to build on Amazon EC2 instance types *t2.micro*, *t2.small*, and other small instance types due to insufficient memory.

To use the AWS SDK for C++, you need one of the following:

- GNU Compiler Collection (GCC) 4.9 or later, or
- Clang 3.3 or later.

Additional Requirements for Linux Systems

You must have the header files (-dev packages) for `libcurl`, `libopenssl`, `libuuid`, `zlib`, and, optionally, `libpulse` for Amazon Polly support. You can find the packages by using your system's package manager.

To install the packages on *Debian/Ubuntu-based systems*

- ```
sudo apt-get install libcurl4-openssl-dev libssl-dev uuid-dev zlib1g-dev libpulse-dev
```

### To install the packages on *Amazon Linux/Redhat/Fedora/CentOS-based systems*

- ```
sudo yum install libcurl-devel openssl-devel libuuid-devel pulseaudio-libs-devel
```

Building the SDK from Source

You can build the SDK from source using command-line tools as an alternative to using `vcpkg`. Using this method, you can customize your SDK build. For information about available options, see [CMake Parameters](#).

To build the SDK from source

1. Install [CMake](#) (minimum version 3.13) and the relevant build tools for your platform. It is recommended to add `cmake` to your `PATH`. To check your version of CMake, open a command prompt and run command `cmake --version`

2. In a command prompt, navigate to a folder where you want to store the SDK.
3. Get the latest source code.

Version 1.11 uses git submodules to wrap external dependencies. This includes the [CRT libraries](#) described in the *AWS SDKs and Tools Reference Guide*.

Download or clone the SDK source from [aws/aws-sdk-cpp](#) on GitHub:

- Clone with Git: HTTPS

```
git clone --recurse-submodules https://github.com/aws/aws-sdk-cpp
```

- Clone with Git: SSH

```
git clone --recurse-submodules git@github.com:aws/aws-sdk-cpp.git
```

4. We recommend you store the generated build files outside of the SDK source directory. Create a new directory to store the build files in and navigate to that folder.

```
mkdir sdk_build  
cd sdk_build
```

5. Generate the build files by running cmake. Specify on the cmake command line whether to build a *Debug* or *Release* version. Choose Debug throughout this procedure to run a debug configuration of your application code. Choose Release throughout this procedure to run a release configuration of your application code. Command syntax:

```
{path to cmake if not in PATH} {path to source location of aws-sdk-cpp} -DCMAKE_BUILD_TYPE=[Debug | Release] -DCMAKE_PREFIX_PATH={path to install} -DCMAKE_INSTALL_PREFIX={path to install}
```

For more ways to modify the build output, see [CMake Parameters](#).

To generate the build files, do one of the following:

- **Generate build files (all AWS services):** To build the entire SDK, run cmake, specifying whether to build a *Debug* or *Release* version. For example:

```
cmake ../aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH=/usr/local/ -  
DCMAKE_INSTALL_PREFIX=/usr/local/
```

- **Generate build files (subset AWS services):** To build only a particular service or services package(s) for the SDK, add the CMake [BUILD_ONLY](#) parameter. The following example builds only the Amazon S3 service package:

```
cmake ../aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH=/usr/local/ -  
DCMAKE_INSTALL_PREFIX=/usr/local/ -DBUILD_ONLY="s3"
```

Note

If you get an error Failed to build third-party libraries, check your version of CMake by running `cmake --version`. You must use CMake minimum version 3.13.

6. Build the SDK binaries. If you're building the entire SDK, the operation can take one hour or longer.

```
make
```

7. Install the SDK. You may need to escalate privileges depending on the location you chose to install to.

```
make install
```

Building for Android on Linux

To build for Android, add `-DTARGET_ARCH=ANDROID` to your `cmake` command line. The AWS SDK for C++ includes a CMake toolchain file that includes what you need by referencing the appropriate environment variables (`ANDROID_NDK`). For an example application, see [Setting up an Android application with AWS SDK for C++](#)

"Hello S3" application

[CMake](#) is a build tool that you use to manage your application's dependencies and to create makefiles suitable for the platform you're building on. You can use CMake to create and build projects using the AWS SDK for C++.

This example reports the Amazon S3 buckets you own. Having an Amazon S3 bucket in your AWS account is not required for this example, but it will be far more interesting if you have at least one. See [Create a Bucket](#) in the *Amazon Simple Storage Service User Guide* if you don't already have one.

Step 1: Write the code

This example consists of one folder containing one source file (`hello_s3.cpp`) and one `CMakeLists.txt` file. The program uses Amazon S3 to report storage bucket information. This code is also available in the [AWS Code Examples Repository](#) on GitHub.

You can set many options in a `CMakeLists.txt` build configuration file. For more information, see the [CMake tutorial](#) on the CMake website.

Note

Deep Dive: Setting CMAKE_PREFIX_PATH

By default, the AWS SDK for C++ on macOS, Linux, Android and other non-Windows platforms is installed into `/usr/local` and on Windows is installed into `\Program Files (x86)\aws-cpp-sdk-all`.

CMake needs to know where to find several resources that result from building the SDK ([Windows](#), [Linux/macOS](#)):

- the file `AWSSDKConfig.cmake` so that it can properly resolve the AWS SDK libraries that your application uses.
- (for version 1.8 and earlier) the location of dependencies: `aws-c-event-stream`, `aws-c-common`, `aws-checksums`

Note

Deep Dive: Windows Runtime Libraries

To run your program, several DLLs are required in your program's executable location: `aws-c-common.dll`, `aws-c-event-stream.dll`, `aws-checksums.dll`, `aws-cpp-sdk-core.dll`, as well as any specific DLLs based on the components of your program (this example also requires `aws-cpp-sdk-s3` because it uses Amazon S3). The second `if` statement in the `CMakeLists.txt` file copies these libraries from the installation location to the executable location to satisfy this requirement. `AWSSDK_CPY_DYN_LIBS` is a macro defined by AWS SDK for C++ that copies the SDK's DLLs from the installation

location to the executable location of your program. If these DLLs are not in the executable location then runtime exceptions of 'file not found' occur. Review this portion of the `CMakeLists.txt` file for necessary changes for your unique environment if you encounter these errors.

To create the folder and source files

1. Create a `hello_s3` directory and/or project to hold your source files.

Note

To complete this example in Visual Studio: choose **Create New Project** and then choose **CMake Project**. Name the project `hello_s3`. This project name is used in the `CMakeLists.txt` file.

2. Within that folder, add a `hello_s3.cpp` file that includes the following code, which reports the Amazon S3 buckets you own.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
```

```

Aws::InitAPI(options); // Should only be called once.
int result = 0;
{
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    // You don't normally have to test that you are authenticated. But the S3
    // service permits anonymous requests, thus the s3Client will return "success" and 0
    // buckets even if you are unauthenticated, which can be confusing to a new user.
    auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
    auto creds = provider->GetAWSCredentials();
    if (creds.IsEmpty()) {
        std::cerr << "Failed authentication" << std::endl;
    }

    Aws::S3::S3Client s3Client(clientConfig);
    auto outcome = s3Client.ListBuckets();

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = 1;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size()
            << " buckets\n";
        for (auto &bucket: outcome.GetResult().GetBuckets()) {
            std::cout << bucket.GetName() << std::endl;
        }
    }
}

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

```

3. Add a `CMakeLists.txt` file that specifies your project's name, executables, source files, and linked libraries.

```

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

# Set the AWS service components used by this project.

```

```
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
  hello_s3.cpp)

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


Step 2: Build with CMake

CMake uses the information in `CMakeLists.txt` to build an executable program.

We recommend building the application following standard practices for your IDE.

To build the application from the command line

1. Create a directory where `cmake` will build your application.

```
mkdir my_project_build
```

2. Change to the build directory and run `cmake` using the path to your project's source directory.

```
cd my_project_build  
cmake ../
```

3. After `cmake` generates your build directory, you can use `make` (or `nmake` on Windows), or MSBUILD (`msbuild ALL_BUILD.vcxproj` or `cmake --build . --config=Debug`) to build your application.

Step 3: Run

When you run this application, it displays console output that lists the total number of Amazon S3 buckets and the name of each bucket.

We recommend running the application following standard practices for your IDE.

Note

Remember to sign in! If you are using IAM Identity Center to authenticate, remember to sign in using the AWS CLI `aws sso login` command.

To run the program via command line

1. Change to the Debug directory where the result of the build was generated.
2. Run the program using the name of the executable.

```
hello_s3
```

For additional examples using the AWS SDK for C++, see [Code examples with guidance for the AWS SDK for C++](#).

Get the AWS SDK for C++ from a package manager

Important

If you are using a package manager like homebrew or vcpkg:
After updating the SDK for C++ to a new version, you must recompile any library or executable that depends on the SDK.

To set up the AWS SDK for C++, you can either build the SDK yourself directly from the source or download the libraries using a package manager.

The SDK source is separated into individual packages by service. Installing the entire SDK can take up to an hour. Installing only the specific subset of services that your program uses decreases installation time and also reduces size on disk. To choose which services to install, you need to know the package name of each service your program uses. You can see the list of package directories at [aws/aws-sdk-cpp](#) on GitHub. The package name is the suffix of the directory name for the service.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName> # Repo directory name and packageName  
aws-sdk-cpp\aws-cpp-sdk-s3           # Example: Package name is s3
```

Prerequisites

You need a minimum of 4 GB of RAM to build some of the larger AWS clients. The SDK might fail to build on Amazon EC2 instance types *t2.micro*, *t2.small*, and other small instance types due to insufficient memory.

Linux/macOS

To use the AWS SDK for C++ on Linux/macOS, you need one of the following:

- GNU Compiler Collection (GCC) 4.9 or later, or
- Clang 3.3 or later.

Windows

To use the AWS SDK for C++ on Windows, you need one of the following:

- Microsoft Visual Studio 2015 or later,
- GNU Compiler Collection (GCC) 4.9 or later, or
- Clang 3.3 or later.

Get the SDK using vcpkg

Important

The available vcpkg distribution is supported by external contributors and is not provided through AWS. The most recent version is always available through [installing from source](#).

[vcpkg](#) is a package manager updated and maintained by external contributors. Note that this package manager is not provided through AWS and may not reflect the latest available version for the AWS SDK for C++. There is a delay between when a version is released by AWS and when it is available through an external package manager. The most recent version is always available through [installing from source](#).

You must install [vcpkg](#) on your system.

- Download and bootstrap [vcpkg](#) by following the instructions on the vcpkg GitHub Readme, substituting the following options when prompted:

- As part of those instructions, you are guided to enter:

```
.\vcpkg\vcpkg install [packages to install]
```

To install the entire SDK, enter `.\vcpkg\vcpkg install "aws-sdk-cpp[*]" --recurse` or indicate only specific services of the SDK to install by appending a package name in brackets, for example, `.\vcpkg\vcpkg install "aws-sdk-cpp[s3, ec2]" --recurse`

The output displays a messages including the following:

```
CMake projects should use: "-DCMAKE_TOOLCHAIN_FILE=C:/dev/vcpkg/vcpkg/scripts/buildsystems/vcpkg.cmake"
```

- Copy the complete `-DCMAKE_TOOLCHAIN_FILE` command to use for CMake later. The vcpkg GitHub Readme also instructs on where to use this for your toolset.
- You may also need to note the build configuration type that you installed via vcpkg. The console output shows the build configuration and the version of the SDK. The following example output indicates the build configuration is "x86-windows" and the AWS SDK for C++ version installed is 1.8.

```
The following packages will be built and installed:  
aws-sdk-cpp[core,dynamodb,kinesis,s3]:x86-windows -> 1.8.126#6
```

After you install the AWS SDK for C++, you can develop your own application using the SDK. The ["Hello, S3" starter application](#) reports the Amazon S3 buckets you own.

Troubleshoot build issues

Topics

- [CMake Error: Could not find a package configuration file provided by "AWSSDK"](#)
- [CMake Error: Could not find load file \(and you're on SDK version 1.8\)](#)
- [CMake Error: Could not find load file](#)
- [Runtime Error: cannot proceed because aws-*.dll was not found](#)

CMake Error: Could not find a package configuration file provided by "AWSSDK"

CMake raises the following error if it cannot find the installed SDK.

```
1> [CMake] CMake Error at C:\CodeRepos\CMakeProject1\CMakeLists.txt:4 (find_package):  
1> [CMake]   Could not find a package configuration file provided by "AWSSDK" with any  
1> [CMake]   of the following names:  
1> [CMake]
```

```
1> [CMake]     AWSSDKConfig.cmake
1> [CMake]     awssdk-config.cmake
1> [CMake]
1> [CMake]     Add the installation prefix of "AWSSDK" to CMAKE_PREFIX_PATH or set
1> [CMake]     "AWSSDK_DIR" to a directory containing one of the above files.  If
    "AWSSDK"
1> [CMake]     provides a separate development package or SDK, be sure it has been
1> [CMake]     installed.
```

To resolve this error, tell CMake where to find the installed SDK (e.g. the folder that was generated as a result of the SDK install ([Windows](#), [Linux/macOS](#))). Insert the following command before your first call to `find_package()` in your `CMakeLists.txt` file. See [???](#) for an example.

```
list(APPEND CMAKE_PREFIX_PATH "C:\\Program Files (x86)\\aws-cpp-sdk-all\\lib\\cmake")
```

CMake Error: Could not find load file (and you're on SDK version 1.8)

CMake raises the following error if it cannot find the installed libraries.

```
1> [CMake]     include could not find load file:
1> [CMake]
1> [CMake]     C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-common/cmake/static/
aws-c-common-targets.cmake

1> [CMake]     include could not find load file:
1> [CMake]
1> [CMake]     C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-checksums/cmake/static/
aws-checksums-targets.cmake
1> [CMake]     include could not find load file:
1> [CMake]
1> [CMake]     C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-checksums/cmake/static/
aws-checksums-targets.cmake
```

To resolve this error, tell CMake where to find the installed SDK (e.g. the folder that was generated as a result of the SDK install ([Windows](#), [Linux/macOS](#))). Insert the following commands before your first call to `find_package()` in your `CMakeLists.txt` file. See [???](#) for an example.

```
#Set the location of where Windows can find the installed libraries of the SDK.
if(MSVC)
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif()
```

This solution is only for v1.8 of the SDK because these dependencies are handled differently in later versions. Version 1.9 addresses these issues by introducing an intermediate layer between the `aws-sdk-cpp` and `aws-c-*` libraries. This new layer is called `aws-crt-cpp`, and is a git submodule of the SDK for C++. `aws-crt-cpp` also has the `aws-c-*` libraries (including `aws-c-common`, `aws-checksums`, `aws-c-event-stream`, etc.) as its own git submodules. This allows the SDK for C++ to get all CRT libraries recursively and improves the build process.

CMake Error: Could not find load file

CMake raises the following error if it cannot find the installed libraries.

```
CMake Error at C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-auth/cmake/aws-c-auth-config.cmake:11
  (include): include could not find load file:
  C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-auth/cmake/static/aws-c-auth-targets.cmake
```

To resolve this error, tell CMake to build shared libraries. Insert the following command before your first call to `find_package()` in your `CMakeLists.txt` file. See [???](#) for an example.

```
set(BUILD_SHARED_LIBS ON CACHE STRING "Link to shared libraries by default.")
```

Runtime Error: cannot proceed because aws-* .dll was not found

CMake raises an error similar to the following if it cannot find a required DLL.

```
The code execution cannot proceed because aws-cpp-sdk-[dynamodb].dll was not found.
Reinstalling the program may fix this problem.
```

This error occurs because the required libraries or executables for the SDK for C++ are not available in the same folder as your application executables. To resolve this error, copy the SDK build output in your executable location. The specific DLL filename of the error will vary depending on which AWS services you are using. Do *one* of the following:

- Copy the contents of the `/bin` folder of the AWS SDK for C++ install to your application's build folder.

- In your `CMakeLists.txt` file, use macro `AWSSDK_CPY_DYN_LIBS` to copy these for you.

Add a call to either `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST ""
${CMAKE_CURRENT_BINARY_DIR})` or `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST ""
${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_BUILD_TYPE})` to your `CMakeLists.txt` file to use this macro to do the copying for you. See [???](#) for an example.

Choose the correct copy path for your build environment. Building via command line frequently puts the build output into a subfolder (`/Debug`), but Visual Studio and other IDEs often do not. Verify where your output executables are, and ensure the macro is copying to that location. When making these types of changes, it is good practice to delete the contents of your build output directory so that you get a clean starting point for the next build.

Configure the AWS SDK for C++

This section presents information about how to configure the AWS SDK for C++.

Topics

- [CMake parameters](#)
- [Override your HTTP client](#)
- [Control iostreams used by the HttpClient and the AWSClient](#)

CMake parameters

Use the [CMake](#) parameters listed in this section to customize how your SDK builds.

You can set these options with CMake GUI tools or the command line by using *-D*. For example:

```
cmake -DENABLE_UNITY_BUILD=ON -DREGENERATE_CLIENTS=1
```

General CMake Variables and Options

The following are general **cmake** variables and options that affect the build process of the SDK source code.

Note

Use these parameters when building the SDK source code for the SDK for C++ itself.

Topics

- [ADD_CUSTOM_CLIENTS](#)
- [AUTORUN_UNIT_TESTS](#)
- [AWS_AUTORUN_LD_LIBRARY_PATH](#)
- [AWS_SDK_WARNINGS_ARE_ERRORS](#)
- [AWS_USE_CRYPTO_SHARED_LIBS](#)

- [AWS_TEST_REGION](#)
- [BUILD_BENCHMARKS](#)
- [BUILD_DEPS](#)
- [BUILD_ONLY](#)
- [BUILD_OPTEL](#)
- [BUILD_SHARED_LIBS](#)
- [BYPASS_DEFAULT_PROXY](#)
- [CPP_STANDARD](#)
- [CURL_INCLUDE_DIR](#)
- [CURL_LIBRARY](#)
- [CUSTOM_MEMORY_MANAGEMENT](#)
- [DISABLE_INTERNAL_IMDSV1_CALLS](#)
- [ENABLE_ADDRESS_SANITIZER](#)
- [ENABLE_CURL_LOGGING](#)
- [ENABLE_HTTP_CLIENT_TESTING](#)
- [ENABLE_RTTI](#)
- [ENABLE_TESTING](#)
- [ENABLE_UNITY_BUILD](#)
- [ENABLE_VIRTUAL_OPERATIONS](#)
- [ENABLE_ZLIB_REQUEST_COMPRESSION](#)
- [FORCE_CURL](#)
- [FORCE_SHARED_CRT](#)
- [G](#)
- [MINIMIZE_SIZE](#)
- [NO_ENCRYPTION](#)
- [NO_HTTP_CLIENT](#)
- [REGENERATE_CLIENTS](#)
- [REGENERATE_DEFAULTS](#)

- [SIMPLE_INSTALL](#)
- [TARGET_ARCH](#)
- [USE_CRT_HTTP_CLIENT](#)
- [USE_IXML_HTTP_REQUEST_2](#)
- [USE_OPENSSL](#)
- [USE_TLS_V1_2](#)
- [USE_TLS_V1_3](#)

ADD_CUSTOM_CLIENTS

Builds any arbitrary clients based on the API definition. Place your definition in the code-generation/api-definitions folder, and then pass this argument to **cmake**. The **cmake** configure step generates your client and includes it as a subdirectory in your build. This is particularly useful to generate a C++ client for using one of your [API Gateway](#) services. For example:

```
-  
DADD_CUSTOM_CLIENTS="serviceName=myCustomService,version=2015-12-21;serviceName=someOtherService"
```

Note

To use the `ADD_CUSTOM_CLIENTS` parameter, you must have [Python 2.7](#), Java ([JDK 1.8+](#)), and [Maven](#) installed and in your `PATH`.

AUTORUN_UNIT_TESTS

If `ON`, run unit tests automatically after building.

Values

`ON` | `OFF`

Default

`ON`

AWS_AUTORUN_LD_LIBRARY_PATH

The path to append into LD_LIBRARY_PATH for unit tests autorun by CMake. Set this path if custom runtime libraries are required for overridden dependencies.

Values

String.

Default

N/A

AWS_SDK_WARNINGS_ARE_ERRORS

If ON, treat compiler warnings as errors. Try turning this OFF if observing errors on a new or uncommon compiler.

Values

ON | OFF

Default

ON

AWS_USE_CRYPTOSHARED_LIBS

Forces FindCrypto to use a shared crypto library if found. Turn this OFF to use [BUILD_SHARED_LIBS](#)'s setting instead.

Values

ON | OFF

Default

OFF

AWS_TEST_REGION

The AWS Region to use for integration tests.

Values

String.

Default

N/A

BUILD_BENCHMARKS

If ON, build the benchmark executable.

Values

ON | OFF

Default

OFF

BUILD_DEPS

If ON, build third-party dependencies.

Values

ON | OFF

Default

ON

BUILD_ONLY

Builds only the clients you want to use. If set to a high-level SDK such as `aws-cpp-sdk-transfer`, `BUILD_ONLY` resolves any low-level client dependencies. It also builds integration and unit tests related to the projects you select, if they exist. This is a list argument, with values separated by semicolon (;) characters. For example:

```
-DBUILD_ONLY="s3;cognito-identity"
```

Note

The core SDK module, `aws-sdk-cpp-core`, is *always* built, regardless of the value of the `BUILD_ONLY` parameter.

BUILD_OPTEL

If ON, builds the open telemetry implementation of tracing.

Values

`ON` | `OFF`

Default

`OFF`

BUILD_SHARED_LIBS

A built-in CMake option, re-exposed here for visibility. If ON, it builds shared libraries; otherwise, it builds only static libraries.

Note

To dynamically link to the SDK, you must define the `USE_IMPORT_EXPORT` symbol for all build targets using the SDK.

Values

`ON` | `OFF`

Default

`ON`

BYPASS_DEFAULT_PROXY

If ON, bypass the machine's default proxy settings when using `IXmlHttpRequest2`.

Values

ON | OFF

Default

ON

CPP_STANDARD

Specifies a custom C++ standard for use with C++ 14 and 17 code bases.

Values

11 | 14 | 17

Default

11

CURL_INCLUDE_DIR

Path to curl include directory containing `libcurl` headers.

Values

String path to selected include directory. For example, `D:/path/to/dir/with/curl/include`.

Default

N/A

CURL_LIBRARY

Path to curl library file to link against. This library can be a static library or an import library, depending on the needs of your application.

Values

String path to the curl library file. For example, `D:/path/to/static/libcurl/file/ie/libcurl.lib.a`.

Default

N/A

CUSTOM_MEMORY_MANAGEMENT

To use a custom memory manager, set the value to 1. You can install a custom allocator so that all STL types use the custom allocation interface. If you set the value 0, you still might want to use the STL template types to help with DLL safety on Windows.

If static linking is ON, custom memory management defaults to *off* (0). If dynamic linking is ON, custom memory management defaults to *on* (1) and avoids cross-DLL allocation and deallocation.

Note

To prevent linker mismatch errors, you must use the same value (0 or 1) throughout your build system.

To install your own memory manager to handle allocations made by the SDK, you must set `DCUSTOM_MEMORY_MANAGEMENT` and define `AWS_CUSTOM_MEMORY_MANAGEMENT` for all build targets that depend on the SDK.

DISABLE_INTERNAL_IMDSV1_CALLS

If ON, no internal calls are made to the V1 API of the [Instance Metadata Service](#).

Values

ON | OFF

Default

OFF

ENABLE_ADDRESS_SANITIZER

If ON, turns on Address Sanitizer for gcc or clang.

Values

ON | OFF

Default

OFF

ENABLE_CURL_LOGGING

If ON, pipe the internal log for curl to the SDK logger.

Values

ON | OFF

Default

OFF

ENABLE_HTTP_CLIENT_TESTING

If ON, build and run corresponding HTTP client test suites.

Values

ON | OFF

Default

OFF

ENABLE_RTTI

Controls whether the SDK is built to enable run-time type information (RTTI).

Values

ON | OFF

Default

ON

ENABLE_TESTING

Controls whether unit and integration test projects are built during the SDK build.

Values

ON | *OFF*

Default

ON

ENABLE_UNITY_BUILD

If ON, most SDK libraries are built as a single, generated .cpp file. This can significantly reduce static library size and speed up compilation time.

Values

ON | *OFF*

Default

OFF

ENABLE_VIRTUAL_OPERATIONS

This parameter usually works together with REGENERATE_CLIENTS for code generation.

If ENABLE_VIRTUAL_OPERATIONS is ON and REGENERATE_CLIENTS is ON, operation-related functions in service clients will be marked as `virtual`.

If ENABLE_VIRTUAL_OPERATIONS is OFF and REGENERATE_CLIENTS is ON, `virtual` won't be added to operation functions and service client classes will be marked as `final`.

If ENABLE_VIRTUAL_OPERATIONS is OFF, the SDK will also add `-ffunction-sections` and `-fdata-sections` compiler flags for gcc and clang when compiling.

For more information, see [CMake Parameters](#) on GitHub.

Values

ON | *OFF*

Default

ON

ENABLE_ZLIB_REQUEST_COMPRESSION

For services that support it, request content will be compressed. On by default if dependency is available.

Values

ON | OFF

Default

ON

FORCE_CURL

Windows only. If *ON*, forces usage of the curl client instead of the default [WinHTTP](#) data transfer provider.

Values

ON | OFF

Default

OFF

FORCE_SHARED_CRT

If *ON*, the SDK links to the C runtime *dynamically*; otherwise, it uses the *BUILD_SHARED_LIBS* setting (sometimes necessary for backward compatibility with earlier versions of the SDK).

Values

ON | OFF

Default

ON

G

Generates build artifacts, such as Visual Studio solutions and Xcode projects.

For example, on Windows:

```
-G "Visual Studio 12 Win64"
```

For more information, see the CMake documentation for your platform.

MINIMIZE_SIZE

A superset of [ENABLE_UNITY_BUILD](#). If ON, this option turns on *ENABLE_UNITY_BUILD* and additional binary size reduction settings.

Values

ON | *OFF*

Default

OFF

NO_ENCRYPTION

If ON, prevents the default platform-specific cryptography implementation from being built into the library. Turn this *ON* to inject your own cryptography implementation.

Values

ON | *OFF*

Default

OFF

NO_HTTP_CLIENT

If ON, prevents the default platform-specific HTTP client from being built into the library. If *ON*, you will need to provide your own platform-specific HTTP client implementation.

Values

ON | OFF

Default

OFF

REGENERATE_CLIENTS

If ON, this parameter deletes all generated code and generates the client directories from the `code-generation/api-definitions` folder. For example:

```
-DREGENERATE_CLIENTS=1
```

Note

To use the `REGENERATE_CLIENTS` parameter, you must have [Python 2.7](#), Java ([JDK 1.8+](#)), and [Maven](#) installed and in your `PATH`.

REGENERATE_DEFAULTS

If ON, this parameter deletes all generated defaults code and generates it again from the `code-generation/defaults` folder. For example:

```
-DREGENERATE_DEFAULTS=1
```

Note

To use the `REGENERATE_DEFAULTS` parameter, you must have [Python 2.7](#), Java ([JDK 1.8+](#)), and [Maven](#) installed and in your `PATH`.

SIMPLE_INSTALL

If ON, the install process does not insert platform-specific intermediate directories underneath `bin/` and `lib/`. Turn OFF if you need to make multiplatform releases under a single install directory.

Values

ON | OFF

Default

ON

TARGET_ARCH

To cross-compile or build for a mobile platform, you must specify the target platform. By default, the build detects the host operating system and builds for the detected operating system.

Note

When *TARGET_ARCH* is *ANDROID*, additional options are available. See [Android CMake Variables and Options](#).

Values

WINDOWS | LINUX | APPLE | ANDROID

USE_CRT_HTTP_CLIENT

If *ON*, use the common runtime HTTP client, and the legacy systems such as WinHttp and libcurl are not built or included.

Values

ON | OFF

Default

OFF

USE_IXML_HTTP_REQUEST_2

Windows only. If *ON*, use the com object *IXmlHttpRequest2* for the HTTP stack.

Values

ON | OFF

Default

OFF

USE_OPENSSL

If ON, the SDK builds using OpenSSL; otherwise, it uses [awslabs/aws-lc](https://github.com/aws/aws-lc). AWS-LC is a general-purpose cryptographic library maintained by the AWS Cryptography team for AWS and their customers. Turning OFF the parameter installs AWS-LC as replacement of OpenSSL in the system default directory. Don't use if you already have an OpenSSL installation in your system.

Values

ON | OFF

Default

ON

USE_TLS_V1_2

If ON, the HTTP client enforces TLS 1.2.

Values

ON | OFF

Default

ON

USE_TLS_V1_3

If ON, the HTTP client enforces TLS 1.3.

Values

ON | OFF

Default

OFF

Android CMake Variables and Options

Use the following variables when you are creating an Android build of the SDK (when [TARGET_ARCH](#) is set to *ANDROID*).

Topics

- [ANDROID_ABI](#)
- [ANDROID_BUILD_CURL](#)
- [ANDROID_BUILD_OPENSSL](#)
- [ANDROID_BUILD_ZLIB](#)
- [ANDROID_NATIVE_API_LEVEL](#)
- [ANDROID_STL](#)
- [ANDROID_TOOLCHAIN_NAME](#)
- [DISABLE_ANDROID_STANDALONE_BUILD](#)
- [NDK_DIR](#)

ANDROID_ABI

Android only. Controls which Application Binary Interface (ABI) to output code for.

Note

Not all valid Android ABI values are currently supported.

Values

arm64 | armeabi-v7a | x86_64 | x86 | mips64 | mips

Default

armeabi-v7a

ANDROID_BUILD_CURL

Android only. If ON, build curl also.

Values

ON | *OFF*

Default

ON

ANDROID_BUILD_OPENSSL

Android only. If ON, build Openssl also.

Values

ON | *OFF*

Default

ON

ANDROID_BUILD_ZLIB

Android only. If ON, build Zlib also.

Values

ON | *OFF*

Default

ON

ANDROID_NATIVE_API_LEVEL

Android only. Controls what API level the SDK builds against. If you set [ANDROID_STL](#) to *gnustl*, you can choose any API level. If you use *libc++*, you must use an API level of at least 21.

Default

Varies by STL choice.

ANDROID_STL

Android only. Controls what flavor of the C++ standard library the SDK uses.

Important

Performance problems can occur within the SDK if the `gnustl` options are used; we strongly recommend using `libc++_shared` or `libc++_static`.

Values

`libc++_shared` | `libc++_static` | `gnustl_shared` | `gnustl_static`

Default

`libc++_shared`

ANDROID_TOOLCHAIN_NAME

Android only. Controls which compiler is used to build the SDK.

Note

With GCC being deprecated by the Android NDK, we recommend using the default value.

Default

`standalone-clang`

DISABLE_ANDROID_STANDALONE_BUILD

Android only. By default, Android builds use a standalone clang-based toolchain constructed via NDK scripts. To use your own toolchain, turn this option *ON*.

Values

`ON` | `OFF`

Default

OFF

NDK_DIR

Android only. Specifies an override path where the build system should find the Android NDK. By default, the build system checks environment variables (ANDROID_NDK) if this variable is not set.

Override your HTTP client

The default HTTP client for Windows is [WinHTTP](#). The default HTTP client for all other platforms is [curl](#).

Optionally, you can override the HTTP client default by creating a custom `HttpClientFactory` to pass to any service client's constructor. To override the HTTP client, the SDK must be built with curl support. Curl support is built by default in Linux and macOS, but additional steps are required to build on Windows. For more information about building the SDK on Windows with curl support, see [Build the AWS SDK for C++ on Windows](#).

Control iostreams used by the HttpClient and the AWSClient

By default, all responses use an input stream backed by a `stringbuf`. If needed, you can override the default behavior. For example, if you are using an Amazon S3 `GetObject` and don't want to load the entire file into memory, you can use `IOStreamFactory` in `AmazonWebServiceRequest` to pass a lambda to create a file stream.

Example file stream request

```
GetObjectRequest getObjectRequest;
getObjectRequest.SetBucket(fullBucketName);
getObjectRequest.SetKey(keyName);
getObjectRequest.SetResponseStreamFactory([](){
    return Aws::New<Aws::FStream>(
        ALLOCATION_TAG, DOWNLOADED_FILENAME, std::ios_base::out); });

auto getObjectOutcome = s3Client->GetObject(getObjectRequest);
```

Use the SDK

This section provides information about general use of the AWS SDK for C++, beyond that covered in [Getting Started Using the AWS SDK for C++](#).

For service-specific programming examples, see [AWS SDK for C++ Code Examples](#).

Topics

- [Basic Use](#)
- [AWS Client configuration](#)
- [Service Client Classes](#)
- [Utility Modules](#)
- [Memory Management](#)
- [Logging](#)
- [Error Handling](#)
- [Using a custom libcrypto library](#)

Basic Use

Initializing and Shutting Down the SDK

Applications that use the AWS SDK for C++ must initialize it. Similarly, before the application terminates, the SDK must be shut down. Both operations accept configuration options that affect the initialization and shutdown processes and subsequent calls to the SDK.

All applications that use the AWS SDK for C++ must include the file `aws/core/Aws.h`.

The AWS SDK for C++ must be initialized by calling `Aws::InitAPI`. Before the application terminates, the SDK must be shut down by calling `Aws::ShutdownAPI`. Each method accepts an argument of [Aws::SDKOptions](#). All other calls to the SDK can be performed between these two method calls.

All AWS SDK for C++ calls performed between `Aws::InitAPI` and `Aws::ShutdownAPI` should either to be contained within a pair of curly braces or should be invoked by functions called between the two methods.

A basic skeleton application is shown below.

```
#include <aws/core/Aws.h>
int main(int argc, char** argv)
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        // make your SDK calls here.
    }
    Aws::ShutdownAPI(options);
    return 0;
}
```

The SDK for C++ and its dependencies use C++ static objects, and the order of static object destruction is not determined by the C++ standard. To avoid memory issues caused by the nondeterministic order of static variable destruction, do not wrap the calls to `Aws::InitAPI` and `Aws::ShutdownAPI` into another static object.

Setting SDK Options

The [Aws::SDKOptions](#) struct contains SDK configuration options.

An instance of [Aws::SDKOptions](#) is passed to the `Aws::InitAPI` and `Aws::ShutdownAPI` methods. The same instance should be sent to both methods.

The following samples demonstrate some of the available options.

- Turn logging on using the default logger

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Info;
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

- Override the default HTTP client factory

```
Aws::SDKOptions options;
```

```
options.httpOptions.httpClientFactory_create_fn = [](){
    return Aws::MakeShared<MyCustomHttpClientFactory>(
        "ALLOC_TAG", arg1);
};
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

Note

`httpOptions` takes a closure (also called an anonymous function or lambda expression) rather than a `std::shared_ptr`. Each of the SDK factory functions operates in this manner because at the time at which the factory memory allocation occurs, the memory manager has not yet been installed. By passing a closure to the method, the memory manager will be called to perform the memory allocation when it is safe to do so. A simple technique to accomplish this procedure is by using a Lambda expression.

- Use a global SIGPIPE handler

If you build the SDK for C++ with curl and OpenSSL, you must specify a signal handler. If you don't use your own custom signal handler, set `installSigPipeHandler` to `true`.

```
Aws::SDKOptions options;
options.httpOptions.installSigPipeHandler = true;
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

When `installSigPipeHandler` is `true`, the SDK for C++ uses a handler that ignores SIGPIPE signals. For more information on SIGPIPE, see [Operation Error Signals](#) on the GNU Operating System website. For more information on the curl handler, see [CURLOPT_NOSIGNAL explained](#) on the curl website.

The underlying libraries of curl and OpenSSL can send a SIGPIPE signal to notify when the remote side closes a connection. These signals must be handled by the application. For more information this curl functionality, see [libcurl thread safety](#) on the curl website. This behavior is

not automatically built-in to the SDK because signal handlers are global for each application and the library is a dependency for the SDK.

Credential provider chain

The AWS SDKs and CLIs use *provider chains* to look for AWS credentials in several different places, including system/user environment variables and local AWS configuration files. For details on the order of precedence, see [Credentials Providers](#) in the `aws-sdk-cpp` repository in GitHub. For configuration details about other AWS SDK for C++ credential providers, see [Standardized credential providers](#) in the *AWS SDKs and Tools Reference Guide*.

Default AWS Region

You must set a default AWS Region for the AWS SDK for C++ to use for AWS requests. This Region is used for SDK service requests that aren't specified with a Region. The *AWS SDKs and Tools Reference Guide* has additional information on the following:

- This is often set in the [shared AWSconfig file](#), which is a file with a set of configuration values that can be referenced from the SDK.
- To find the location of the shared config file file, see [Location of the shared files](#).
- For more information on the `region` setting, see [AWS Region](#).

AWS Client configuration

The AWS SDK for C++ enables you to change the default client configuration, which is helpful when you want to do things like:

- Connect to the Internet through proxy
- Change HTTP transport settings, such as connection timeout and request retries
- Specify TCP socket buffer size hints

`ClientConfiguration` is a structure in the SDK for C++ that you can instantiate and utilize in your code. The following snippet illustrates using this class to access Amazon S3 through a proxy.

```
Aws::Client::ClientConfiguration clientConfig;
```

```
clientConfig.proxyHost = "localhost";
clientConfig.proxyPort = 1234;
clientConfig.proxyScheme = Aws::Http::Scheme::HTTPS;
Aws::S3::S3Client(clientConfig);
```

ClientConfiguration declaration (See latest at [Aws::Client::ClientConfiguration](#) in the *AWS SDK for C++ API Reference*):

```
struct AWS_CORE_API ClientConfiguration
{
    ClientConfiguration();

    Aws::String userAgent;
    Aws::Http::Scheme scheme;
    Aws::Region region;
    bool useDualStack;
    unsigned maxConnections;
    long requestTimeoutMs;
    long connectTimeoutMs;
    bool enableTcpKeepAlive;
    unsigned long tcpKeepAliveIntervalMs;
    unsigned long lowSpeedLimit;
    std::shared_ptr<RetryStrategy> retryStrategy;
    Aws::String endpointOverride;
    Aws::Http::Scheme proxyScheme;
    Aws::String proxyHost;
    unsigned proxyPort;
    Aws::String proxyUserName;
    Aws::String proxyPassword;
    std::shared_ptr<Aws::Utils::Threading::Executor> executor;
    bool verifySSL;
    Aws::String caPath;
    Aws::String caFile;
    std::shared_ptr<Aws::Utils::RateLimits::RateLimiterInterface> writeRateLimiter;
    std::shared_ptr<Aws::Utils::RateLimits::RateLimiterInterface> readRateLimiter;
    Aws::Http::TransferLibType httpLibOverride;
    Aws::Client::FollowRedirectsPolicy followRedirects;
    bool disableExpectHeader;
    bool enableClockSkewAdjustment;
    bool enableHostPrefixInjection;
    bool enableEndpointDiscovery;
};
```

Configuration Variables

userAgent

For internal use only. Do not change the setting of this variable.

scheme

Specifies the URI addressing scheme, either HTTP or HTTPS. The default scheme is HTTPS.

region

Specifies the AWS Region to use, such as *us-east-1*. By default, the Region used is the default Region configured in the applicable AWS credentials.

useDualStack

Controls whether to use dual stack IPv4 and IPv6 endpoints. Note that not all AWS services support IPv6 in all Regions.

maxConnections

Specifies the maximum number of HTTP connections to a single server. The default value is 25. No maximum allowed value exists other than what your bandwidth can reasonably support.

requestTimeoutMs and connectTimeoutMs

Specifies the amount of time in milliseconds to wait before timing out an HTTP request. For example, consider increasing these times when transferring large files.

enableTcpKeepAlive

Controls whether to send TCP keep-alive packets. The default setting is true. Use in conjunction with the `tcpKeepAliveIntervalMs` variable. This variable is not applicable for WinINet and the IXMLHttpRequest2 client.

tcpKeepAliveIntervalMs

Specifies the time interval in milliseconds at which to send a keep-alive packet over a TCP connection. The default interval is 30 seconds. The minimum setting is 15 seconds. This variable is not applicable for WinINet and the IXMLHttpRequest2 client.

lowSpeedLimit

Specifies the minimum allowed transfer speed in bytes per second. If the transfer speed falls below the specified speed, the transfer operation is aborted. The default setting is 1 byte/second. This variable is applicable only for CURL clients.

retryStrategy

References the implementation of the retry strategy. The default strategy implements an exponential backoff policy. To perform a different strategy, implement a subclass of the `RetryStrategy` class and assign an instance to this variable.

endpointOverride

Specifies an overriding HTTP endpoint with which to communicate with a service.

proxyScheme, proxyHost, proxyPort, proxyUserName, and proxyPassword

Used to set up and configure a proxy for all communications with AWS. Examples of when this functionality might be useful include debugging in conjunction with the Burp suite, or using a proxy to connect to the Internet.

executor

References the implementation of the asynchronous `Executor` handler. The default behavior is to create and detach a thread for each async call. To change this behavior, implement a subclass of the `Executor` class and assign an instance to this variable.

verifySSL

Controls whether to verify SSL certificates. By default SSL certificates are verified. To disable verification, set the variable to false.

caPath, caFile

Instructs the HTTP client where to find your SSL certificate trust store. An example trust store might be a directory prepared with the OpenSSL `c_rehash` utility. These variables should not need to be set unless your environment uses symlinks. These variables have no effect on Windows and macOS systems.

writeRateLimiter and readRateLimiter

References to the implementations of read and write rate limiters which are used to throttle the bandwidth used by the transport layer. By default, the read and write rates are not throttled. To introduce throttling, implement a subclass of the `RateLimiterInterface` and assign an instance to these variables.

httpLibOverride

Specifies the HTTP implementation returned by the default HTTP factory. The default HTTP client for Windows is `WinHTTP`. The default HTTP client for all other platforms is `CURL`.

followRedirects

Controls the behavior when handling HTTP 300 redirect codes.

disableExpectHeader

Applicable only for CURL HTTP clients. By default, CURL adds an "Expect: 100-Continue" header in an HTTP request to avoid sending the HTTP payload in situations where the server responds with an error immediately after receiving the header. This behavior can save a round-trip and is useful in situations where the payload is small and network latency is relevant. The variable's default setting is false. If set to true, CURL is instructed to send both the HTTP request header and body payload together.

enableClockSkewAdjustment

Controls whether clock skew is adjusted after each HTTP attempt. The default setting is false.

enableHostPrefixInjection

Controls whether the HTTP host adds a "data-" prefix to DiscoverInstances requests. By default, this behavior is enabled. To disable it, set the variable to false.

enableEndpointDiscovery

Controls whether endpoint discovery is used. By default, regional or overridden endpoints are used. To enable endpoint discovery, set the variable to true.

Service Client Classes

The AWS SDK for C++ includes client classes that provide interfaces to the AWS services. Each client class supports a particular AWS service. For example, the `S3Client` provides an interface to the Amazon S3 service.

The namespace for a client class follows the convention `Aws::Service::ServiceClient`. For example, the client class for AWS Identity and Access Management (IAM) is `Aws::IAM::IAMClient` and the Amazon S3 client class is `Aws::S3::S3Client`.

All client classes for all AWS services are thread-safe.

When instantiating a client class, AWS credentials must be supplied. For more information about credentials, see [Providing AWS Credentials](#).

Utility Modules

The AWS SDK for C++ includes many [utility modules](#) to reduce the complexity of developing AWS applications in C++.

HTTP Stack

An HTTP stack that provides connection pooling, is thread-safe, and can be reused as you need. For more information, see [AWS Client Configuration](#).

Headers	/aws/core/http/
API Documentation	Aws::Http

String Utils

Core string functions, such as `trim`, `lowercase`, and numeric conversions.

Header	aws/core/utils/StringUtils.h
API Documentation	Aws::Utils::StringUtils

Hashing Utils

Hashing functions such as SHA256, MD5, Base64, and SHA256_HMAC.

Header	/aws/core/utils/HashingUtils.h
API Documentation	Aws::Utils::HashingUtils

JSON Parser

A fully functioning yet lightweight JSON parser (a thin wrapper around *JsonCpp*).

Header	/aws/core/utils/json/JsonSerializer.h
API Documentation	Aws::Utils::Json::JsonValue

XML Parser

A lightweight XML parser (a thin wrapper around *tinyclang*). The [RAII pattern](#) has been added to the interface.

Header	/aws/core/utils/xml/XmlSerializer.h
API Documentation	Aws::Utils::Xml

Memory Management

The AWS SDK for C++ provides a way to control memory allocation and deallocation in a library.

Note

Custom memory management is available only if you use a version of the library built using the defined compile-time constant `AWS_CUSTOM_MEMORY_MANAGEMENT`. If you use a version of the library that is built without the compile-time constant, global memory system functions such as `InitializeAWSMemorySystem` won't work; the global `new` and `delete` functions are used instead.

For more information about the compile-time constant, see [STL and AWS Strings and Vectors](#).

Allocating and Deallocating Memory

To allocate or deallocate memory

1. Subclass `MemorySystemInterface`: `aws/core/utils/memory/MemorySystemInterface.h`.

```
class MyMemoryManager : public Aws::Utils::Memory::MemorySystemInterface
{
public:
    // ...
    virtual void* AllocateMemory(
        std::size_t blockSize, std::size_t alignment,
        const char *allocationTag = nullptr) override;
    virtual void FreeMemory(void* memoryPtr) override;
};
```

Note

You can change the type signature for `AllocateMemory` as needed.

2. Install a memory manager with an instance of your subclass by calling `InitializeAWSMemorySystem`. This should occur at the beginning of your application. For example, in your `main()` function:

```
int main(void)
{
    MyMemoryManager sdkMemoryManager;
    Aws::Utils::Memory::InitializeAWSMemorySystem(sdkMemoryManager);
    // ... do stuff
    Aws::Utils::Memory::ShutdownAWSMemorySystem();
    return 0;
}
```

3. Just before exit, call `ShutdownAWSMemorySystem` (as shown in the preceding example, but repeated here):

```
Aws::Utils::Memory::ShutdownAWSMemorySystem();
```

STL and AWS Strings and Vectors

When initialized with a memory manager, the AWS SDK for C++ defers all allocation and deallocation to the memory manager. If a memory manager doesn't exist, the SDK uses global `new` and `delete`.

If you use custom STL allocators, you must alter the type signatures for all STL objects to match the allocation policy. Because STL is used prominently in the SDK implementation and interface, a single approach in the SDK would inhibit direct passing of default STL objects into the SDK or control of STL allocation. Alternately, a hybrid approach—using custom allocators internally and allowing standard and custom STL objects on the interface—could potentially make it more difficult to investigate memory issues.

The solution is to use the memory system's compile-time constant `AWS_CUSTOM_MEMORY_MANAGEMENT` to control which STL types the SDK uses.

If the compile-time constant is enabled (on), the types resolve to STL types with a custom allocator connected to the AWS memory system.

If the compile-time constant is disabled (off), all `Aws::*` types resolve to the corresponding default `std::*` type.

Example code from the ```AWSAllocator.h``` file in the SDK

```
#ifndef AWS_CUSTOM_MEMORY_MANAGEMENT

template< typename T >
class AwsAllocator : public std::allocator< T >
{
    ... definition of allocator that uses AWS memory system
};

#else

template< typename T > using Allocator = std::allocator<T>;

#endif
```

In the example code, the `AwsAllocator` can be a custom allocator or a default allocator, depending on the compile-time constant.

Example code from the ```AWSVector.h``` file in the SDK

```
template<typename T> using Vector = std::vector<T, Aws::Allocator<T>>;
```

In the example code, we define the `Aws::*` types.

If the compile-time constant is enabled (on), the type maps to a vector using custom memory allocation and the AWS memory system.

If the compile-time constant is disabled (off), the type maps to a regular `std::vector` with default type parameters.

Type aliasing is used for all `std::` types in the SDK that perform memory allocation, such as containers, string streams, and string buffers. The AWS SDK for C++ uses these types.

Remaining Issues

You can control memory allocation in the SDK; however, STL types still dominate the public interface through string parameters to the model object `initialize` and `set` methods. If you don't use STL and use strings and containers instead, you have to create a lot of temporaries whenever you want to make a service call.

To remove most of the temporaries and allocation when you make service calls using non-STL, we have implemented the following:

- Every `Init/Set` function that takes a string has an overload that takes a `const char*`.
- Every `Init/Set` function that takes a container (map/vector) has an `add` variant that takes a single entry.
- Every `Init/Set` function that takes binary data has an overload that takes a pointer to the data and a `length` value.
- (Optional) Every `Init/Set` function that takes a string has an overload that takes a non-zero terminated `const char*` and a `length` value.

Native SDK Developers and Memory Controls

Follow these rules in the SDK code:

- Don't use `new` and `delete`; use `Aws::New<>` and `Aws::Delete<>` instead.
- Don't use `new[]` and `delete[]`; use `Aws::NewArray<>` and `Aws::DeleteArray<>`.
- Don't use `std::make_shared`; use `Aws::MakeShared`.
- Use `Aws::UniquePtr` for unique pointers to a single object. Use the `Aws::MakeUnique` function to create the unique pointer.

- Use `Aws::UniqueArray` for unique pointers to an array of objects. Use the `Aws::MakeUniqueArray` function to create the unique pointer.
- Don't directly use STL containers; use one of the `Aws::` typedefs or add a typedef for the container you want. For example:

```
Aws::Map<Aws::String, Aws::String> m_kvPairs;
```

- Use `shared_ptr` for any external pointer passed into and managed by the SDK. You must initialize the shared pointer with a destruction policy that matches how the object was allocated. You can use a raw pointer if the SDK is not expected to clean up the pointer.

Logging

The AWS SDK for C++ includes configurable logging that generates a record of actions performed by the SDK during execution. To enable logging, set the `LogLevel` of `SDKOptions` to the appropriate verbosity for your application.

```
Aws::SDKOptions options;  
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Info;
```

There are seven levels of verbosity to choose from. The default value is `Off` and no logs will be generated. `Trace` will generate the most level of detail, and `Fatal` will generate the least messages reporting only fatal error conditions.

Once logging is enabled in your application, the SDK will generate log files in your executable directory following the default naming pattern of `aws_sdk_<date>.log`. The log file generated by the prefix-naming option rolls over once per hour to allow for archiving or deleting log files.

The later versions of the SDK increasingly depend on the underlying AWS Common Runtime (CRT) libraries. These libraries provide common functionality and basic operations among SDKs. All log messages from the CRT libraries will be redirected to the SDK for C++ by default. The log level and logging system you specify for the SDK for C++ also applies to the CRT.

In the previous example, the CRT will inherit `LogLevel::Info` and also log messages at the `Info` level to the same file.

You can independently control the logging for the CRT libraries, either by redirecting its output to a separate log file, or by setting a different log level for messages from the CRT. Often it can be

beneficial to reduce the verbosity of the CRT libraries so that they don't overwhelm the logs. For example, the log level for *only* the CRT output can be set to `Warn` as follows:

```
options.loggingOptions.crt_logger_create_fn =
    [](){ return
    Aws::MakeShared<Aws::Utils::Logging::DefaultCRTLogSystem>("CRTLogSystem",
    Aws::Utils::Logging::LogLevel::Warn); };
```

By optionally using the method `InitializeAWSLogging`, you can control the verbosity level and the log output of the `DefaultLogSystem`. You can configure the log filename prefix, or redirect the output to a stream instead of a file.

```
Aws::Utils::Logging::InitializeAWSLogging(
    Aws::MakeShared<Aws::Utils::Logging::DefaultLogSystem>(
        "RunUnitTests", Aws::Utils::Logging::LogLevel::Trace, "aws_sdk_"));
```

Alternatively, instead of using the `DefaultLogSystem`, you can also use this method to provide your own logging implementation.

```
InitializeAWSLogging(Aws::MakeShared<CustomLoggingSystem>());
```

If you call method `InitializeAWSLogging`, free resources at the end of your program by calling `ShutdownAWSLogging`.

```
Aws::Utils::Logging::ShutdownAWSLogging();
```

Example integration test with logging

```
#include <aws/external/gtest.h>

#include <aws/core/utils/memory/stl/AWSString.h>
#include <aws/core/utils/logging/DefaultLogSystem.h>
#include <aws/core/utils/logging/AWSLogging.h>

#include <iostream>

int main(int argc, char** argv)
{
    Aws::Utils::Logging::InitializeAWSLogging(
        Aws::MakeShared<Aws::Utils::Logging::DefaultLogSystem>(
```

```
        "RunUnitTests", Aws::Utils::Logging::LogLevel::Trace, "aws_sdk_"));
    ::testing::InitGoogleTest(&argc, argv);
    int exitCode = RUN_ALL_TESTS();
    Aws::Utils::Logging::ShutdownAWSLogging();
    return exitCode;
}
```

Error Handling

The AWS SDK for C++ does not use exceptions; however, you can use exceptions in your code. Every service client returns an outcome object that includes the result and an error code.

Example of handling error conditions

```
bool CreateTableAndWaitForItToBeActive()
{
    CreateTableRequest createTableRequest;
    AttributeDefinition hashKey;
    hashKey.SetAttributeName(HASH_KEY_NAME);
    hashKey.SetAttributeType(ScalarAttributeType::S);
    createTableRequest.AddAttributeDefinitions(hashKey);
    KeySchemaElement hashKeySchemaElement;
    hashKeySchemaElement.WithAttributeName(HASH_KEY_NAME).WithKeyType(KeyType::HASH);
    createTableRequest.AddKeySchema(hashKeySchemaElement);
    ProvisionedThroughput provisionedThroughput;
    provisionedThroughput.SetReadCapacityUnits(readCap);
    provisionedThroughput.SetWriteCapacityUnits(writeCap);
    createTableRequest.WithProvisionedThroughput(provisionedThroughput);
    createTableRequest.WithTableName(tableName);

    CreateTableOutcome createTableOutcome = dynamoDbClient-
>CreateTable(createTableRequest);
    if (createTableOutcome.IsSuccess())
    {
        DescribeTableRequest describeTableRequest;
        describeTableRequest.SetTableName(tableName);
        bool shouldContinue = true;
        DescribeTableOutcome outcome = dynamoDbClient-
>DescribeTable(describeTableRequest);

        while (shouldContinue)
        {
```

```
        if (outcome.GetResult().GetTable().GetTableStatus() == TableStatus::ACTIVE)
        {
            break;
        }
        else
        {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
    }
    return true;
}
else if(createTableOutcome.GetError().GetErrorType() ==
DynamoDBErrors::RESOURCE_IN_USE)
{
    return true;
}

return false;
}
```

Using a custom libcrypto library

By default, the AWS SDK for C++ uses the default system cryptographic library for transport layer security. However, the SDK for C++ can optionally be configured to use a different libcrypto library when building the SDK from source. This functionally means that all cryptographic operations will be diverted to a custom implementation of OpenSSL. For example, you might want to use the [AWS-LC](#) library in [FIPS mode](#) to achieve a FIPS standard in your application.

How to build a custom libcrypto into the SDK for C++

Step 1: Build or obtain your libcrypto library

The [AWS-LC](#) is one example of an alternative libcrypto library, but any distribution of OpenSSL or OpenSSL-equivalent would work.

The SDK for C++, and its dependency the CRT, both use libcrypto for their cryptographic functions and both need to handle dependencies the same. The SDK for C++ depends on two different HTTP clients depending on if the request uses the SDK's CRT S3 functionality. The CRT specifically is dependent on [s2n](#), a TLS implementation that is initialized at start up time. Both the SDK and the s2n team have a cmake parameter to force the use of a shared libcrypto library regardless of

the value of [BUILD_SHARED_LIBS](#). Typically, you want the CRT HTTP client and the regular HTTP client to use the same libcrypto. In this case, that would mean both referencing OpenSSL in the dependency tree. The SDK provides this via [AWS_USE_CRYPTO_SHARED_LIBS](#) and s2n (for CRT-based calls) provides this through [S2N_USE_CRYPTO_SHARED_LIBS](#). Dependency resolution is the same between these two libraries, and typically these are set to match, though you can explicitly set them to be different.

For example, to use AWS-LC as the libcrypto library, you would build it as follows:

```
git clone --depth 1 -b fips-2022-11-02 https://github.com/aws/aws-lc && \  
  cd aws-lc && \  
  mkdir build && \  
  cd build && \  
  cmake -G Ninja \  
    -DCMAKE_INSTALL_LIBDIR=lib \  
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \  
  cmake --build . && \  
  cmake --install . && \  
  rm -rf ./ * && \  
  cmake -G Ninja \  
    -DBUILD_SHARED_LIBS=ON \  
    -DCMAKE_INSTALL_LIBDIR=lib \  
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \  
  cmake --build . && \  
  cmake --install .
```

Step 2: Build curl from source or use a curl distribution with your libcrypto library

The SDK for C++ requires that an HTTP client is installed on the system that will be used to make HTTP requests. The HTTP client must be built with the libcrypto that you intend to use. The HTTP client is responsible for TLS operations and, thus, uses your libcrypto library.

In the following example, the curl library is rebuilt using an installed version of AWS-LC.

```
git clone --depth 1 -b curl-8_5_0 https://github.com/curl/curl && \  
  cd curl && \  
  autoreconf -fi && \  
  mkdir build && \  
  cd build && \  
  ../configure \  
    --enable-warnings \  
    --enable-werror \  
  make
```

```

--with-openssl=/lc-install \
--prefix=/curl-install && \
make && \
make install

```

Step 3: Build the SDK using libcrypto and curl libraries

The SDK for C++ can now be built using the previously created libcrypto and curl artifacts. This build of the SDK will use the custom libcrypto library for all cryptographic functionality.

```

git clone --depth 1 --recurse-submodules https://github.com/aws/aws-sdk-cpp \
cd aws-sdk-cpp && \
mkdir build && \
cd build && \
cmake -G Ninja \
  -DCMAKE_PREFIX_PATH="/curl-install;/lc-install;" \
  -DBUILD_ONLY="s3" \
  -DCMAKE_INSTALL_PREFIX=/sdk-install \
  -DAUTORUN_UNIT_TESTS=OFF .. && \
cmake --build . && \
cmake --install .

```

Bringing it all together in a docker image

The following sample Docker file shows how to implement these steps in Amazon Linux 2023 environment.

```

# User AL2023 Base image
FROM public.ecr.aws/amazonlinux/amazonlinux:2023

# Install Dev Tools
RUN yum groupinstall -y "Development Tools"
RUN yum install -y cmake3 ninja-build

# Build and install AWS-LC on the fips branch both statically and dynamically.
RUN git clone --depth 1 -b fips-2022-11-02 https://github.com/aws/aws-lc && \
  cd aws-lc && \
  mkdir build && \
  cd build && \
  cmake -G Ninja \
    -DCMAKE_INSTALL_LIBDIR=lib \

```

```
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \\  
cmake --build . && \\  
cmake --install . && \\  
rm -rf ./ * && \\  
cmake -G Ninja \\  
    -DBUILD_SHARED_LIBS=ON \\  
    -DCMAKE_INSTALL_LIBDIR=lib \\  
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \\  
cmake --build . && \\  
cmake --install .  
  
# Build and install curl targeting AWS-LC as openssl  
RUN git clone --depth 1 -b curl-8_5_0 https://github.com/curl/curl && \\  
    cd curl && \\  
    autoreconf -fi && \\  
    mkdir build && \\  
    cd build && \\  
    ../configure \\  
        --enable-warnings \\  
        --enable-werror \\  
        --with-openssl=/lc-install \\  
        --prefix=/curl-install && \\  
    make && \\  
    make install  
  
# Build and install SDK using the Curl and AWS-LC targets previously built  
RUN git clone --depth 1 --recurse-submodules https://github.com/aws/aws-sdk-cpp \\  
    cd aws-sdk-cpp && \\  
    mkdir build && \\  
    cd build && \\  
    cmake -G Ninja \\  
        -DCMAKE_PREFIX_PATH="/curl-install;/lc-install;" \\  
        -DBUILD_ONLY="s3" \\  
        -DCMAKE_INSTALL_PREFIX=/sdk-install \\  
        -DAUTORUN_UNIT_TESTS=OFF .. && \\  
    cmake --build . && \\  
    cmake --install .
```

Work with AWS services using the AWS SDK for C++

The following sections contain examples, tutorials, tasks, and guides that show you how to use the AWS SDK for C++ to work with AWS services.

If you're new to the AWS SDK for C++, you might want to read through the [Get started](#) topic first.

You can find more code examples in the [C++ example folder](#) on GitHub.

You can find more code examples in the [Code examples](#) chapter of this guide or in the [AWS Code Examples Repository](#) on GitHub.

Topics

- [Get started with the AWS SDK for C++ code examples](#)
- [Asynchronous methods](#)
- [Code examples with guidance for the AWS SDK for C++](#)

Get started with the AWS SDK for C++ code examples

Structure of the code examples

The [C++ example folder](#) on Github contains project folders for each AWS service. Typically, individual .cpp source files in the folders demonstrate a specific functionality or action for that service. For example, for Amazon DynamoDB, *getting* an item from the database and *uploading* an item to the database are two different types of action, so there is a separate file for each in the DynamoDB folder: `get_item.cpp` and `put_item.cpp`. Each .cpp file contains a `main()` function as an entrypoint to a standalone executable. The project executables are generated in a folder designated by your build system, and there is one executable file corresponding to each example source file. The file name of the executable follows the conventions of the platform such as `{name}.exe` or just `{name}` and any custom prefix `CMakeLists.txt` applies such as `run_`.

To run an example functionality

1. Download the desired code example from the [AWS Code Examples Repository](#) on GitHub.
2. Open a .cpp file to explore its `main()` function and any called methods.

3. Build the project, as demonstrated with the starter example in [Getting started using the AWS SDK for C++](#). Note that building the project generates each executable for every source file in the project.
4. Run the executable for the selected functionality.
 - In a command prompt, run that program using the executable based on the name of the *.cpp file.
 - If you are working within an IDE, choose the .cpp file of the functionality you want to demonstrate and select it as the startup option (or startup object).

Unit tests

Tests for examples are written using the GoogleTest framework. To learn more, see [GoogleTest Primer](#) on the GoogleTest website.

The unit tests for each example are in a `tests` subfolder containing its own `CMakeLists.txt` file. For each example source file there is a corresponding test file named `gtest_<source file>`. The test executable for the subfolder is named `<AWS service>_gtests`.

CMakeLists.txt file

The folder for each service contains a file named `CMakeLists.txt` file. Many of these files contain a construct similar to the following:

```
foreach(EXAMPLE IN LISTS EXAMPLES)
    add_executable(${EXAMPLE} ${EXAMPLE}.cpp)
    target_link_libraries(${EXAMPLE} aws-cpp-sdk-email aws-cpp-sdk-core)
endforeach()
```

For each .cpp file in the folder, the `CMakeLists.txt` file builds an executable (cmake: `add_executable`) with a name based on the name of the source code file without the file extension.

Using AWS for troubleshooting and diagnostics

As you learn to develop applications with the AWS SDK for C++, it's also valuable to get comfortable in using both the AWS Management Console and the AWS CLI. These tools can be used interchangeably for various troubleshooting and diagnostics .

The following tutorial shows you an example of these troubleshooting and diagnostics tasks. It focuses on the `Access Denied` error, which can be encountered for several different reasons. The tutorial shows an example of how you might determine the actual cause of the error. It focuses on two of the possible causes: incorrect permissions for the current user and a resource that isn't available to the current user.

To get the project source and executables

1. Download the Amazon S3 code example folder from [AWS Code Examples Repository](#) on GitHub.
2. Open `delete_bucket.cpp` and notice that there are two methods: `main()` and `DeleteBucket()`. `DeleteBucket()` uses the SDK to delete the bucket.
3. Also notice that there are two "TODO" blocks with instructions for updating the code. Do **NOT** perform these updates yet; they will be performed later in the tutorial.
4. Build the Amazon S3 example, using the same build steps explained in [Getting started using the AWS SDK for C++](#). The build process generates an executable for each source file.
5. Open a command prompt to the folder where your build system generated your build executables. Run the executable `run_create_bucket` (your actual executable filename will differ based on your operating system). This creates a bucket in your account (so that you have one to delete).
6. In the command prompt, run the executable `run_delete_bucket`.
7. Confirm that you get an `Access Denied` error message. *Getting an Access Denied error message leads you to question whether you created a user with full permissions for Amazon S3, which you'll verify next.*

To install the AWS CLI and find the username that is making calls to AWS

1. To install the latest AWS CLI to your development machine, see [Installing the AWS CLI](#) in the *AWS Command Line Interface User Guide*.
2. To verify the AWS CLI is working, open a command prompt and run command `aws --version`

```
$ aws --  
version  
aws-cli/2.1.29 Python/3.8.8 Windows/10 exe/AMD64 prompt/off
```

3. To obtain the username that is actually making the calls to AWS, run the AWS CLI command `aws sts get-caller-identity`. In the following example output, that username is *userX*

```
$ aws sts get-caller-identity
{
  "UserId": "A12BCD34E5FGHI6JKLM",
  "Account": "1234567890987",
  "Arn": "arn:aws:iam::1234567890987:user/userX"
}
```

There are many ways to specify credentials, but if you followed the approach in [SDK authentication with AWS](#) then this username comes from your AWS shared credentials file. During that procedure you granted your user **AmazonS3FullAccess** permissions.

Note

Generally, most AWS CLI commands follow the syntax structure of:

```
$ aws <command> <subcommand> [options and parameters]
```

where *command* is the service, and *subcommand* is the method being called on that service. For more details, see [Command structure in the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

To verify whether a user has permission to delete a bucket

1. Open the [AWS Management Console](#) and log in. For more details, see [Getting Started with the AWS Management Console](#).
2. In the main navigation bar, for **Search for services...**, enter **IAM** and select the IAM service from the results.
3. From the **Dashboard** sidebar, or under **IAM Resources**, select **Users**.
4. From the table of users available for your account, select the username obtained in the preceding procedure.
5. Choose the **Permissions** tab of the **Summary** page, under the **Policy name** table, select **AmazonS3FullAccess**.

6. Look at the **Policy summary** and the JSON data. Verify that this user has full rights for the Amazon S3 service.

```
"Effect": "Allow",  
"Action": "s3:*",  
"Resource": "*"
```

This process of elimination is common in ruling *out* where the problem might be. In this case, you've verified that the user does have the correct permissions, so the problem must be something else. That is, since you have the correct permissions to access your buckets, the Access Denied error may mean that you are trying to access a bucket that isn't yours. Next, review the bucket name in the code, which is "my-bucket". Notice that a bucket with that name doesn't exist in your account, and thus, you cannot 'access' it.

To update the code example so it runs successfully

1. Back in `delete_bucket.cpp`'s `main()` function, change the Region, using the enum, to the Region of your account. To find your Region of your account, log into the AWS Management Console, and locate the Region in the upper right-hand corner. Also in `main()`, change the bucket name to a bucket that **does** exist in your account. There are several ways to find your current bucket names:
 - You can use the `run_list_buckets` executable that also exists in this code example's folder to programatically get the names of your buckets.
 - Alternatively, you can also use the following AWS CLI command to list your Amazon S3 buckets.

```
$ aws s3  
ls  
2022-01-05 14:27:48 my-bucket-28621ccd-4a48-45be-b660-5252f75635a4
```

- Alternatively, you can also use the [AWS Management Console](#). In the main navigation bar, in **Search for services...**, enter **S3**. The Buckets page lists your account's buckets.
2. Rebuild the code and run the updated executable `run_delete_bucket`.
 3. Using either the AWS Management Console or the AWS CLI, verify that the Amazon S3 bucket that you created earlier has been deleted.

Building and Debugging Code Examples in Visual Studio

Building and running the Amazon S3 code example

1. Obtain the Amazon S3 example source code. This procedure uses the [Amazon S3 code examples using the AWS SDK for C++](#) code example to get up and running using Visual Studio.
2. In Windows Explorer, navigate to the s3 folder (e.g. `\aws-doc-sdk-examples\cpp\example_code\s3`).
3. Right click on the s3 example folder and choose **Open with Visual Studio**. Visual Studio for CMake projects don't have a 'project' file, rather, it is the whole folder.
4. In the **Configuration Selector** dropdown in the top menu of Visual Studio, ensure that the selected configuration matches the build type that you selected when building the SDK from source. E.g. a **Debug** configuration should be selected if you built from source using debug (`-DCMAKE_BUILD_TYPE=Debug` in the CMake command line from the SDK installation instructions).
5. Open file `CMakeLists.txt`.
6. Click **Save**. Every time you click **Save** on the `CMakeLists.txt` file, Visual Studio refreshes the CMake-generated files. If you have your **Output** tab displayed, you can see the resulting log messages from this generation.
 - There is a drop-down box within the **Output** tab that says: "**Show output from:**" and **CMake** should be the option selected by default.
 - Last message output should say "**CMake generation finished.**"
 - If last message is not this, then the CMake file has issues. Do not proceed to further steps until this is resolved. See [Troubleshoot build issues](#).
 - Note that the CMake cache is used by CMake for speed. If you are working through CMake issues, you want to ensure a 'clean slate' so that the error messages you are given is actually reflective of your most recent changes. In the Solution Explorer, right-click on `CMakeLists.txt` and choose **CMake Cache**, then choose **Delete Cache**. Do this frequently when working progressively through CMake issues.
7. To build and run examples from within Visual Studio, Visual Studio places executables in a different folder structure than the command line. To run the code, the SDK executables must be copied to the right place. Find the "TODO" line of the `CMakeLists` file (~line 40) and choose the one commented for use in Visual Studio. Visual Studio does not use a subfolder

- dedicated to the build type so this is not included. Switch the commented-out line in the `CMakeLists.txt` file for Visual Studio use.
8. Delete the CMake cache (as described above), click in the `CMakeLists.txt` file to select/activate the tab, and choose **Save** on the `CMakeLists.txt` file again to initiate the CMake build files generation.
 9. Open the source file of the 'program' you wish to run.
 - For example, open `list_buckets.cpp`.
 - The Amazon S3 example folder is coded so that each showcased 'feature' of Amazon S3 is demonstrated in a dedicated executable for just that feature. E.g. `list_buckets.cpp` will become an executable that only demonstrates the listing of buckets.
 10. In the top menu, choose **Build**, then choose **Build All**.
 - The **Output** tab's **Show output from** should reflect the selection of **Build**, and show all the building and linking messages.
 - The last output should be: "**Build All succeeded.**"
 - Now executables for each of the individual source files are generated. You can confirm this by looking in the build output directory (e.g. `\aws-doc-sdk-examples\cpp\example_code\s3\out\build\x64-Debug`).
 - Note that the executables are prefixed with "run_" because the `CMakeLists.txt` file dictates this.
 11. In the top menu, there is a **green arrow** and a **drop-down selector** for **Debug Target**. Choose `run_list_buckets.exe`.
 12. Click the **green arrow run button** to **Select Startup Item**.
 13. A Visual Studio Debug Console window will open and display the output of the code.
 14. Press a key to close the window, or manually close the window, to terminate the program. You can also set breakpoints in the code and when you click run again the breakpoints will be hit.

Asynchronous methods

Asynchronous SDK methods

For many methods, the SDK for C++ provides both synchronous and asynchronous versions. A method is asynchronous if it includes the `Async` suffix in its name. For example, the Amazon S3 method `PutObject` is synchronous, while `PutObjectAsync` is asynchronous.

Like all asynchronous operations, an asynchronous SDK method returns before its main task is finished. For example, the `PutObjectAsync` method returns before it finishes uploading the file to the Amazon S3 bucket. While the upload operation continues, the application can perform other operations, including calling other asynchronous methods. The application is notified that an asynchronous operation has finished when an associated callback function is invoked.

The following sections describe a code example that demonstrates calling an SDK asynchronous method. Each section focuses on individual portions from the example's [entire source file](#).

Calling SDK asynchronous methods

In general, the asynchronous version of an SDK method accepts the following arguments.

- A reference to the same `Request`-type object as its synchronous counterpart.
- A reference to a response handler callback function. This callback function is invoked when the asynchronous operation finishes. One of the arguments contains the operation's outcome.
- An optional `shared_ptr` to an `AsyncCallerContext` object. The object is passed to the response handler callback. It includes a `UUID` property that can be used to pass text information to the callback.

The `put_s3_object_async` method shown below sets up and calls the SDK's Amazon S3 `PutObjectAsync` method to asynchronously upload a file to an Amazon S3 bucket.

The method initializes a `PutObjectRequest` object in the same manner as its synchronous counterpart. In addition, a `shared_ptr` to an `AsyncCallerContext` object is allocated. Its `UUID` property is set to the Amazon S3 object name. For demonstration purposes, the response handler callback will access the property and output its value.

The call to `PutObjectAsync` includes a reference argument to the response handler callback function `put_object_async_finished`. This callback function is examined in more detail in the next section.

```
bool AwsDoc::S3::PutObjectAsync(const Aws::S3::S3Client &s3Client,
                                const Aws::String &bucketName,
                                const Aws::String &fileName) {
    // Create and configure the asynchronous put object request.
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);
```

```
const std::shared_ptr<Aws::IOStream> input_data =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
        fileName.c_str(),
        std::ios_base::in | std::ios_base::binary);

if (!*input_data) {
    std::cerr << "Error: unable to open file " << fileName << std::endl;
    return false;
}

request.SetBody(input_data);

// Create and configure the context for the asynchronous put object request.
std::shared_ptr<Aws::Client::AsyncCallerContext> context =
    Aws::MakeShared<Aws::Client::AsyncCallerContext>("PutObjectAllocationTag");
context->SetUUID(fileName);

// Make the asynchronous put object call. Queue the request into a
// thread executor and call the PutObjectAsyncFinished function when the
// operation has finished.
s3Client.PutObjectAsync(request, PutObjectAsyncFinished, context);

return true;
}
```

The resources directly associated with an asynchronous operation must continue to exist until the operation finishes. For example, the client object used to invoke an asynchronous SDK method must exist until the application receives notification that the operation has finished. Similarly, the application itself cannot terminate until the asynchronous operation completes.

For this reason, the `put_s3_object_async` method accepts a reference to an `S3Client` object instead of creating the client in a local variable. In the example, the method returns to the caller immediately after beginning the asynchronous operation, enabling the caller to perform additional tasks while the upload operation is in progress. If the client is stored in a local variable, it would go out of scope when the method returned. However, the client object must continue to exist until the asynchronous operation finishes.

Notification of the Completion of an Asynchronous Operation

When an asynchronous operation finishes, an application response handler callback function is invoked. This notification includes the outcome of the operation. The outcome is contained in the

same Outcome-type class returned by the method's synchronous counterpart. In the code example, the outcome is in a `PutObjectOutcome` object.

The example's response handler callback function `put_object_async_finished` is shown below. It checks whether the asynchronous operation succeeded or failed. It uses a `std::condition_variable` to notify the application thread that the async operation has finished.

```
// A mutex is a synchronization primitive that can be used to protect shared
// data from being simultaneously accessed by multiple threads.
std::mutex AwsDoc::S3::upload_mutex;

// A condition_variable is a synchronization primitive that can be used to
// block a thread, or to block multiple threads at the same time.
// The thread is blocked until another thread both modifies a shared
// variable (the condition) and notifies the condition_variable.
std::condition_variable AwsDoc::S3::upload_variable;
```

```
void PutObjectAsyncFinished(const Aws::S3::S3Client *s3Client,
                           const Aws::S3::Model::PutObjectRequest &request,
                           const Aws::S3::Model::PutObjectOutcome &outcome,
                           const std::shared_ptr<const
Aws::Client::AsyncCallerContext> &context) {
    if (outcome.IsSuccess()) {
        std::cout << "Success: PutObjectAsyncFinished: Finished uploading '"
                  << context->GetUUID() << "'." << std::endl;
    }
    else {
        std::cerr << "Error: PutObjectAsyncFinished: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    // Unblock the thread that is waiting for this function to complete.
    AwsDoc::S3::upload_variable.notify_one();
}
```

With the asynchronous operation finished, the resources associated with it can be released. The application can also terminate if it wishes.

The following code demonstrates how the `put_object_async` and `put_object_async_finished` methods are used by an application.

The `S3Client` object is allocated so it continues to exist until the asynchronous operation finishes. After calling `put_object_async`, the application can perform whatever operations it wishes. For simplicity, the example uses a `std::mutex` and `std::condition_variable` to wait until the response handler callback notifies it that the upload operation has finished.

```
int main() {
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        // TODO(user): Change bucket_name to the name of a bucket in your account.
        const Aws::String bucket_name = "<Enter a bucket name>";
        //TODO(user): Create a file called "my-file.txt" in the local folder where your
        executables are built to.
        const Aws::String object_name = "my-file.txt";

        // A unique_lock is a general-purpose mutex ownership wrapper allowing
        // deferred locking, time-constrained attempts at locking, recursive
        // locking, transfer of lock ownership, and use with
        // condition variables.
        std::unique_lock<std::mutex> lock(AwsDoc::S3::upload_mutex);

        // Create and configure the Amazon S3 client.
        // This client must be declared here, as this client must exist
        // until the put object operation finishes.
        Aws::Client::ClientConfiguration config;
        // Optional: Set to the AWS Region in which the bucket was created (overrides
        config file).
        // config.region = "us-east-1";

        Aws::S3::S3Client s3_client(config);

        AwsDoc::S3::PutObjectAsync(s3_client, bucket_name, object_name);

        std::cout << "main: Waiting for file upload attempt..." <<
            std::endl << std::endl;

        // While the put object operation attempt is in progress,
        // you can perform other tasks.
        // This example simply blocks until the put object operation
        // attempt finishes.
        AwsDoc::S3::upload_variable.wait(lock);

        std::cout << std::endl << "main: File upload attempt completed."
    }
}
```

```
        << std::endl;
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

See the [complete example](#) on Github.

Code examples with guidance for the AWS SDK for C++

If you are new to AWS or the AWS code examples, we recommend you start with [Get started on code examples](#).

Source code that shows how to work with AWS services using the AWS SDK for C++ is available in the [Code examples](#) chapter of this guide or in the [AWS Code Examples Repository](#) on GitHub.

This section selects several AWS services and guides you through the examples using them. The following guided examples are a subset of what is available on Github.

Service examples with additional explanation (see [AWS Code Examples Repository](#) for full list)

Service	Summary of what the service provides to your program
Amazon CloudWatch	Collects and monitors metrics for AWS resources you are using
Amazon DynamoDB	A NoSQL database service
Amazon Elastic Compute Cloud (Amazon EC2)	Secure, resizable compute capacity
Amazon Simple Storage Service (Amazon S3)	Data storage and retrieval (objects into buckets)
Amazon Simple Queue Service (Amazon SQS)	Message queuing service to send, store, and receive messages between software components

There are also examples that show how to use [Asynchronous methods](#).

To propose a new code example to the AWS documentation team, see [Contributing guidelines](#) on GitHub to create a new request. The team prefers to create code examples that show broad scenarios rather than individual API calls.

Using the Code Examples on Windows

If you are building the examples on Windows with SDK version 1.9, see [Troubleshoot build issues](#).

Amazon CloudWatch examples using the AWS SDK for C++

Amazon CloudWatch (CloudWatch) is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use the following examples to program [CloudWatch](#) using the AWS SDK for C++.

Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.

For more information about CloudWatch, see the [Amazon CloudWatch User Guide](#).

Note

Only the code that is necessary to demonstrate certain techniques is supplied in this Guide, but the [complete example code is available on GitHub](#). On GitHub you can download a single source file or you can clone the repository locally to get, build, and run all examples.

Topics

- [Getting Metrics from CloudWatch](#)
- [Publishing Custom Metric Data](#)
- [Working with CloudWatch Alarms](#)
- [Using Alarm Actions in CloudWatch](#)
- [Sending Events to CloudWatch](#)

Getting Metrics from CloudWatch

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Listing Metrics

To list CloudWatch metrics, create a [ListMetricsRequest](#) and call the CloudWatchClient's ListMetrics function. You can use the ListMetricsRequest to filter the returned metrics by namespace, metric name, or dimensions.

Note

A list of metrics and dimensions that are posted by AWS services can be found within the [Amazon CloudWatch Metrics and Dimensions Reference](#) in the Amazon CloudWatch User Guide.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}
```

```
    }

    if (argc > 2)
    {
        request.SetNamespace(argv[2]);
    }

    bool done = false;
    bool header = false;
    while (!done)
    {
        auto outcome = cw.ListMetrics(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to list CloudWatch metrics:" <<
                outcome.GetError().GetMessage() << std::endl;
            break;
        }

        if (!header)
        {
            std::cout << std::left << std::setw(48) << "MetricName" <<
                std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                std::endl;
            header = true;
        }

        const auto &metrics = outcome.GetResult().GetMetrics();
        for (const auto &metric : metrics)
        {
            std::cout << std::left << std::setw(48) <<
                metric.GetMetricName() << std::setw(32) <<
                metric.GetNamespace();
            const auto &dimensions = metric.GetDimensions();
            for (auto iter = dimensions.cbegin();
                iter != dimensions.cend(); ++iter)
            {
                const auto &dimkv = *iter;
                std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                if (iter + 1 != dimensions.cend())
                {
                    std::cout << ", ";
                }
            }
        }
    }
}
```

```
        std::cout << std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}
```

The metrics are returned in a [ListMetricsResult](#) by calling its `GetMetrics` function. The results may be *paged*. To retrieve the next batch of results, call `SetNextToken` on the original request object with the return value of the `ListMetricsResult` object's `GetNextToken` function, and pass the modified request object back to another call to `ListMetrics`.

See the [complete example](#).

More Information

- [ListMetrics](#) in the Amazon CloudWatch API Reference.

Publishing Custom Metric Data

A number of AWS services publish [their own metrics](#) in namespaces beginning with `AWS/`. You can also publish custom metric data using your own namespace (as long as it doesn't begin with `AWS/`).

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Publish Custom Metric Data

To publish your own metric data, call the `CloudWatchClient`'s `PutMetricData` function with a [PutMetricDataRequest](#). The `PutMetricDataRequest` must include the custom namespace to use for the data, and information about the data point itself in a [MetricDatum](#) object.

Note

You cannot specify a namespace that begins with `AWS/`. Namespaces that begin with `AWS/` are reserved for use by Amazon Web Services products.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully put sample metric data" << std::endl;
}
```

See the [complete example](#).

More Information

- [Using Amazon CloudWatch Metrics](#) in the Amazon CloudWatch User Guide.
- [AWS Namespaces](#) in the Amazon CloudWatch User Guide.
- [PutMetricData](#) in the Amazon CloudWatch API Reference.

Working with CloudWatch Alarms

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create an Alarm

To create an alarm based on a CloudWatch metric, call the `CloudWatchClient`'s `PutMetricAlarm` function with a [PutMetricAlarmRequest](#) filled with the alarm conditions.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
```



```
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch alarm " << alarm_name
        << std::endl;
}
```

See the [complete example](#).

List Alarms

To list the CloudWatch alarms that you have created, call the CloudWatchClient's DescribeAlarms function with a [DescribeAlarmsRequest](#) that you can use to set options for the result.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.DescribeAlarms(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to describe CloudWatch alarms:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Arn" <<
            std::setw(64) << "Description" <<
            std::setw(20) << "LastUpdated" <<
            std::endl;
        header = true;
    }

    const auto &alarms = outcome.GetResult().GetMetricAlarms();
    for (const auto &alarm : alarms)
    {
        std::cout << std::left <<
            std::setw(32) << alarm.GetAlarmName() <<
            std::setw(64) << alarm.GetAlarmArn() <<
            std::setw(64) << alarm.GetAlarmDescription() <<
            std::setw(20) <<
            alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                SIMPLE_DATE_FORMAT_STR) <<
            std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
}
```

```
        done = next_token.empty();
    }
```

The list of alarms can be obtained by calling `getMetricAlarms` on the [DescribeAlarmsResult](#) that is returned by `DescribeAlarms`.

The results may be *paged*. To retrieve the next batch of results, call `SetNextToken` on the original request object with the return value of the `DescribeAlarmsResult` object's `GetNextToken` function, and pass the modified request object back to another call to `DescribeAlarms`.

Note

You can also retrieve alarms for a specific metric by using the `CloudWatchClient`'s `DescribeAlarmsForMetric` function. Its use is similar to `DescribeAlarms`.

See the [complete example](#).

Delete Alarms

To delete CloudWatch alarms, call the `CloudWatchClient`'s `DeleteAlarms` function with a [DeleteAlarmsRequest](#) containing one or more names of alarms that you want to delete.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to delete CloudWatch alarm:" <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
            << std::endl;
    }
}
```

See the [complete example](#).

More Information

- [Creating Amazon CloudWatch Alarms](#) in the Amazon CloudWatch User Guide
- [PutMetricAlarm](#) in the Amazon CloudWatch API Reference
- [DescribeAlarms](#) in the Amazon CloudWatch API Reference
- [DeleteAlarms](#) in the Amazon CloudWatch API Reference

Using Alarm Actions in CloudWatch

Using CloudWatch alarm actions, you can create alarms that perform actions such as automatically stopping, terminating, rebooting, or recovering Amazon EC2 instances.

Alarm actions can be added to an alarm by using the [PutMetricAlarmRequest](#)'s `SetAlarmActions` function when [creating an alarm](#).

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Enable Alarm Actions

To enable alarm actions for a CloudWatch alarm, call the `CloudWatchClient`'s `EnableAlarmActions` with a [EnableAlarmActionsRequest](#) containing one or more names of alarms whose actions you want to enable.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
```

```
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;
```

See the [complete example](#).

Disable Alarm Actions

To disable alarm actions for a CloudWatch alarm, call the CloudWatchClient's `DisableAlarmActions` with a [DisableAlarmActionsRequest](#) containing one or more names of alarms whose actions you want to disable.

Includes

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Code

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::DisableAlarmActionsRequest disableAlarmActionsRequest;
disableAlarmActionsRequest.AddAlarmNames(alarm_name);

auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
if (!disableAlarmActionsOutcome.IsSuccess())
{
    std::cout << "Failed to disable actions for alarm " << alarm_name <<
        ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
        std::endl;
}
else
{
```

```
        std::cout << "Successfully disabled actions for alarm " <<
            alarm_name << std::endl;
    }
```

See the [complete example](#).

More Information

- [Create Alarms to Stop, Terminate, Reboot, or Recover an Instance](#) in the Amazon CloudWatch User Guide
- [PutMetricAlarm](#) in the Amazon CloudWatch API Reference
- [EnableAlarmActions](#) in the Amazon CloudWatch API Reference
- [DisableAlarmActions](#) in the Amazon CloudWatch API Reference

Sending Events to CloudWatch

CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources to Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, or built-in targets. You can match events and route them to one or more target functions or streams by using simple rules.

Note

These code snippets assume that you understand the material in [Getting Started Using the AWS SDK for C++](#) and have configured default AWS credentials using the information in [Providing AWS Credentials](#).

Add Events

To add custom CloudWatch events, call the CloudWatchEventsClient's PutEvents function with a [PutEventsRequest](#) object that contains one or more [PutEventsRequestEntry](#) objects that provide details about each event. You can specify several parameters for the entry such as the source and type of the event, resources associated with the event, and so on.

Note

You can specify a maximum of 10 events per call to putEvents.

Includes

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Code

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}
```

Add Rules

To create or update a rule, call the `CloudWatchEventsClient`'s `PutRule` function with a [PutRuleRequest](#) with the name of the rule and optional parameters such as the [event pattern](#), IAM role to associate with the rule, and a [scheduling expression](#) that describes how often the rule is run.

Includes

```
#include <aws/core/Aws.h>
```



```
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Code

```
Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;
request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}
}
```

Add Targets

Targets are the resources that are invoked when a rule is triggered. Example targets include Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, and built-in targets.

To add a target to a rule, call the `CloudWatchEventsClient`'s `PutTargets` function with a [PutTargetsRequest](#) containing the rule to update and a list of targets to add to the rule.

Includes

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
```

```
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Code

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);
target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
        "Successfully created CloudWatch events target for rule "
        << rule_name << std::endl;
}
}
```

See the [complete example](#).

More Information

- [Adding Events with PutEvents](#) in the Amazon CloudWatch Events User Guide
- [Schedule Expressions for Rules](#) in the Amazon CloudWatch Events User Guide
- [Event Types for CloudWatch Events](#) in the Amazon CloudWatch Events User Guide
- [Events and Event Patterns](#) in the Amazon CloudWatch Events User Guide
- [PutEvents](#) in the Amazon CloudWatch Events API Reference
- [PutTargets](#) in the Amazon CloudWatch Events API Reference

- [PutRule](#) in the Amazon CloudWatch Events API Reference

Amazon DynamoDB examples using the AWS SDK for C++

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. The following examples show how you can program [Amazon DynamoDB](#) using the AWS SDK for C++.

Note

Only the code that is necessary to demonstrate certain techniques is supplied in this Guide, but the [complete example code is available on GitHub](#). On GitHub you can download a single source file or you can clone the repository locally to get, build, and run all examples.

Topics

- [Working with Tables in DynamoDB](#)
- [Working with Items in DynamoDB](#)

Working with Tables in DynamoDB

Tables are the containers for all items in a DynamoDB database. Before you can add or remove data from DynamoDB, you must create a table.

For each table, you must define:

- A table *name* that is unique for your AWS account and AWS Region.
- A *primary key* for which every value must be unique. No two items in your table can have the same primary key value.

A primary key can be *simple*, consisting of a single partition (HASH) key, or *composite*, consisting of a partition and a sort (RANGE) key.

Each key value has an associated *data type*, enumerated by the [ScalarAttributeType](#) class. The key value can be binary (B), numeric (N), or a string (S). For more information, see [Naming Rules and Data Types](#) in the Amazon DynamoDB Developer Guide.

- *Provisioned throughput* values that define the number of reserved read/write capacity units for the table.

Note

[Amazon DynamoDB pricing](#) is based on the provisioned throughput values that you set on your tables, so reserve only as much capacity as you think you'll need for your table. Provisioned throughput for a table can be modified at any time, so you can adjust capacity if your needs change.

Create a Table

Use the [DynamoDB client](#) `CreateTable` method to create a new DynamoDB table. You need to construct table attributes and a table schema, both of which are used to identify the primary key of your table. You must also supply initial provisioned throughput values and a table name. `CreateTable` is an asynchronous operation. `GetTableStatus` will return `CREATING` until the table is `ACTIVE` and ready for use.

Create a Table with a Simple Primary Key

This code creates a table with a simple primary key ("Name").

Includes

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/CreateTableRequest.h>
#include <aws/dynamodb/model/KeySchemaElement.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/ScalarAttributeType.h>
#include <iostream>
```

Code

```
//! Create an Amazon DynamoDB table.
/*!
    \sa createTable()
    \param tableName: Name for the DynamoDB table.
    \param primaryKey: Primary key for the DynamoDB table.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
```

```
*/
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                    const Aws::String &primaryKey,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
        " with a simple primary key: \"" << primaryKey << "\"." << std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome = dynamoClient.CreateTable(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
            << outcome.GetResult().GetTableDescription().GetTableName() <<
            " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the [complete example](#).

Create a Table with a Composite Primary Key

Add another [AttributeDefinition](#) and [KeySchemaElement](#) to [CreateTableRequest](#).

Includes

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/CreateTableRequest.h>
#include <aws/dynamodb/model/KeySchemaElement.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/ScalarAttributeType.h>
#include <iostream>
```

Code

```
//! Create an Amazon DynamoDB table with a composite key.
/*!
 \sa createTableWithCompositeKey()
 \param tableName: Name for the DynamoDB table.
 \param partitionKey: Name for the partition (hash) key.
 \param sortKey: Name for the sort (range) key.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTableWithCompositeKey(const Aws::String &tableName,
                                                    const Aws::String &partitionKey,
                                                    const Aws::String &sortKey,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
                " with a composite primary key:\n" \
                "* " << partitionKey << " - partition key\n" \
                "* " << sortKey << " - sort key\n";

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey1, hashKey2;
    hashKey1.WithAttributeName(partitionKey).WithAttributeType(
        Aws::DynamoDB::Model::ScalarAttributeType::S);
```

```

request.AddAttributeDefinitions(hashKey1);
hashKey2.WithAttributeName(sortKey).WithAttributeType(
    Aws::DynamoDB::Model::ScalarAttributeType::S);
request.AddAttributeDefinitions(hashKey2);

Aws::DynamoDB::Model::KeySchemaElement keySchemaElement1, keySchemaElement2;
keySchemaElement1.WithAttributeName(partitionKey).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(keySchemaElement1);
keySchemaElement2.WithAttributeName(sortKey).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(keySchemaElement2);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
request.SetProvisionedThroughput(throughput);

request.SetTableName(tableName);

const Aws::DynamoDB::Model::CreateTableOutcome &outcome = dynamoClient.CreateTable(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Table \""
        << outcome.GetResult().GetTableDescription().GetTableName() <<
        "\" was created!" << std::endl;
}
else {
    std::cerr << "Failed to create table:" << outcome.GetError().GetMessage()
        << std::endl;
}

return outcome.IsSuccess();
}

```

See the [complete example](#) on GitHub.

List Tables

You can list the tables in a particular region by calling the [DynamoDB client](#) `ListTables` method.

Includes

```
#include <aws/core/Aws.h>
```

```
#include <aws/core/utils/Outcome.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <aws/dynamodb/model/ListTablesResult.h>
#include <iostream>
```

Code

```
//! List the Amazon DynamoDB tables for the current AWS account.
/*!
 \sa listTables()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamoClient.ListTables(
            listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
            outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}
```

By default, up to 100 tables are returned per call. Use `GetExclusiveStartTableName` on the returned [ListTablesOutcome](#) object to get the last table that was evaluated. You can use this value to start the listing after the last returned value of the previous listing.

See the [complete example](#).

Retrieve Information about a Table

You can find out more about a table by calling the [DynamoDB client](#) `DescribeTable` method.

Includes

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/DescribeTableRequest.h>
#include <iostream>
```

Code

```
//! Describe an Amazon DynamoDB table.
/*!
 \sa describeTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
    dynamoClient.DescribeTable(
        request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
    outcome.GetResult().GetTable();
        std::cout << "Table name   : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN    : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
            << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;
    }
}
```

```

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() << std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() << std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
                a.GetAttributeType()) <<
                ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}

```

See the [complete example](#) on GitHub.

Modify a Table

You can modify your table's provisioned throughput values at any time by calling the [DynamoDB client](#) `UpdateTable` method.

Includes

```

#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/UpdateTableRequest.h>
#include <iostream>

```

Code

```

//! Update a DynamoDB table.

```

```

/*!
 \sa updateTable()
 \param tableName: Name for the DynamoDB table.
 \param readCapacity: Provisioned read capacity.
 \param writeCapacity: Provisioned write capacity.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long writeCapacity,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput values"
                << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;
    provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
        writeCapacity);
    request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome = dynamoClient.UpdateTable(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

See the [complete example](#).

Delete a Table

Call the [DynamoDB client](#) DeleteTable method and pass it the table's name.

Includes

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/DeleteTableRequest.h>
#include <iostream>
```

Code

```
//! Delete an Amazon DynamoDB table.
/*!
 \sa deleteTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result = dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

See the [complete example](#) on GitHub.

More Info

- [Guidelines for Working with Tables](#) in the Amazon DynamoDB Developer Guide

- [Working with Tables in DynamoDB](#) in the Amazon DynamoDB Developer Guide

Working with Items in DynamoDB

In DynamoDB, an item is a collection of *attributes*, each of which has a *name* and a *value*. An attribute value can be a scalar, set, or document type. For more information, see [Naming Rules and Data Types](#) in the Amazon DynamoDB Developer Guide.

Retrieve an Item from a Table

Call the [DynamoDB client](#) `GetItem` method. Pass it a [GetItemRequest](#) object with the table name and primary key value of the item you want. It returns a [GetItemResult](#) object.

You can use the returned `GetItemResult` object's `GetItem()` method to retrieve an `Aws::Map` of key `Aws::String` and value [AttributeValue](#) pairs associated with the item.

Includes

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/GetItemRequest.h>
#include <iostream>
```

Code

```
//! Get an item from an Amazon DynamoDB table.
/*!
 \sa getItem()
 \param tableName: The table name.
 \param partitionKey: The partition key.
 \param partitionValue: The value for the partition key.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
```

```
Aws::DynamoDB::Model::GetItemRequest request;

// Set up the request.
request.SetTableName(tableName);
request.AddKey(partitionKey,
               Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

// Retrieve the item's fields and values.
const Aws::DynamoDB::Model::GetItemOutcome &outcome =
dynamoClient.GetItem(request);
if (outcome.IsSuccess()) {
    // Reference the retrieved fields/values.
    const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
outcome.GetResult().GetItem();
    if (!item.empty()) {
        // Output each retrieved field and its value.
        for (const auto &i: item)
            std::cout << "Values: " << i.first << ": " << i.second.GetS()
                << std::endl;
    }
    else {
        std::cout << "No item found with the key " << partitionKey << std::endl;
    }
}
else {
    std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
}

return outcome.IsSuccess();
}
```

See the [complete example](#) on GitHub.

Add an Item to a Table

Create key `Aws::String` and value `AttributeValue` pairs that represent each item. These must include values for the table's primary key fields. If the item identified by the primary key already exists, its fields are *updated* by the request. Add them to the [PutItemRequest](#) using the `AddItem` method.

Includes

```
#include <aws/core/Aws.h>
```

```
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/PutItemRequest.h>
#include <aws/dynamodb/model/PutItemResult.h>
#include <iostream>
```

Code

```
//! Put an item in an Amazon DynamoDB table.
/*!
 \sa putItem()
 \param tableName: The table name.
 \param artistKey: The artist key. This is the partition key for the table.
 \param artistValue: The artist value.
 \param albumTitleKey: The album title key.
 \param albumTitleValue: The album title value.
 \param awardsKey: The awards key.
 \param awardsValue: The awards value.
 \param songTitleKey: The song title key.
 \param songTitleValue: The song title value.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
                               const Aws::String &awardsKey,
                               const Aws::String &awardsValue,
                               const Aws::String &songTitleKey,
                               const Aws::String &songTitleValue,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
        artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey, Aws::DynamoDB::Model::AttributeValue().SetS(
        albumTitleValue));
```

```

    putItemRequest.AddItem(awardsKey,
                          Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

    Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
        putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

See the [complete example](#) on GitHub.

Update an Existing Item in a Table

You can update an attribute for an item that already exists in a table by using the `DynamoDBClient`'s `UpdateItem` method, providing a table name, primary key value, and fields to update and their corresponding value.

Imports

```

#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/UpdateItemRequest.h>
#include <aws/dynamodb/model/UpdateItemResult.h>
#include <iostream>

```

Code

```

//! Update an Amazon DynamoDB table item.
/*!
    \sa updateItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.

```



```
\param attributeKey: The key for the attribute to be updated.
\param attributeValue: The value for the attribute to be updated.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/

/*
 * The example code only sets/updates an attribute value. It processes
 * the attribute value as a string, even if the value could be interpreted
 * as a number. Also, the example code does not remove an existing attribute
 * from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::String &attributeKey,
                                   const Aws::String &attributeValue,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
```

```
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
expressionAttributeValues[":valueA"] = attributeUpdatedValue;
request.SetExpressionAttributeValues(expressionAttributeValues);

// Update the item.
const Aws::DynamoDB::Model::UpdateItemOutcome &outcome = dynamoClient.UpdateItem(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Item was updated" << std::endl;
}
else {
    std::cerr << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

See the [complete example](#).

More Info

- [Guidelines for Working with Items](#) in the Amazon DynamoDB Developer Guide
- [Working with Items in DynamoDB](#) in the Amazon DynamoDB Developer Guide

Amazon EC2 examples using the AWS SDK for C++

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity—literally servers in Amazon’s data centers—that you use to build and host your software systems. You can use the following examples to program [Amazon EC2](#) using the AWS SDK for C++.

Note

Only the code that is necessary to demonstrate certain techniques is supplied in this Guide, but the [complete example code is available on GitHub](#). On GitHub you can download a single source file or you can clone the repository locally to get, build, and run all examples.

Topics

- [Managing Amazon EC2 Instances](#)

- [Using Elastic IP Addresses in Amazon EC2](#)
- [Using Regions and Availability Zones for Amazon EC2](#)
- [Working with Amazon EC2 Key Pairs](#)
- [Working with Security Groups in Amazon EC2](#)

Managing Amazon EC2 Instances

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create an Instance

Create a new Amazon EC2 instance by calling the EC2Client's RunInstances function, providing it with a [RunInstancesRequest](#) containing the [Amazon Machine Image \(AMI\)](#) to use and an [instance type](#).

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/CreateTagsRequest.h>
#include <aws/ec2/model/RunInstancesRequest.h>
#include <aws/ec2/model/RunInstancesResponse.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::RunInstancesRequest runRequest;
runRequest.SetImageId(amiId);
runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
runRequest.SetMinCount(1);
```

```
runRequest.SetMaxCount(1);

Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
    runRequest);
if (!runOutcome.IsSuccess()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();
if (instances.empty()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}
```

See the [complete example](#).

Start an Instance

To start an Amazon EC2 instance, call the EC2Client's `StartInstances` function, providing it with a [StartInstancesRequest](#) containing the ID of the instance to start.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/StartInstancesRequest.h>
#include <aws/ec2/model/StartInstancesResponse.h>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::StartInstancesRequest start_request;
start_request.AddInstanceIds(instanceId);
start_request.SetDryRun(true);

auto dry_run_outcome = ec2Client.StartInstances(start_request);
```

```
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    }
    else if (dry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    start_request.SetDryRun(false);
    auto start_instancesOutcome = ec2Client.StartInstances(start_request);

    if (!start_instancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
            start_instancesOutcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully started instance " << instanceId <<
            std::endl;
    }
}
```

See the [complete example](#).

Stop an Instance

To stop an Amazon EC2 instance, call the EC2Client's `StopInstances` function, providing it with a [StopInstancesRequest](#) containing the ID of the instance to stop.

Includes

```
#include <aws/ec2/model/StopInstancesRequest.h>
#include <aws/ec2/model/StopInstancesResponse.h>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::StopInstancesRequest request;
request.AddInstanceIds(instanceId);
```

```
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.StopInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to stop instance. A dry run should trigger an
error."
        << std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to stop instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
auto outcome = ec2Client.StopInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to stop instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully stopped instance " << instanceId <<
        std::endl;
}
}
```

See the [complete example](#).

Reboot an Instance

To reboot an Amazon EC2 instance, call the EC2Client's `RebootInstances` function, providing it with a [RebootInstancesRequest](#) containing the ID of the instance to reboot.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/RebootInstancesRequest.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::RebootInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.RebootInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to reboot on instance. A dry run should trigger an
error."
        <<
        std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to reboot instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
auto outcome = ec2Client.RebootInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to reboot instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully rebooted instance " << instanceId <<
        std::endl;
}
}
```

See the [complete example](#).

Describe Instances

To list your instances, create a [DescribeInstancesRequest](#) and call the EC2Client's `DescribeInstances` function. It will return a [DescribeInstancesResponse](#) object that you can use to list the Amazon EC2 instances for your AWS account and AWS Region.

Instances are grouped by *reservation*. Each reservation corresponds to the call to `StartInstances` that launched the instance. To list your instances, you must first call the

DescribeInstancesResponse class' GetReservations function, and then call getInstances on each returned Reservation object.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <aws/ec2/model/DescribeInstancesResponse.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeInstancesRequest request;
bool header = false;
bool done = false;
while (!done) {
    auto outcome = ec2Client.DescribeInstances(request);
    if (outcome.IsSuccess()) {
        if (!header) {
            std::cout << std::left <<
                std::setw(48) << "Name" <<
                std::setw(20) << "ID" <<
                std::setw(25) << "Ami" <<
                std::setw(15) << "Type" <<
                std::setw(15) << "State" <<
                std::setw(15) << "Monitoring" << std::endl;
            header = true;
        }

        const std::vector<Aws::EC2::Model::Reservation> &reservations =
            outcome.GetResult().GetReservations();

        for (const auto &reservation: reservations) {
            const std::vector<Aws::EC2::Model::Instance> &instances =
                reservation.GetInstances();
            for (const auto &instance: instances) {
                Aws::String instanceStateString =
                    Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                        instance.GetState().GetName());
            }
        }
    }
}
```



```

        Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
            instance.GetInstanceType());

        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
            instance.GetMonitoring().GetState());
        Aws::String name = "Unknown";

        const std::vector<Aws::EC2::Model::Tag> &tags = instance.GetTags();
        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
            [](const Aws::EC2::Model::Tag &tag) {
                return tag.GetKey() == "Name";
            });
        if (nameIter != tags.cend()) {
            name = nameIter->GetValue();
        }
        std::cout <<
            std::setw(48) << name <<
            std::setw(20) << instance.GetInstanceId() <<
            std::setw(25) << instance.GetImageId() <<
            std::setw(15) << typeString <<
            std::setw(15) << instanceStateString <<
            std::setw(15) << monitorString << std::endl;
    }
}

if (!outcome.GetResult().GetNextToken().empty()) {
    request.SetNextToken(outcome.GetResult().GetNextToken());
}
else {
    done = true;
}
}
else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}

```

Results are paged; you can get further results by passing the value returned from the result object's `GetNextToken` function to your original request object's `SetNextToken` function, then using the same request object in your next call to `DescribeInstances`.

See the [complete example](#).

Enable Instance Monitoring

You can monitor various aspects of your Amazon EC2 instances, such as CPU and network utilization, available memory, and disk space remaining. To learn more about instance monitoring, see [Monitoring Amazon EC2](#) in the Amazon EC2 User Guide.

To start monitoring an instance, you must create a [MonitorInstancesRequest](#) with the ID of the instance to monitor, and pass it to the `EC2Client`'s `MonitorInstances` function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/MonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesResponse.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::MonitorInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.MonitorInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to enable monitoring on instance. A dry run should
trigger an error."
        <<
        std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
```

```
        std::cerr << "Failed dry run to enable monitoring on instance " <<
            instanceId << ": " << dry_run_outcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    request.SetDryRun(false);
    auto monitorInstancesOutcome = ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
            instanceId << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully enabled monitoring on instance " <<
            instanceId << std::endl;
    }
}
```

See the [complete example](#).

Disable Instance Monitoring

To stop monitoring an instance, create an [UnmonitorInstancesRequest](#) with the ID of the instance to stop monitoring, and pass it to the EC2Client's `UnmonitorInstances` function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/MonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesResponse.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
unrequest.AddInstanceIds(instanceId);
unrequest.SetDryRun(true);

auto undryRunOutcome = ec2Client.UnmonitorInstances(unrequest);
```

```
    if (undryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to disable monitoring on instance. A dry run should
trigger an error."
            <<
            std::endl;
        return false;
    }
    else if (undryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instanceId << ": " << undryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    auto unmonitorInstancesOutcome = ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
            << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
            std::endl;
    }
    else {
        std::cout << "Successfully disable monitoring on instance " <<
            instanceId << std::endl;
    }
}
```

See the [complete example](#).

More Information

- [RunInstances](#) in the Amazon EC2 API Reference
- [DescribeInstances](#) in the Amazon EC2 API Reference
- [StartInstances](#) in the Amazon EC2 API Reference
- [StopInstances](#) in the Amazon EC2 API Reference
- [RebootInstances](#) in the Amazon EC2 API Reference
- [DescribeInstances](#) in the Amazon EC2 API Reference
- [MonitorInstances](#) in the Amazon EC2 API Reference
- [UnmonitorInstances](#) in the Amazon EC2 API Reference

Using Elastic IP Addresses in Amazon EC2

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Allocate an Elastic IP Address

To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.

To allocate an Elastic IP address, call the EC2Client's `AllocateAddress` function with an [AllocateAddressRequest](#) object containing the network type (classic EC2 or VPC).

Warning

We are retiring EC2-Classic on August 15, 2022. We recommend that you migrate from EC2-Classic to a VPC. For more information, see **Migrate from EC2-Classic to a VPC** in the the [Amazon EC2 User Guide for Linux Instances](#) or the [Amazon EC2 User Guide for Windows Instances](#). Also see the blog post [EC2-Classic Networking is Retiring – Here's How to Prepare](#).

The [AllocateAddressResponse](#) class in the response object contains an allocation ID that you can use to associate the address with an instance, by passing the allocation ID and instance ID in a [AssociateAddressRequest](#) to the EC2Client's `AssociateAddress` function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/AllocateAddressRequest.h>
#include <aws/ec2/model/AllocateAddressResponse.h>
#include <aws/ec2/model/AssociateAddressRequest.h>
```

```
#include <aws/ec2/model/AssociateAddressResponse.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::AllocateAddressRequest request;
request.SetDomain(Aws::EC2::Model::DomainType::vpc);

const Aws::EC2::Model::AllocateAddressOutcome outcome =
    ec2Client.AllocateAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to allocate Elastic IP address:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

allocationId = outcome.GetResult().GetAllocationId();

Aws::EC2::Model::AssociateAddressRequest associate_request;
associate_request.SetInstanceId(instanceId);
associate_request.SetAllocationId(allocationId);

const Aws::EC2::Model::AssociateAddressOutcome associate_outcome =
    ec2Client.AssociateAddress(associate_request);
if (!associate_outcome.IsSuccess()) {
    std::cerr << "Failed to associate Elastic IP address " << allocationId
        << " with instance " << instanceId << ":" <<
        associate_outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully associated Elastic IP address " << allocationId
    << " with instance " << instanceId << std::endl;
```

See the [complete example](#).

Describe Elastic IP Addresses

To list the Elastic IP addresses assigned to your account, call the EC2Client's `DescribeAddresses` function. It returns an outcome object that contains a [DescribeAddressesResponse](#) which you can use to get a list of [Address](#) objects that represent the Elastic IP addresses on your account.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeAddressesRequest.h>
#include <aws/ec2/model/DescribeAddressesResponse.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeAddressesRequest request;
auto outcome = ec2Client.DescribeAddresses(request);
if (outcome.IsSuccess()) {
    std::cout << std::left << std::setw(20) << "InstanceId" <<
        std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
        std::setw(30) << "Allocation ID" << std::setw(25) <<
        "NIC ID" << std::endl;

    const auto &addresses = outcome.GetResult().GetAddresses();
    for (const auto &address: addresses) {
        Aws::String domainString =
            Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                address.GetDomain());

        std::cout << std::left << std::setw(20) <<
            address.GetInstanceId() << std::setw(15) <<
            address.GetPublicIp() << std::setw(10) << domainString <<
            std::setw(30) << address.GetAllocationId() << std::setw(25)
            << address.GetNetworkInterfaceId() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe Elastic IP addresses:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

See the [complete example](#).

Release an Elastic IP Address

To release an Elastic IP address, call the EC2Client's `ReleaseAddress` function, passing it a [ReleaseAddressRequest](#) containing the allocation ID of the Elastic IP address you want to release.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/ReleaseAddressRequest.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2(clientConfiguration);

Aws::EC2::Model::ReleaseAddressRequest request;
request.SetAllocationId(allocationID);

auto outcome = ec2.ReleaseAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to release Elastic IP address " <<
        allocationID << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully released Elastic IP address " <<
        allocationID << std::endl;
}
```

After you release an Elastic IP address, it is released to the AWS IP address pool and might be unavailable to you afterward. Be sure to update your DNS records and any servers or devices that communicate with the address. If you attempt to release an Elastic IP address that you already released, you'll get an *AuthFailure* error if the address is already allocated to another AWS account.

If you are using a *default VPC*, then releasing an Elastic IP address automatically disassociates it from any instance that it's associated with. To disassociate an Elastic IP address without releasing it, use the EC2Client's `DisassociateAddress` function.

If you are using a non-default VPC, you *must* use `DisassociateAddress` to disassociate the Elastic IP address before you try to release it. Otherwise, Amazon EC2 returns an error (*InvalidIPAddress.InUse*).

See the [complete example](#).

More Information

- [Elastic IP Addresses](#) in the Amazon EC2 User Guide
- [AllocateAddress](#) in the Amazon EC2 API Reference
- [DescribeAddresses](#) in the Amazon EC2 API Reference
- [ReleaseAddress](#) in the Amazon EC2 API Reference

Using Regions and Availability Zones for Amazon EC2

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Describe Regions

To list the AWS Regions available to your AWS account, call the EC2Client's DescribeRegions function with a [DescribeRegionsRequest](#).

You will receive a [DescribeRegionsResponse](#) in the outcome object. Call its GetRegions function to get a list of [Region](#) objects that represent each Region.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeRegionsRequest.h>
#include <aws/ec2/model/DescribeRegionsResponse.h>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeRegionsRequest request;
auto outcome = ec2Client.DescribeRegions(request);
```

```
bool result = true;
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "RegionName" <<
        std::setw(64) << "Endpoint" << std::endl;

    const auto &regions = outcome.GetResult().GetRegions();
    for (const auto &region: regions) {
        std::cout << std::left <<
            std::setw(32) << region.GetRegionName() <<
            std::setw(64) << region.GetEndpoint() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe regions:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

See the [complete example](#).

Describe Availability Zones

To list each availability zone available to your account, call the EC2Client's `DescribeAvailabilityZones` function with a [DescribeAvailabilityZonesRequest](#).

You will receive a [DescribeAvailabilityZonesResponse](#) in the outcome object. Call its `GetAvailabilityZones` function to get a list of [AvailabilityZone](#) objects that represent each availability zone.

Includes

```
#include <aws/ec2/model/DescribeAvailabilityZonesRequest.h>
#include <aws/ec2/model/DescribeAvailabilityZonesResponse.h>
```

Code

```
Aws::EC2::Model::DescribeAvailabilityZonesRequest describe_request;
auto describe_outcome = ec2Client.DescribeAvailabilityZones(describe_request);

if (describe_outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "ZoneName" <<
```

```
        std::setw(20) << "State" <<
        std::setw(32) << "Region" << std::endl;

    const auto &zones =
        describe_outcome.GetResult().GetAvailabilityZones();

    for (const auto &zone: zones) {
        Aws::String stateString =

    Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
        zone.GetState());
        std::cout << std::left <<
            std::setw(32) << zone.GetZoneName() <<
            std::setw(20) << stateString <<
            std::setw(32) << zone.GetRegionName() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe availability zones:" <<
        describe_outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

See the [complete example](#).

More Information

- [Regions and Availability Zones](#) in the Amazon EC2 User Guide
- [DescribeRegions](#) in the Amazon EC2 API Reference
- [DescribeAvailabilityZones](#) in the Amazon EC2 API Reference

Working with Amazon EC2 Key Pairs

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create a Key Pair

To create a key pair, call the EC2Client's `CreateKeyPair` function with a [CreateKeyPairRequest](#) that contains the key's name.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/CreateKeyPairRequest.h>
#include <aws/ec2/model/CreateKeyPairResponse.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::CreateKeyPairRequest request;
request.SetKeyName(keyPairName);

Aws::EC2::Model::CreateKeyPairOutcome outcome = ec2Client.CreateKeyPair(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create key pair:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created key pair named " <<
        keyPairName << std::endl;
}
```

See the [complete example](#).

Describe Key Pairs

To list your key pairs or to get information about them, call the EC2Client's `DescribeKeyPairs` function with a [DescribeKeyPairsRequest](#).

You will receive a [DescribeKeyPairsResponse](#) that you can use to access the list of key pairs by calling its `GetKeyPairs` function, which returns a list of [KeyPairInfo](#) objects.

Includes

```
#include <aws/core/Aws.h>
```

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeKeyPairsRequest.h>
#include <aws/ec2/model/DescribeKeyPairsResponse.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeKeyPairsRequest request;

auto outcome = ec2Client.DescribeKeyPairs(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "Name" <<
        std::setw(64) << "Fingerprint" << std::endl;

    const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
        outcome.GetResult().GetKeyPairs();
    for (const auto &key_pair: key_pairs) {
        std::cout << std::left <<
            std::setw(32) << key_pair.GetKeyName() <<
            std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe key pairs:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

See the [complete example](#).

Delete a Key Pair

To delete a key pair, call the EC2Client's `DeleteKeyPair` function, passing it a [DeleteKeyPairRequest](#) that contains the name of the key pair to delete.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DeleteKeyPairRequest.h>
```

```
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteKeyPairRequest request;

request.SetKeyName(keyPairName);
const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete key pair " << keyPairName <<
        ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted key pair named " << keyPairName <<
        std::endl;
}
```

See the [complete example](#).

More Information

- [Amazon EC2 Key Pairs](#) in the Amazon EC2 User Guide
- [CreateKeyPair](#) in the Amazon EC2 API Reference
- [DescribeKeyPairs](#) in the Amazon EC2 API Reference
- [DeleteKeyPair](#) in the Amazon EC2 API Reference

Working with Security Groups in Amazon EC2

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create a Security Group

To create a security group, call the EC2Client's `CreateSecurityGroup` function with a [CreateSecurityGroupRequest](#) that contains the key's name.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/CreateSecurityGroupRequest.h>
#include <aws/ec2/model/CreateSecurityGroupResponse.h>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::CreateSecurityGroupRequest request;

request.SetGroupName(groupName);
request.SetDescription(description);
request.SetVpcId(vpcID);

const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
    ec2Client.CreateSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create security group:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully created security group named " << groupName <<
    std::endl;
```

See the [complete example](#).

Configure a Security Group

A security group can control both inbound (ingress) and outbound (egress) traffic to your Amazon EC2 instances.

To add ingress rules to your security group, use the EC2Client's `AuthorizeSecurityGroupIngress` function, providing the name of the security group and the access rules ([IpPermission](#)) you want to assign to it within an

[AuthorizeSecurityGroupIngressRequest](#) object. The following example shows how to add IP permissions to a security group.

Includes

```
#include <aws/ec2/model/AuthorizeSecurityGroupIngressRequest.h>
```

Code

```
Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest authorizeRequest;

authorizeRequest.SetGroupId(groupIDResult);
```

```
Aws::EC2::Model::IpRange ip_range;
ip_range.SetCidrIp("0.0.0.0/0");

Aws::EC2::Model::IpPermission permission1;
permission1.SetIpProtocol("tcp");
permission1.SetToPort(80);
permission1.SetFromPort(80);
permission1.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission1);

Aws::EC2::Model::IpPermission permission2;
permission2.SetIpProtocol("tcp");
permission2.SetToPort(22);
permission2.SetFromPort(22);
permission2.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission2);
```

```
const Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome authorizeOutcome =
    ec2Client.AuthorizeSecurityGroupIngress(authorizeRequest);

if (!authorizeOutcome.IsSuccess()) {
    std::cerr << "Failed to set ingress policy for security group " <<
        groupName << ":" << authorizeOutcome.GetError().GetMessage() <<
        std::endl;
    return false;
}
```



```
std::cout << "Successfully added ingress policy to security group " <<
    groupName << std::endl;
```

To add an egress rule to the security group, provide similar data in an [AuthorizeSecurityGroupEgressRequest](#) to the EC2Client's `AuthorizeSecurityGroupEgress` function.

See the [complete example](#).

Describe Security Groups

To describe your security groups or get information about them, call the EC2Client's `DescribeSecurityGroups` function with a [DescribeSecurityGroupsRequest](#).

You will receive a [DescribeSecurityGroupsResponse](#) in the outcome object that you can use to access the list of security groups by calling its `GetSecurityGroups` function, which returns a list of [SecurityGroup](#) objects.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeSecurityGroupsRequest.h>
#include <aws/ec2/model/DescribeSecurityGroupsResponse.h>
#include <iomanip>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeSecurityGroupsRequest request;

if (!groupID.empty()) {
    request.AddGroupIds(groupID);
}

Aws::String nextToken;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }
}
```

```
auto outcome = ec2Client.DescribeSecurityGroups(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "Name" <<
        std::setw(30) << "GroupId" <<
        std::setw(30) << "VpcId" <<
        std::setw(64) << "Description" << std::endl;

    const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
        outcome.GetResult().GetSecurityGroups();

    for (const auto &securityGroup: securityGroups) {
        std::cout << std::left <<
            std::setw(32) << securityGroup.GetGroupName() <<
            std::setw(30) << securityGroup.GetGroupId() <<
            std::setw(30) << securityGroup.GetVpcId() <<
            std::setw(64) << securityGroup.GetDescription() <<
            std::endl;
    }
}
else {
    std::cerr << "Failed to describe security groups:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());
```

See the [complete example](#).

Delete a Security Group

To delete a security group, call the EC2Client's `DeleteSecurityGroup` function, passing it a [DeleteSecurityGroupRequest](#) that contains the ID of the security group to delete.

Includes

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DeleteSecurityGroupRequest.h>
#include <iostream>
```

Code

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteSecurityGroupRequest request;

request.SetGroupId(securityGroupID);
auto outcome = ec2Client.DeleteSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete security group " << securityGroupID <<
        ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted security group " << securityGroupID <<
        std::endl;
}
```

See the [complete example](#).

More Information

- [Amazon EC2 Security Groups](#) in the Amazon EC2 User Guide
- [Authorizing Inbound Traffic for Your Linux Instances](#) in the Amazon EC2 User Guide
- [CreateSecurityGroup](#) in the Amazon EC2 API Reference
- [DescribeSecurityGroups](#) in the Amazon EC2 API Reference
- [DeleteSecurityGroup](#) in the Amazon EC2 API Reference
- [AuthorizeSecurityGroupIngress](#) in the Amazon EC2 API Reference

Amazon S3 code examples using the AWS SDK for C++

[Amazon S3](#) is object storage built to store and retrieve any amount of data from anywhere. There are multiple classes provided by the AWS SDK for C++ to interface with Amazon S3.

Note

Only the code that is necessary to demonstrate certain techniques is supplied in this Guide, but the [complete example code is available on GitHub](#). On GitHub you can download a single source file or you can clone the repository locally to get, build, and run all examples.

- [S3Client](#) class

The `S3Client` library is a fully-featured Amazon S3 interface.

The `list_buckets_disabling_dns_cache.cpp` example in this set is catered specifically to work with CURL on Linux/Mac (though can be modified to work on Windows). If you are on Windows, delete the file `list_buckets_disabling_dns_cache.cpp` before building the project because it relies on the `curl HttpClient` of Linux.

The example code utilizing the `S3Client` is in the [s3 folder](#) on Github. See the [Readme](#) on Github for a full list of functions demonstrated by this example set.

Portions of the `s3` example set are covered in additional detail in this guide:

- [Creating, listing, and deleting buckets](#)
- [Operations on objects](#) – Uploading and downloading data objects
- [Managing Amazon S3 Access Permissions](#)
- [Managing Access to Amazon S3 Buckets Using Bucket Policies](#)
- [Configuring an Amazon S3 Bucket as a Website](#)
- [S3CrtClient](#) class

The `S3CrtClient` was added in version 1.9 of the SDK. `S3CrtClient` provides high throughput for Amazon S3 GET (download) and PUT (upload) operations. The `S3CrtClient` is implemented on the top of the AWS Common Runtime (CRT) libraries.

The example code utilizing the `S3CrtClient` is in the [s3-crt folder](#) on Github. See the [Readme](#) on Github for a full list of functions demonstrated by this example set.

- [Using S3CrtClient for Amazon S3 operations](#)
- [TransferManager](#) class

`TransferManager` is a fully managed service that enables the transfer of files over the File Transfer Protocol (FTP), File Transfer Protocol over SSL (FTPS), or Secure Shell (SSH) File Transfer Protocol (SFTP) directly into and out of Amazon S3.

The example code utilizing the `TransferManager` is in the [transfer-manager folder](#) on Github. See the [Readme](#) on Github for a full list of functions demonstrated by this example set.

- [Using TransferManager for Amazon S3 operations](#)

Creating, listing, and deleting buckets

Every *object* or file in Amazon Simple Storage Service (Amazon S3) is contained in a *bucket*, which represents a folder of objects. Each bucket has a name that is globally unique within AWS. For more information, see [Working with Amazon S3 Buckets](#) in the Amazon Simple Storage Service User Guide.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

List buckets

To run the `list_buckets` example, at a command prompt, navigate to the folder where your build system creates your build executables. Run the executable like `run_list_buckets` (your full executable filename will differ based on your operating system). The output lists your account's buckets if you have any, or it displays an empty list if you don't have any buckets.

In `list_buckets.cpp`, there are two methods.

- `main()` calls `ListBuckets()`.
- `ListBuckets()` uses the SDK to query your buckets.

The `S3Client` object calls the SDK's `ListBuckets()` method. If successful, the method returns a `ListBucketOutcome` object, which contains a `ListBucketResult` object. The `ListBucketResult` object calls the `GetBuckets()` method to get a list of `Bucket` objects that contain information about each Amazon S3 bucket in your account.

Code

```
bool AwsDoc::S3::ListBuckets(const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();
```

```
bool result = true;
if (!outcome.IsSuccess()) {
    std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
    result = false;
}
else {
    std::cout << "Found " << outcome.GetResult().GetBuckets().size() << " buckets
\n";
    for (auto &&b: outcome.GetResult().GetBuckets()) {
        std::cout << b.GetName() << std::endl;
    }
}

return result;
}
```

See the complete [list_buckets example](#) on Github.

Create a bucket

To run the `create_bucket` example, at a command prompt, navigate to the folder where your build system creates your build executables. Run the executable like `run_create_bucket` (your full executable filename will differ based on your operating system). The code creates an empty bucket under your account and then displays the success or failure of the request.

In `create_bucket.cpp`, there are two methods.

- `main()` calls `CreateBucket()`. In `main()`, you need to change the AWS Region to the Region of your account by using the enum. You can view the Region of your account by logging into the [AWS Management Console](#), and locating the Region in the upper right-hand corner.
- `CreateBucket()` uses the SDK to create a bucket.

The `S3Client` object calls the SDK's `CreateBucket()` method, passing in a `CreateBucketRequest` with the bucket's name. By default, buckets are created in the *us-east-1* (N. Virginia) Region. If your Region is not *us-east-1* then the code sets up a bucket constraint to ensure the bucket is created in your Region.

Code

```
bool AwsDoc::S3::CreateBucket(const Aws::String &bucketName,
                              const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    //TODO(user): Change the bucket location constraint enum to your target Region.
    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

    Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: CreateBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the complete [create_buckets example](#) on Github.

Delete a bucket

To run the `delete_bucket` example, at a command prompt, navigate to the folder where your build system creates your build executables. Run the executable like `run_delete_bucket` (your full executable filename will differ based on your operating system). The code deletes the specified bucket in your account and then displays the success or failure of the request.

In `delete_bucket.cpp` there are two methods.

- `main()` calls `DeleteBucket()`. In `main()`, you need to change the AWS Region to the Region of your account by using the enum. You also need to change the `bucket_name` to the name of the bucket to delete.
- `DeleteBucket()` uses the SDK to delete the bucket.

The `S3Client` object uses the SDK's `DeleteBucket()` method, passing in a `DeleteBucketRequest` object with the name of the bucket to delete. The bucket must be empty to be successful.

Code

```
bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName,
                              const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the complete [delete_bucket example](#) on Github.

Operations on objects

An Amazon S3 object represents a *file*, which is a collection of data. Every object must reside within a [bucket](#).

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Upload a file to a bucket

Use the `S3Client` object `PutObject` function, supplying it with a bucket name, key name, and file to upload. `Aws::FStream` is used to upload the contents of the local file to the bucket. The bucket must exist or an error will result.

For an example on uploading objects asynchronously, see [Asynchronous methods](#)

Code

```
bool AwsDoc::S3::PutObject(const Aws::String &bucketName,
                          const Aws::String &fileName,
                          const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                     fileName.c_str(),
                                     std::ios_base::in | std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
```

```

        s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << fileName << "' to bucket '"
            << bucketName << "'.";
    }

    return outcome.IsSuccess();
}

```

See the [complete example](#) on Github.

Upload a string to a bucket

Use the `S3Client` object `PutObject` function, supplying it with a bucket name, key name, and file to upload. The bucket must exist or an error will result. This example differs from the previous one by using `Aws::StringStream` to upload an in-memory string data object directly to a bucket.

For an example on uploading objects asynchronously, see [Asynchronous methods](#)

Code

```

bool AwsDoc::S3::PutObjectBuffer(const Aws::String &bucketName,
                                const Aws::String &objectName,
                                const std::string &objectContent,
                                const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectName);

    const std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::StringStream>("");
    *inputData << objectContent.c_str();

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome = s3_client.PutObject(request);
}

```

```
if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutObjectBuffer: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Success: Object '" << objectName << "' with content '"
        << objectContent << "' uploaded to bucket '" << bucketName << "'.";
}

return outcome.IsSuccess();
}
```

See the [complete example](#) on Github.

List objects

To get a list of objects within a bucket, use the `S3Client` object `ListObjects` function. Supply it with a `ListObjectsRequest` that you set with the name of a bucket to list the contents of.

The `ListObjects` function returns a `ListObjectsOutcome` object that you can use to get a list of objects in the form of `Object` instances.

Code

```
bool AwsDoc::S3::ListObjects(const Aws::String &bucketName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3_client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
```

```

        outcome.GetError().GetMessage() << std::endl;
    return false;
}
else {
    Aws::Vector<Aws::S3::Model::Object> objects =
        outcome.GetResult().GetContents();

    allObjects.insert(allObjects.end(), objects.begin(), objects.end());
    continuationToken = outcome.GetResult().GetNextContinuationToken();
}
} while (!continuationToken.empty());

std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
}

return true;
}

```

See the [complete example](#) on Github.

Download an object

Use the `S3Client` object `GetObject` function, passing it a `GetObjectRequest` that you set with the name of a bucket and the object key to download. `GetObject` returns a [GetObjectOutcome](#) object that consists of a [GetObjectResult](#) and a [S3Error](#). `GetObjectResult` can be used to access the S3 object's data.

The following example downloads an object from Amazon S3. The object contents are stored in a local variable and the first line of the contents is output to the console.

Code

```

bool AwsDoc::S3::GetObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);
}

```

```

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

See the [complete example](#) on Github.

Delete an object

Use the `S3Client` object's `DeleteObject` function, passing it a `DeleteObjectRequest` that you set with the name of a bucket and object to download. *The specified bucket and object key must exist or an error will result.*

Code

```

bool AwsDoc::S3::DeleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
}

```

```
    }
    else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the [complete example](#) on Github.

Managing Amazon S3 Access Permissions

Access permissions for an Amazon S3 bucket or object are defined in an access control list (ACL). The ACL specifies the owner of the bucket/object and a list of grants. Each grant specifies a user (or grantee) and the user's permissions to access the bucket/object, such as READ or WRITE access.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Manage an Object's Access Control List

The access control list for an object can be retrieved by calling the `S3Client` method `GetObjectAcl`. The method accepts the names of the object and its bucket. The return value includes the ACL's Owner and list of Grants.

```
bool AwsDoc::S3::GetObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3_client.GetObjectAcl(request);
}
```

```
if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: GetObjectAcl: "
                << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
}
else {
    Aws::Vector<Aws::S3::Model::Grant> grants =
        outcome.GetResult().GetGrants();

    for (auto it = grants.begin(); it != grants.end(); it++) {
        std::cout << "For object " << objectKey << ": "
                    << std::endl << std::endl;

        Aws::S3::Model::Grant grant = *it;
        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                        << GetGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                        << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                        << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                        << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:         "
                        << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:   " <<
                    GetPermissionString(grant.GetPermission()) <<
```

```

        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \fn GetGranteeTypeString()
 \param type Type enumeration.
 */

Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \fn GetPermissionString()
 \param permission Permission enumeration.
 */

Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:

```



```

        return "Can read this object's permissions";
        // case Aws::S3::Model::Permission::WRITE // Not applicable.
    case Aws::S3::Model::Permission::WRITE_ACP:
        return "Can write this object's permissions";
    default:
        return "Permission unknown";
    }
}

```

The ACL can be modified by either creating a new ACL or changing the grants specified in the current ACL. The updated ACL becomes the new current ACL by passing it to the `PutObjectAcl` method.

The following code uses the ACL retrieved by `GetObjectAcl` and adds a new grant to it. The user or grantee is given READ permission for the object. The modified ACL is passed to `PutObjectAcl`, making it the new current ACL.

```

bool AwsDoc::S3::PutObjectAcl(const Aws::String &bucketName,
                             const Aws::String &objectKey,
                             const Aws::String &ownerID,
                             const Aws::String &granteePermission,
                             const Aws::String &granteeType,
                             const Aws::String &granteeID,
                             const Aws::Client::ClientConfiguration &clientConfig,
                             const Aws::String &granteeDisplayName,
                             const Aws::String &granteeEmailAddress,
                             const Aws::String &granteeURI) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(SetGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }
}

```

```

    if (!granteeDisplayName.empty()) {
        grantee.SetDisplayName(granteeDisplayName);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(SetGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3_client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: PutObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \sa SetGranteePermission()

```

```

    \param access Human readable string.
    */

Aws::S3::Model::Permission SetGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
    \sa SetGranteeType()
    \param type Human readable string.
    */

Aws::S3::Model::Type SetGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

See the [complete example](#) on Github.

Manage a Bucket's Access Control List

In most cases, the preferred method for setting the access permissions of a bucket is to define a bucket policy. However, buckets also support access control lists for users who wish to use them.

Management of an access control list for a bucket is identical to that used for an object. The `GetBucketAcl` method retrieves a bucket's current ACL and `PutBucketAcl` applies a new ACL to the bucket.

The following code demonstrates getting and setting a bucket ACL.

```
//! Routine which demonstrates setting the ACL for an S3 bucket.
/*!
  \sa GetPutBucketAcl()
  \param bucketName Name of a bucket.
  \param ownerID The canonical ID of the bucket owner.
  See https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html for more information.
  \param granteePermission The access level to enable for the grantee.
  \param granteeType The type of grantee.
  \param granteeID The canonical ID of the grantee.
  \param clientConfig Aws client configuration.
  \param granteeDisplayName The display name of the grantee.
  \param granteeEmailAddress The email address associated with the grantee's AWS account.
  \param granteeURI The URI of a built-in access group.
*/

bool AwsDoc::S3::GetPutBucketAcl(const Aws::String &bucketName,
                                const Aws::String &ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType,
                                const Aws::String &granteeID,
                                const Aws::Client::ClientConfiguration &clientConfig,
                                const Aws::String &granteeDisplayName,
                                const Aws::String &granteeEmailAddress,
                                const Aws::String &granteeURI) {
    bool result = ::PutBucketAcl(bucketName, ownerID, granteePermission, granteeType,
                                granteeID, clientConfig, granteeDisplayName,
                                granteeEmailAddress,
                                granteeURI);

    if (result) {
        result = ::GetBucketAcl(bucketName, clientConfig);
    }

    return result;
}
```

```
//! Routine which demonstrates setting the ACL for an S3 bucket.
/*!
  \sa PutBucketAcl()
  \param bucketName Name of from bucket.
  \param ownerID The canonical ID of the bucket owner.
  See https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html for more information.
  \param granteePermission The access level to enable for the grantee.
  \param granteeType The type of grantee.
  \param granteeID The canonical ID of the grantee.
  \param clientConfig Aws client configuration.
  \param granteeDisplayName The display name of the grantee.
  \param granteeEmailAddress The email address associated with the grantee's AWS account.
  \param granteeURI The URI of a built-in access group.
*/

bool PutBucketAcl(const Aws::String &bucketName,
                 const Aws::String &ownerID,
                 const Aws::String &granteePermission,
                 const Aws::String &granteeType,
                 const Aws::String &granteeID,
                 const Aws::Client::ClientConfiguration &clientConfig,
                 const Aws::String &granteeDisplayName,
                 const Aws::String &granteeEmailAddress,
                 const Aws::String &granteeURI) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(SetGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeDisplayName.empty()) {
```

```

        grantee.SetDisplayName(granteeDisplayName);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(SetGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3_client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: PutBucketAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
            << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which demonstrates getting the ACL for an S3 bucket.
/*!
    \sa GetBucketAcl()
    \param bucketName Name of the s3 bucket.
    \param clientConfig Aws client configuration.

```

```

*/

bool GetBucketAcl(const Aws::String &bucketName,
                 const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3_client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << GetGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {

```

```

        std::cout << "ID:          "
                  << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI:          "
                  << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission:    " <<
              GetPermissionString(grant.GetPermission()) <<
              std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \sa GetPermissionString()
 \param permission Permission enumeration.
 */

Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                  "objects in this bucket, and read/write this "
                  "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
}

```



```
    return "Permission unknown";
}

//! Routine which converts a human-readable string to a built-in type enumeration
/*!
 \sa SetGranteePermission()
 \param access Human readable string.
 */

Aws::S3::Model::Permission SetGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \sa GetGranteeTypeString()
 \param type Type enumeration.
 */

Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}
}
```

```
Aws::S3::Model::Type SetGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

See the [complete example](#) on Github.

Managing Access to Amazon S3 Buckets Using Bucket Policies

You can set, get, or delete a *bucket policy* to manage access to your Amazon S3 buckets.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Set a Bucket Policy

You can set the bucket policy for a particular S3 bucket by calling the `S3Client`'s `PutBucketPolicy` function and providing it with the bucket name and policy's JSON representation in a [PutBucketPolicyRequest](#).

Code

```
//! Build a policy JSON string.
/*
    \sa GetPolicyString()
    \param userArn Aws user Amazon Resource Name (ARN).
        For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
    \param bucketName Name of a bucket.
*/
```

```

Aws::String GetPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"      }, \n"
        "      \"Action\": [ \"s3:GetObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

```

bool AwsDoc::S3::PutBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3_client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```
else {
    std::cout << "Set the following policy body for the bucket '" <<
        bucketName << "':" << std::endl << std::endl;
    std::cout << policyBody << std::endl;
}

return outcome.IsSuccess();
}
```

Note

The [Aws::Utils::Json::JsonValue](#) utility class can be used to help you construct valid JSON objects to pass to `PutBucketPolicy`.

See the [complete example](#) on Github.

Get a Bucket Policy

To retrieve the policy for an Amazon S3 bucket, call the `S3Client`'s `GetBucketPolicy` function, passing it the name of the bucket in a [GetBucketPolicyRequest](#).

Code

```
bool AwsDoc::S3::GetBucketPolicy(const Aws::String &bucketName,
                                const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3_client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketPolicy: "
            << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
```

```
Aws::StringStream policy_stream;
Aws::String line;

outcome.GetResult().GetPolicy() >> line;
policy_stream << line;

std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
    policy_stream.str() << std::endl;
}

return outcome.IsSuccess();
}
```

See the [complete example](#) on Github.

Delete a Bucket Policy

To delete a bucket policy, call the S3Client's DeleteBucketPolicy function, providing it with the bucket name in a [DeleteBucketPolicyRequest](#).

Code

```
bool AwsDoc::S3::DeleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
    client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
}
```

This function succeeds even if the bucket doesn't already have a policy. If you specify a bucket name that doesn't exist or if you don't have access to the bucket, an `AmazonServiceException` is thrown.

See the [complete example](#) on Github.

More Info

- [PutBucketPolicy](#) in the Amazon Simple Storage Service API Reference
- [GetBucketPolicy](#) in the Amazon Simple Storage Service API Reference
- [DeleteBucketPolicy](#) in the Amazon Simple Storage Service API Reference
- [Access Policy Language Overview](#) in the Amazon Simple Storage Service User Guide
- [Bucket Policy Examples](#) in the Amazon Simple Storage Service User Guide

Configuring an Amazon S3 Bucket as a Website

You can configure an Amazon S3 bucket to behave as a website. To do this, you need to set its website configuration.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Set a Bucket's Website Configuration

To set an Amazon S3 bucket's website configuration, call the `S3Client`'s `PutBucketWebsite` function with a [PutBucketWebsiteRequest](#) object containing the bucket name and its website configuration, provided in a [WebsiteConfiguration](#) object.

Setting an index document is *required*; all other parameters are optional.

Code

```
bool AwsDoc::S3::PutWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPath, const Aws::String
&errorPage,
                                   const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

Note

Setting a website configuration does not modify the access permissions for your bucket. To make your files visible on the web, you will also need to set a *bucket policy* that allows

public read access to the files in the bucket. For more information, see [Managing Access to Amazon S3 Buckets Using Bucket Policies](#).

See the [complete example](#) on Github.

Get a Bucket's Website Configuration

To get an Amazon S3 bucket's website configuration, call the S3Client's `GetBucketWebsite` function with a [GetBucketWebsiteRequest](#) containing the name of the bucket to retrieve the configuration for.

The configuration will be returned as a [GetBucketWebsiteResult](#) object within the outcome object. If there is no website configuration for the bucket, then `null` will be returned.

Code

```
bool AwsDoc::S3::GetWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3_client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    }
    else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
```



```
        << websiteResult.GetIndexDocument().GetSuffix()
        << std::endl
        << "Error page: "
        << websiteResult.GetErrorDocument().GetKey()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the [complete example](#) on Github.

Delete a Bucket's Website Configuration

To delete an Amazon S3 bucket's website configuration, call the `S3Client`'s `DeleteBucketWebsite` function with a [DeleteBucketWebsiteRequest](#): containing the name of the bucket to delete the configuration from.

Code

```
bool AwsDoc::S3::DeleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

See the [complete example](#) on Github.

More Information

- [PUT Bucket website](#) in the Amazon Simple Storage Service API Reference
- [GET Bucket website](#) in the Amazon Simple Storage Service API Reference
- [DELETE Bucket website](#) in the Amazon Simple Storage Service API Reference

Using TransferManager for Amazon S3 operations

You can use the AWS SDK for C++ `TransferManager` class to reliably transfer files from the local environment to Amazon S3 and to copy objects from one Amazon S3 location to another. `TransferManager` can get the progress of a transfer and pause or resume uploads and downloads.

Note

To avoid being charged for incomplete or partial uploads, we recommend that you enable the [AbortIncompleteMultipartUpload](#) lifecycle rule on your Amazon S3 buckets. This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data. For more information, see [Setting lifecycle configuration on a bucket](#) in the Amazon S3 User Guide.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Upload and download of object using TransferManager

This example demonstrates how [TransferManager](#) transfers large object in memory. `UploadFile` and `DownloadFile` methods are both called asynchronously and return a `TransferHandle` to manage the status of your request. If the object uploaded is larger than

`bufferSize` then a multipart upload will be performed. The `bufferSize` defaults to 5MB, but this can be configured via [TransferManagerConfiguration](#).

```
auto s3_client = Aws::MakeShared<Aws::S3::S3Client>("S3Client");
auto executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>("executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = s3_client;

// Create buffer to hold data received by the data stream.
    Aws::Utils::Array<unsigned char> buffer(BUFFER_SIZE);

// The local variable 'streamBuffer' is captured by reference in a lambda.
// It must persist until all downloading by the 'transfer_manager' is complete.
    Stream::PreallocatedStreamBuf streamBuffer(buffer.GetUnderlyingData(),
    buffer.GetLength());

    auto transfer_manager =
    Aws::Transfer::TransferManager::Create(transfer_config);

    auto uploadHandle = transfer_manager->UploadFile(LOCAL_FILE, BUCKET, KEY,
    "text/plain", Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success = uploadHandle->GetStatus() ==
    Transfer::TransferStatus::COMPLETED;

    if (!success)
    {
        auto err = uploadHandle->GetLastError();
        std::cout << "File upload failed: " << err.GetMessage() << std::endl;
    }
    else
    {
        std::cout << "File upload finished." << std::endl;

        // Verify that the upload retrieved the expected amount of data.
        assert(uploadHandle->GetBytesTotalSize() == uploadHandle-
        >GetBytesTransferred());

        auto downloadHandle = transfer_manager->DownloadFile(BUCKET,
            KEY,
            [&]() { //Define a lambda expression for the callback method parameter
            to stream back the data.
```

```
        return Aws::New<MyUnderlyingStream>("TestTag", &streamBuffer);
    });
    downloadHandle->WaitUntilFinished();// Block calling thread until download
is complete.
    auto downStat = downloadHandle->GetStatus();
    if (downStat != Transfer::TransferStatus::COMPLETED)
    {
        auto err = downloadHandle->GetLastError();
        std::cout << "File download failed: " << err.GetMessage() <<
std::endl;
    }
    std::cout << "File download to memory finished." << std::endl;
```

See the [complete example](#) on Github.

Using S3CrtClient for Amazon S3 operations

The S3CrtClient class is available in version 1.9 of the AWS SDK for C++ and improves the throughput of uploading and downloading large data files to and from Amazon S3. For more information on the improvements of this release, see [Improving Amazon S3 Throughput with AWS SDK for C++ v1.9](#)

The S3CrtClient is implemented on the top of the [AWS Common Runtime \(CRT\) libraries](#).

Note

To avoid being charged for incomplete or partial uploads, we recommend that you enable the [AbortIncompleteMultipartUpload](#) lifecycle rule on your Amazon S3 buckets. This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data. For more information, see [Setting lifecycle configuration on a bucket](#) in the Amazon S3 User Guide.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Upload and download of object using `S3CrtClient`

This example demonstrates how to use the [S3CrtClient](#). The example creates a bucket, uploads an object, downloads the object, then deletes the file and the bucket. A PUT operation turns into a multipart upload. A GET operation turns into multiple "ranged" GET requests. For more information on multipart uploads, see [Uploading and copying objects using multipart upload](#) in the Amazon S3 User Guide.

The provided data file, `ny.json`, gets uploaded as a multipart upload in this example. This can be confirmed by viewing the debug logs after a successful run of the program.

If the upload fails, an `AbortMultipartUpload` is issued in the underlying CRT library to clean up any already-uploaded parts. However, not all failures can be handled internally (such as a network cable being unplugged). It is recommended to create a lifecycle rule on your Amazon S3 bucket to ensure partially uploaded data does not linger on your account (partially uploaded data is still billable). To find out how to set up a lifecycle rule, see [Discovering and Deleting Incomplete Multipart Uploads to Lower Amazon S3 Costs](#).

Using the debug log to explore multipart upload details

1. In `main()`, note that there are "TODO" comments with instructions for updating the code.
 - a. For `file_name`: From the link provided in the code comment download sample data file `ny.json`, or use a large data file of your own.
 - b. For `region`: Update the `region` variable, using the enum, to the AWS Region of your account. To find your Region of your account, log into the AWS Management Console, and locate the Region in the upper right-hand corner.
2. Build the example.
3. Copy the file specified by variable `file_name` to your executable folder and run the `s3-crt-demo` executable.
4. In your executable folder, find the most recent `.log` file.
5. Open the log file, select **search**, and enter **partNumber**.
6. The log contains entries similar to the following, where the `partNumber` and `uploadId` are specified for each portion of the uploaded file:

```
PUT /my-object
```

```
partNumber=1&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8  
content-length:8388608 host:my-bucketasdfasdf.s3.us-  
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

and

```
PUT /my-object
```

```
partNumber=2&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8  
content-length:8388608 host:my-bucketasdfasdf.s3.us-  
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

See the [complete example](#) on Github.

Amazon SQS code examples using the AWS SDK for C++

Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications. You can use the following examples to program [Amazon SQS](#) using the AWS SDK for C++.

Note

Only the code that is necessary to demonstrate certain techniques is supplied in this Guide, but the [complete example code is available on GitHub](#). On GitHub you can download a single source file or you can clone the repository locally to get, build, and run all examples.

Topics

- [Working with Amazon SQS Message Queues](#)
- [Sending, Receiving, and Deleting Amazon SQS Messages](#)
- [Enabling Long Polling for Amazon SQS Message Queues](#)
- [Setting Visibility Timeout in Amazon SQS](#)
- [Using Dead Letter Queues in Amazon SQS](#)

Working with Amazon SQS Message Queues

A *message queue* is the logical container you use to send messages reliably in Amazon SQS. There are two types of queues: *standard* and *first-in, first-out* (FIFO). To learn more about queues and the differences between these types, see the [Amazon Simple Queue Service Developer Guide](#).

These C++ examples show you how to use the AWS SDK for C++ to create, list, delete, and get the URL of an Amazon SQS queue.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create a Queue

Use the `SQSClient` class `CreateQueue` member function, and provide it with a [CreateQueueRequest](#) object that describes the queue parameters.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::CreateQueueRequest request;
request.SetQueueName(queueName);

const Aws::SQS::Model::CreateQueueOutcome outcome = sqsClient.CreateQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created queue " << queueName << " with a queue URL "
              << outcome.GetResult().GetQueueUrl() << "." << std::endl;
```

```
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
}
```

See the [complete example](#).

List Queues

To list Amazon SQS queues for your account, call the `SQSClient` class `ListQueues` member function, and pass it a [ListQueuesRequest](#) object.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

Aws::String nextToken; // Used for pagination.
Aws::Vector<Aws::String> allQueueUrls;

do {
    if (!nextToken.empty()) {
        listQueuesRequest.SetNextToken(nextToken);
    }
    const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
        listQueuesRequest);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::String> &queueUrls =
outcome.GetResult().GetQueueUrls();
        allQueueUrls.insert(allQueueUrls.end(),
            queueUrls.begin(),
            queueUrls.end());

        nextToken = outcome.GetResult().GetNextToken();
    }
}
```



```
    }
    else {
        std::cerr << "Error listing queues: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

} while (!nextToken.empty());

std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
for (const auto &iter: allQueueUrls) {
    std::cout << " " << iter << std::endl;
}
}
```

See the [complete example](#).

Get the Queue's URL

To get the URL for an existing Amazon SQS queue, call the `SQSClient` class `GetQueueUrl` member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/GetQueueUrlRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::GetQueueUrlRequest request;
request.SetQueueName(queueName);

const Aws::SQS::Model::GetQueueUrlOutcome outcome = sqsClient.GetQueueUrl(request);
if (outcome.IsSuccess()) {
    std::cout << "Queue " << queueName << " has url " <<
        outcome.GetResult().GetQueueUrl() << std::endl;
}
else {
    std::cerr << "Error getting url for queue " << queueName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

```
}
```

See the [complete example](#).

Delete a Queue

Provide the [URL](#) to the SQSClient class DeleteQueue member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/core/client/DefaultRetryStrategy.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteQueueRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::Model::DeleteQueueRequest request;
request.SetQueueUrl(queueURL);

const Aws::SQS::Model::DeleteQueueOutcome outcome = sqsClient.DeleteQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted queue with url " << queueURL <<
        std::endl;
}
else {
    std::cerr << "Error deleting queue " << queueURL << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

See the [complete example](#).

More Info

- [How Amazon SQS Queues Work](#) in the Amazon Simple Queue Service Developer Guide
- [CreateQueue](#) in the Amazon Simple Queue Service API Reference
- [GetQueueUrl](#) in the Amazon Simple Queue Service API Reference
- [ListQueues](#) in the Amazon Simple Queue Service API Reference
- [DeleteQueues](#) in the Amazon Simple Queue Service API Reference

Sending, Receiving, and Deleting Amazon SQS Messages

Messages are always delivered using an [SQS queue](#). These C++ examples show you how to use the AWS SDK for C++ to send, receive, and delete Amazon SQS messages from SQS queues.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Send a Message

You can add a single message to an Amazon SQS queue by calling the `SQSClient` class `SendMessage` member function. You provide `SendMessage` with a [SendMessageRequest](#) object containing the queue's [URL](#), the message body, and an optional delay value (in seconds).

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SendMessageRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::SendMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMessageBody(messageBody);

const Aws::SQS::Model::SendMessageOutcome outcome = sqsClient.SendMessage(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully sent message to " << queueUrl <<
        std::endl;
}
else {
```

```
std::cerr << "Error sending message to " << queueUrl << ": " <<
    outcome.GetError().GetMessage() << std::endl;
}
```

See the [complete example](#).

Receive Messages

Retrieve any messages that are currently in the queue by calling the `SQSClient` class `ReceiveMessage` member function, passing it the queue's URL. Messages are returned as a list of [Message](#) objects.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::ReceiveMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMaxNumberOfMessages(1);

const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
    request);
if (outcome.IsSuccess()) {

    const Aws::Vector<Aws::SQS::Model::Message> &messages =
        outcome.GetResult().GetMessages();
    if (!messages.empty()) {
        const Aws::SQS::Model::Message &message = messages[0];
        std::cout << "Received message:" << std::endl;
        std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
        std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
        std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
    }
    else {
        std::cout << "No messages received from queue " << queueUrl <<
```

```
        std::endl;

    }
}
else {
    std::cerr << "Error receiving message from queue " << queueUrl << ": "
              << outcome.GetError().GetMessage() << std::endl;
}
}
```

See the [complete example](#).

Delete Messages after Receipt

After receiving a message and processing its contents, delete the message from the queue by sending the message's receipt handle and the queue URL to the SQSClient class DeleteMessage member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteMessageRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::Model::DeleteMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetReceiptHandle(messageReceiptHandle);

const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted message from queue " << queueUrl
              << std::endl;
}
else {
    std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
              outcome.GetError().GetMessage() << std::endl;
}
}
```

See the [complete example](#).

More Info

- [How Amazon SQS Queues Work](#) in the Amazon Simple Queue Service Developer Guide
- [SendMessage](#) in the Amazon Simple Queue Service API Reference
- [SendMessageBatch](#) in the Amazon Simple Queue Service API Reference
- [ReceiveMessage](#) in the Amazon Simple Queue Service API Reference
- [DeleteMessage](#) in the Amazon Simple Queue Service API Reference

Enabling Long Polling for Amazon SQS Message Queues

Amazon SQS uses *short polling* by default, querying only a subset of the servers—based on a weighted random distribution—to determine whether any messages are available for inclusion in the response.

Long polling helps reduce your cost of using Amazon SQS by reducing the number of empty responses when there are no messages available to return in reply to a `ReceiveMessage` request sent to an Amazon SQS queue and eliminating false empty responses. You can set a long polling frequency from *1–20 seconds*.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Enable Long Polling when Creating a Queue

To enable long polling when creating an Amazon SQS queue, set the `ReceiveMessageWaitTimeSeconds` attribute on the [CreateQueueRequest](#) object before calling the `SQSClient` class' `CreateQueue` member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
```

```
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::CreateQueueRequest request;
request.SetQueueName(queueName);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
    pollTimeSeconds);

const Aws::SQS::Model::CreateQueueOutcome outcome = sqsClient.CreateQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created queue " << queueName <<
        std::endl;
}
else {
    std::cout << "Error creating queue " << queueName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

See the [complete example](#).

Enable Long Polling on an Existing Queue

In addition to enabling long polling when creating a queue, you can also enable it on an existing queue by setting `ReceiveMessageWaitTimeSeconds` on the [SetQueueAttributesRequest](#) before calling the `SQSClient` class' `SetQueueAttributes` member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
Aws::SQS::Model::SetQueueAttributesRequest request;
request.SetQueueUrl(queueURL);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
    pollTimeSeconds);

const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
sqscClient.SetQueueAttributes(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully updated long polling time for queue " <<
        queueURL << " to " << pollTimeSeconds << std::endl;
}
else {
    std::cout << "Error updating long polling time for queue " <<
        queueURL << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
}
```

See the [complete example](#).

Enable Long Polling on Message Receipt

You can enable long polling when receiving a message by setting the wait time in seconds on the [ReceiveMessageRequest](#) that you supply to the SQSClient class' ReceiveMessage member function.

Note

You should make sure that the AWS client's request timeout is larger than the maximum long poll time (20s) so that your ReceiveMessage requests don't time out while waiting for the next poll event!

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
```

Code


```
Aws::SQS::SQSClient sqsClient(customConfiguration);

Aws::SQS::Model::ReceiveMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMaxNumberOfMessages(1);
request.SetWaitTimeSeconds(waitTimeSeconds);

auto outcome = sqsClient.ReceiveMessage(request);
if (outcome.IsSuccess()) {
    const auto &messages = outcome.GetResult().GetMessages();
    if (messages.empty()) {
        std::cout << "No messages received from queue " << queueUrl <<
            std::endl;
    }
    else {
        const auto &message = messages[0];
        std::cout << "Received message:" << std::endl;
        std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
        std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
        std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
    }
}
else {
    std::cout << "Error receiving message from queue " << queueUrl << ": "
        << outcome.GetError().GetMessage() << std::endl;
}
}
```

See the [complete example](#).

More Info

- [Amazon SQS Long Polling](#) in the Amazon Simple Queue Service Developer Guide
- [CreateQueue](#) in the Amazon Simple Queue Service API Reference
- [ReceiveMessage](#) in the Amazon Simple Queue Service API Reference
- [SetQueueAttributes](#) in the Amazon Simple Queue Service API Reference

Setting Visibility Timeout in Amazon SQS

When a message is received in Amazon SQS, it remains on the queue until it's deleted in order to ensure receipt. A message that was received, but not deleted, will be available in subsequent

requests after a given *visibility timeout* to help prevent the message from being received more than once before it can be processed and deleted.

When using [standard queues](#), visibility timeout isn't a guarantee against receiving a message twice. If you are using a standard queue, be sure that your code can handle the case where the same message has been delivered more than once.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Set the Message Visibility Timeout upon Message Receipt

When you have received a message, you can modify its visibility timeout by passing its receipt handle in a [ChangeMessageVisibilityRequest](#) that you pass to the `SQSClient` class' `ChangeMessageVisibility` member function.

Includes

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ChangeMessageVisibilityRequest.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::Model::ChangeMessageVisibilityRequest request;
request.SetQueueUrl(queue_url);
request.SetReceiptHandle(messageReceiptHandle);
request.SetVisibilityTimeout(visibilityTimeoutSeconds);

auto outcome = sqsClient.ChangeMessageVisibility(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully changed visibility of message " <<
        messageReceiptHandle << " from queue " << queue_url << std::endl;
```

```
    }  
    else {  
        std::cout << "Error changing visibility of message from queue "  
                  << queue_url << ": " <<  
                  << outcome.GetError().GetMessage() << std::endl;  
    }  
}
```

See the [complete example](#).

More Info

- [Visibility Timeout](#) in the Amazon Simple Queue Service Developer Guide
- [SetQueueAttributes](#) in the Amazon Simple Queue Service API Reference
- [GetQueueAttributes](#) in the Amazon Simple Queue Service API Reference
- [ReceiveMessage](#) in the Amazon Simple Queue Service API Reference
- [ChangeMessageVisibility](#) in the Amazon Simple Queue Service API Reference
- [ChangeMessageVisibilityBatch](#) in the Amazon Simple Queue Service API Reference

Using Dead Letter Queues in Amazon SQS

Amazon SQS provides support for *dead letter queues*. A dead letter queue is a queue that other queues can target for messages that can't be processed successfully. You can set aside and isolate these messages in the dead letter queue to determine why their processing did not succeed.

To create a dead letter queue, you must first create a *redrive policy*, and then set the policy in the queue's attributes.

Important

A dead letter queue must be the same type of queue (FIFO or standard) that the source queue is. It must also be created using the same AWS account and AWS Region as the source queue.

Prerequisites

Before you begin, we recommend you read [Getting started using the AWS SDK for C++](#).

Download the example code and build the solution as described in [Get started on code examples](#).

To run the examples, the user profile your code uses to make the requests must have proper permissions in AWS (for the service and the action). For more information, see [Providing AWS credentials](#).

Create a Redrive Policy

A redrive policy is specified in JSON. To create it, you can use the JSON utility class provided with the AWS SDK for C++.

Here is an example function that creates a redrive policy by providing it with the ARN of your dead letter queue and the maximum number of times the message can be received and not processed before it's sent to the dead letter queue.

Includes

```
#include <aws/core/Aws.h>
#include <aws/core/utils/json/JsonSerializer.h>
```

Code

```
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}
```

See the [complete example](#).

Set the Redrive Policy on your Source Queue

To finish setting up your dead letter queue, call the SQSClient class' `SetQueueAttributes` member function with a [SetQueueAttributesRequest](#) object for which you've set the `RedrivePolicy` attribute with your JSON redrive policy.

Includes

```
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

Code

```
Aws::SQS::Model::SetQueueAttributesRequest request;
request.SetQueueUrl(srcQueueUrl);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
    redrivePolicy);

const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully set dead letter queue for queue " <<
        srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
}
else {
    std::cerr << "Error setting dead letter queue for queue " <<
        srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
```

See the [complete example](#).

More Info

- [Using Amazon SQS Dead Letter Queues](#) in the Amazon Simple Queue Service Developer Guide
- [SetQueueAttributes](#) in the Amazon Simple Queue Service API Reference

SDK for C++ code examples

The code examples in this topic show you how to use the AWS SDK for C++ with AWS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Cross-service examples are sample applications that work across multiple AWS services.

Examples

- [Actions and scenarios using SDK for C++](#)
- [Cross-service examples using SDK for C++](#)

Actions and scenarios using SDK for C++

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS services.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Services

- [ACM examples using SDK for C++](#)
- [Aurora examples using SDK for C++](#)
- [Auto Scaling examples using SDK for C++](#)
- [CloudTrail examples using SDK for C++](#)
- [CloudWatch examples using SDK for C++](#)
- [CloudWatch Logs examples using SDK for C++](#)

- [CodeBuild examples using SDK for C++](#)
- [Amazon Cognito Identity Provider examples using SDK for C++](#)
- [DynamoDB examples using SDK for C++](#)
- [Amazon EC2 examples using SDK for C++](#)
- [EventBridge examples using SDK for C++](#)
- [AWS Glue examples using SDK for C++](#)
- [HealthImaging examples using SDK for C++](#)
- [IAM examples using SDK for C++](#)
- [AWS IoT examples using SDK for C++](#)
- [AWS IoT data examples using SDK for C++](#)
- [Lambda examples using SDK for C++](#)
- [MediaConvert examples using SDK for C++](#)
- [Amazon RDS examples using SDK for C++](#)
- [Amazon Rekognition examples using SDK for C++](#)
- [Amazon S3 examples using SDK for C++](#)
- [Secrets Manager examples using SDK for C++](#)
- [Amazon SES examples using SDK for C++](#)
- [Amazon SNS examples using SDK for C++](#)
- [Amazon SQS examples using SDK for C++](#)
- [AWS STS examples using SDK for C++](#)
- [Amazon Transcribe examples using SDK for C++](#)

ACM examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with ACM.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

AddTagsToCertificate

The following code example shows how to use AddTagsToCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Add tags to an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::addTagsToCertificate(const Aws::String &certificateArn,
                                       const Aws::String &tagKey,
                                       const Aws::String &tagValue,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::AddTagsToCertificateRequest request;
    Aws::Vector<Aws::ACM::Model::Tag> tags;
    Aws::ACM::Model::Tag tag;

    tag.WithKey(tagKey).WithValue(tagValue);
    tags.push_back(tag);
```



```

request.WithCertificateArn(certificateArn).WithTags(tags);

Aws::ACM::Model::AddTagsToCertificateOutcome outcome =
    acmClient.AddTagsToCertificate(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: addTagsToCertificate: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Success: Tag with key '" << tagKey <<
        "' and value '" << tagValue <<
        "' added to certificate with ARN '" <<
        certificateArn << "'." << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [AddTagsToCertificate](#) in *AWS SDK for C++ API Reference*.

DeleteCertificate

The following code example shows how to use DeleteCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Delete an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::ACM::deleteCertificate(const Aws::String &certificateArn,

```

```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::DeleteCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DeleteCertificateOutcome outcome =
        acmClient.DeleteCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: The certificate with the ARN '" <<
            certificateArn << "' is deleted." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteCertificate](#) in *AWS SDK for C++ API Reference*.

DescribeCertificate

The following code example shows how to use DescribeCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Describe an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.

```

```

*/
bool AwsDoc::ACM::describeCertificate(const Aws::String &certificateArn,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::DescribeCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DescribeCertificateOutcome outcome =
        acm_client.DescribeCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DescribeCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::ACM::Model::CertificateDetail certificate =
            outcome.GetResult().GetCertificate();

        std::cout << "Success: Information about certificate "
            "with ARN '" << certificateArn << "':" << std::endl <<
std::endl;

        std::cout << "ARN:                " << certificate.GetCertificateArn()
            << std::endl;
        std::cout << "Authority ARN:            " <<
            certificate.GetCertificateAuthorityArn() << std::endl;
        std::cout << "Created at (GMT):        " <<
            certificate.GetCreatedAt().ToGmtString(
                Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
        std::cout << "Domain name:             " << certificate.GetDomainName()
            << std::endl;

        Aws::Vector<Aws::ACM::Model::DomainValidation> options =
            certificate.GetDomainValidationOptions();

        if (!options.empty()) {
            std::cout << std::endl << "Domain validation information: "
                << std::endl << std::endl;

            for (auto &validation: options) {
                std::cout << "  Domain name:                " <<

```

```
        validation.GetDomainName() << std::endl;

    const Aws::ACM::Model::ResourceRecord &record =
        validation.GetResourceRecord();

    std::cout << " Resource record name:      " <<
        record.GetName() << std::endl;

    Aws::ACM::Model::RecordType recordType = record.GetType();
    Aws::String type;

    switch (recordType) {
        case Aws::ACM::Model::RecordType::CNAME:
            type = "CNAME";
            break;
        case Aws::ACM::Model::RecordType::NOT_SET:
            type = "Not set";
            break;
        default:
            type = "Cannot determine.";
            break;
    }

    std::cout << " Resource record type:      " << type <<
        std::endl;

    std::cout << " Resource record value:      " <<
        record.GetValue() << std::endl;

    std::cout << " Validation domain:          " <<
        validation.GetValidationDomain() << std::endl;

    Aws::Vector<Aws::String> emails =
        validation.GetValidationEmails();

    if (!emails.empty()) {
        std::cout << " Validation emails:" << std::endl <<
            std::endl;

        for (auto &email: emails) {
            std::cout << "      " << email << std::endl;
        }

        std::cout << std::endl;
    }
```

```
    }

    Aws::ACM::Model::ValidationMethod validationMethod =
        validation.GetValidationMethod();
    Aws::String method;

    switch (validationMethod) {
        case Aws::ACM::Model::ValidationMethod::DNS:
            method = "DNS";
            break;
        case Aws::ACM::Model::ValidationMethod::EMAIL:
            method = "Email";
            break;
        case Aws::ACM::Model::ValidationMethod::NOT_SET:
            method = "Not set";
            break;
        default:
            method = "Cannot determine";
    }

    std::cout << " Validation method:          " <<
        method << std::endl;

    Aws::ACM::Model::DomainStatus domainStatus =
        validation.GetValidationStatus();
    Aws::String status;

    switch (domainStatus) {
        case Aws::ACM::Model::DomainStatus::FAILED:
            status = "Failed";
            break;
        case Aws::ACM::Model::DomainStatus::NOT_SET:
            status = "Not set";
            break;
        case Aws::ACM::Model::DomainStatus::PENDING_VALIDATION:
            status = "Pending validation";
            break;
        case Aws::ACM::Model::DomainStatus::SUCCESS:
            status = "Success";
            break;
        default:
            status = "Cannot determine";
    }
}
```

```
        std::cout << " Domain validation status: " << status <<
            std::endl << std::endl;
    }
}

Aws::Vector<Aws::ACM::Model::ExtendedKeyUsage> usages =
    certificate.GetExtendedKeyUsages();

if (!usages.empty()) {
    std::cout << std::endl << "Extended key usages:" <<
        std::endl << std::endl;

    for (auto &usage: usages) {
        Aws::ACM::Model::ExtendedKeyUsageName usageName =
            usage.GetName();
        Aws::String name;

        switch (usageName) {
            case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
                name = "Any";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
                name = "Code signing";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
                name = "Custom";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
                name = "Email protection";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
                name = "IPSEC end system";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
                name = "IPSEC tunnel";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
                name = "IPSEC user";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
                name = "None";
                break;
            case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
```

```

        name = "Not set";
        break;
    case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
        name = "OCSP signing";
        break;
    case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
        name = "Time stamping";
        break;
    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
        name = "TLS web client authentication";
        break;
    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
        name = "TLS web server authentication";
        break;
    default:
        name = "Cannot determine";
    }

    std::cout << "  Name: " << name << std::endl;
    std::cout << "  OID: " << usage.GetOID() <<
        std::endl << std::endl;
}

std::cout << std::endl;
}

Aws::ACM::Model::CertificateStatus certificateStatus =
    certificate.GetStatus();
Aws::String status;

switch (certificateStatus) {
    case Aws::ACM::Model::CertificateStatus::EXPIRED:
        status = "Expired";
        break;
    case Aws::ACM::Model::CertificateStatus::FAILED:
        status = "Failed";
        break;
    case Aws::ACM::Model::CertificateStatus::INACTIVE:
        status = "Inactive";
        break;
    case Aws::ACM::Model::CertificateStatus::ISSUED:
        status = "Issued";

```

```
        break;
    case Aws::ACM::Model::CertificateStatus::NOT_SET:
        status = "Not set";
        break;
    case Aws::ACM::Model::CertificateStatus::PENDING_VALIDATION:
        status = "Pending validation";
        break;
    case Aws::ACM::Model::CertificateStatus::REVOKED:
        status = "Revoked";
        break;
    case Aws::ACM::Model::CertificateStatus::VALIDATION_TIMED_OUT:
        status = "Validation timed out";
        break;
    default:
        status = "Cannot determine";
}

std::cout << "Status:          " << status << std::endl;

if (certificate.GetStatus() ==
    Aws::ACM::Model::CertificateStatus::FAILED) {
    Aws::ACM::Model::FailureReason failureReason =
        certificate.GetFailureReason();
    Aws::String reason;

    switch (failureReason) {
        case
Aws::ACM::Model::FailureReason::ADDITIONAL_VERIFICATION_REQUIRED:
            reason = "Additional verification required";
            break;
        case Aws::ACM::Model::FailureReason::CAA_ERROR:
            reason = "CAA error";
            break;
        case Aws::ACM::Model::FailureReason::DOMAIN_NOT_ALLOWED:
            reason = "Domain not allowed";
            break;
        case Aws::ACM::Model::FailureReason::DOMAIN_VALIDATION_DENIED:
            reason = "Domain validation denied";
            break;
        case Aws::ACM::Model::FailureReason::INVALID_PUBLIC_DOMAIN:
            reason = "Invalid public domain";
            break;
        case Aws::ACM::Model::FailureReason::NOT_SET:
            reason = "Not set";
```



```
        break;
    case Aws::ACM::Model::FailureReason::NO_AVAILABLE_CONTACTS:
        reason = "No available contacts";
        break;
    case Aws::ACM::Model::FailureReason::OTHER:
        reason = "Other";
        break;
    case Aws::ACM::Model::FailureReason::PCA_ACCESS_DENIED:
        reason = "PCA access denied";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_ARGS:
        reason = "PCA invalid args";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_ARN:
        reason = "PCA invalid ARN";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_DURATION:
        reason = "PCA invalid duration";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_STATE:
        reason = "PCA invalid state";
        break;
    case Aws::ACM::Model::FailureReason::PCA_LIMIT_EXCEEDED:
        reason = "PCA limit exceeded";
        break;
    case
Aws::ACM::Model::FailureReason::PCA_NAME_CONSTRAINTS_VALIDATION:
        reason = "PCA name constraints validation";
        break;
    case Aws::ACM::Model::FailureReason::PCA_REQUEST_FAILED:
        reason = "PCA request failed";
        break;
    case Aws::ACM::Model::FailureReason::PCA_RESOURCE_NOT_FOUND:
        reason = "PCA resource not found";
        break;
    default:
        reason = "Cannot determine";
    }

    std::cout << "Failure reason:      " << reason << std::endl;
}

if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::REVOKED)
{
```

```
std::cout << "Revoked at (GMT):    " <<
    certificate.GetRevokedAt().ToGmtString(
        Aws::Utils::DateFormat::ISO_8601)
    << std::endl;

Aws::ACM::Model::RevocationReason revocationReason =
    certificate.GetRevocationReason();
Aws::String reason;

switch (revocationReason) {
    case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
        reason = "Affiliation changed";
        break;
    case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
        reason = "AA compromise";
        break;
    case Aws::ACM::Model::RevocationReason::CA_COMPROMISE:
        reason = "CA compromise";
        break;
    case Aws::ACM::Model::RevocationReason::CERTIFICATE_HOLD:
        reason = "Certificate hold";
        break;
    case Aws::ACM::Model::RevocationReason::CESSATION_OF_OPERATION:
        reason = "Cessation of operation";
        break;
    case Aws::ACM::Model::RevocationReason::KEY_COMPROMISE:
        reason = "Key compromise";
        break;
    case Aws::ACM::Model::RevocationReason::NOT_SET:
        reason = "Not set";
        break;
    case Aws::ACM::Model::RevocationReason::PRIVILEGE_WITHDRAWN:
        reason = "Privilege withdrawn";
        break;
    case Aws::ACM::Model::RevocationReason::REMOVE_FROM_CRL:
        reason = "Revoke from CRL";
        break;
    case Aws::ACM::Model::RevocationReason::SUPERCEDED:
        reason = "Superceded";
        break;
    case Aws::ACM::Model::RevocationReason::UNSPECIFIED:
        reason = "Unspecified";
        break;
    default:
```

```

        reason = "Cannot determine";
    }

    std::cout << "Revocation reason:  " << reason << std::endl;
}

if (certificate.GetType() == Aws::ACM::Model::CertificateType::IMPORTED) {
    std::cout << "Imported at (GMT):  " <<
        certificate.GetImportedAt().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
}

Aws::Vector<Aws::String> inUseBys = certificate.GetInUseBy();

if (!inUseBys.empty()) {
    std::cout << std::endl << "In use by:" << std::endl << std::endl;

    for (auto &in_use_by: inUseBys) {
        std::cout << "  " << in_use_by << std::endl;
    }

    std::cout << std::endl;
}

if (certificate.GetType() == Aws::ACM::Model::CertificateType::AMAZON_ISSUED
&&
    certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
    std::cout << "Issued at (GMT):  " <<
        certificate.GetIssuedAt().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
}

std::cout << "Issuer:          " << certificate.GetIssuer() <<
    std::endl;

Aws::ACM::Model::KeyAlgorithm keyAlgorithm =
    certificate.GetKeyAlgorithm();
Aws::String algorithm;

switch (keyAlgorithm) {
    case Aws::ACM::Model::KeyAlgorithm::EC_prime256v1:
        algorithm = "P-256 (secp256r1, prime256v1)";

```

```

        break;
    case Aws::ACM::Model::KeyAlgorithm::EC_secp384r1:
        algorithm = "P-384 (secp384r1)";
        break;
    case Aws::ACM::Model::KeyAlgorithm::EC_secp521r1:
        algorithm = "P-521 (secp521r1)";
        break;
    case Aws::ACM::Model::KeyAlgorithm::NOT_SET:
        algorithm = "Not set";
        break;
    case Aws::ACM::Model::KeyAlgorithm::RSA_1024:
        algorithm = "RSA 1024";
        break;
    case Aws::ACM::Model::KeyAlgorithm::RSA_2048:
        algorithm = "RSA 2048";
        break;
    case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
        algorithm = "RSA 4096";
        break;
    default:
        algorithm = "Cannot determine";
}

std::cout << "Key algorithm:      " << algorithm << std::endl;

if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
    std::cout << "Not valid after (GMT): " <<
        certificate.GetNotAfter().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
    std::cout << "Not valid before (GMT): " <<
        certificate.GetNotBefore().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
}

    Aws::ACM::Model::CertificateTransparencyLoggingPreference loggingPreference
=
certificate.GetOptions().GetCertificateTransparencyLoggingPreference();
    Aws::String preference;

    switch (loggingPreference) {

```

```
        case
    Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED:
        preference = "Disabled";
        break;
    case Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED:
        preference = "Enabled";
        break;
    case Aws::ACM::Model::CertificateTransparencyLoggingPreference::NOT_SET:
        preference = "Not set";
        break;
    default:
        preference = "Cannot determine";
}

std::cout << "Logging preference: " << preference << std::endl;

std::cout << "Serial:          " << certificate.GetSerial() <<
    std::endl;
std::cout << "Signature algorithm: "
    << certificate.GetSignatureAlgorithm() << std::endl;
std::cout << "Subject:          " << certificate.GetSubject() <<
    std::endl;

Aws::ACM::Model::CertificateType certificateType = certificate.GetType();
Aws::String type;

switch (certificateType) {
    case Aws::ACM::Model::CertificateType::AMAZON_ISSUED:
        type = "Amazon issued";
        break;
    case Aws::ACM::Model::CertificateType::IMPORTED:
        type = "Imported";
        break;
    case Aws::ACM::Model::CertificateType::NOT_SET:
        type = "Not set";
        break;
    case Aws::ACM::Model::CertificateType::PRIVATE_:
        type = "Private";
        break;
    default:
        type = "Cannot determine";
}

std::cout << "Type:          " << type << std::endl;
```

```
Aws::Vector<Aws::String> altNames =
    certificate.GetSubjectAlternativeNames();

if (!altNames.empty()) {
    std::cout << std::endl << "Alternative names:" <<
        std::endl << std::endl;

    for (auto &alt_name: altNames) {
        std::cout << " " << alt_name << std::endl;
    }

    std::cout << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DescribeCertificate](#) in *AWS SDK for C++ API Reference*.

ExportCertificate

The following code example shows how to use `ExportCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Export an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param passphrase: A passphrase to decrypt the exported certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
```

```

bool AwsDoc::ACM::exportCertificate(const Aws::String &certificateArn,
                                   const Aws::String &passphrase,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ExportCertificateRequest request;
    Aws::Utils::CryptoBuffer cryptoBuffer(
        reinterpret_cast<const unsigned char *>(passphrase.c_str()),
        passphrase.length());
    request.WithCertificateArn(certificateArn).WithPassphrase(cryptoBuffer);

    Aws::ACM::Model::ExportCertificateOutcome outcome =
        acm_client.ExportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ExportCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
            << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate:      " << std::endl << std::endl <<
            result.GetCertificate() << std::endl << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
            result.GetCertificateChain() << std::endl << std::endl;
        std::cout << "Private key:      " << std::endl << std::endl <<
            result.GetPrivateKey() << std::endl;
    }

    return outcome.IsSuccess();
}


```

- For API details, see [ExportCertificate](#) in *AWS SDK for C++ API Reference*.

GetCertificate

The following code example shows how to use `GetCertificate`.

SDK for C++

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Get an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::getCertificate(const Aws::String &certificateArn,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::GetCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::GetCertificateOutcome outcome =
        acmClient.GetCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: GetCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
            << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate: " << std::endl << std::endl <<
            result.GetCertificate() << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
            result.GetCertificateChain() << std::endl;
    }

    return outcome.IsSuccess();
}
```



```
}

```

- For API details, see [GetCertificate](#) in *AWS SDK for C++ API Reference*.

ImportCertificate

The following code example shows how to use `ImportCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Import an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateFile: Path to certificate to import.
 \param privateKeyFile: Path to file containing a private key.
 \param certificateChainFile: Path to file containing a PEM encoded certificate
 chain.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                   const Aws::String &privateKeyFile,
                                   const Aws::String &certificateChainFile,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
            "' does not exist." << std::endl;

        return false;
    }

    std::ifstream privateKeyInStream(privateKeyFile.c_str());
    if (!privateKeyInStream) {
        std::cerr << "Error: The private key file '" << privateKeyFile <<

```

```
        "" does not exist." << std::endl;

    return false;
}

std::ifstream certificateChainInStream(certificateChainFile.c_str());
if (!certificateChainInStream) {
    std::cerr << "Error: The certificate chain file '"
        << certificateChainFile << "' does not exist." << std::endl;

    return false;
}

Aws::String certificate;
certificate.assign(std::istreambuf_iterator<char>(certificateInStream),
    std::istreambuf_iterator<char>());

Aws::String privateKey;
privateKey.assign(std::istreambuf_iterator<char>(privateKeyInStream),
    std::istreambuf_iterator<char>());

Aws::String certificateChain;

certificateChain.assign(std::istreambuf_iterator<char>(certificateChainInStream),
    std::istreambuf_iterator<char>());

Aws::ACM::ACMClient acmClient(clientConfiguration);

Aws::ACM::Model::ImportCertificateRequest request;

request.WithCertificate(Aws::Utils::ByteBuffer((unsigned char *)
    certificate.c_str(),
    certificate.size()))
    .WithPrivateKey(Aws::Utils::ByteBuffer((unsigned char *)
    privateKey.c_str(),
    privateKey.size()))
    .WithCertificateChain(Aws::Utils::ByteBuffer((unsigned char *)
certificateChain.c_str(),
    certificateChain.size()));

Aws::ACM::Model::ImportCertificateOutcome outcome =
    acmClient.ImportCertificate(request);
```

```
if (!outcome.IsSuccess()) {
    std::cerr << "Error: ImportCertificate: " <<
        outcome.GetError().GetMessage() << std::endl;

    return false;
}
else {
    std::cout << "Success: Certificate associated with ARN '" <<
        outcome.GetResult().GetCertificateArn() << "' imported."
        << std::endl;

    return true;
}
}
```

- For API details, see [ImportCertificate](#) in *AWS SDK for C++ API Reference*.

ListCertificates

The following code example shows how to use ListCertificates.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! List the AWS Certificate Manager (ACM) certificates in an account.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ListCertificatesRequest request;
    Aws::Vector<Aws::ACM::Model::CertificateSummary> allCertificates;
    Aws::String nextToken;
```

```
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    Aws::ACM::Model::ListCertificatesOutcome outcome =
        acmClient.ListCertificates(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListCertificates: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        const Aws::ACM::Model::ListCertificatesResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates =
            result.GetCertificateSummaryList();
        allCertificates.insert(allCertificates.end(), certificates.begin(),
            certificates.end());

        nextToken = result.GetNextToken();
    }
} while (!nextToken.empty());

if (!allCertificates.empty()) {
    for (const Aws::ACM::Model::CertificateSummary &certificate:
allCertificates) {
        std::cout << "Certificate ARN: " <<
            certificate.GetCertificateArn() << std::endl;
        std::cout << "Domain name:      " <<
            certificate.GetDomainName() << std::endl << std::endl;
    }
}
else {
    std::cout << "No available certificates found in account."
        << std::endl;
}

return true;
}
```

- For API details, see [ListCertificates](#) in *AWS SDK for C++ API Reference*.

ListTagsForCertificate

The following code example shows how to use `ListTagsForCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ List the tags for an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listTagsForCertificate(const Aws::String &certificateArn,
                                        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ListTagsForCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::ListTagsForCertificateOutcome outcome =
        acm_client.ListTagsForCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cout << "Error: ListTagsForCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Information about tags for "
```

```

        "certificate with ARN '"
        << certificateArn << "':" << std::endl << std::endl;

    auto result = outcome.GetResult();

    Aws::Vector<Aws::ACM::Model::Tag> tags =
        result.GetTags();

    if (tags.size() > 0) {
        for (const Aws::ACM::Model::Tag &tag: tags) {
            std::cout << "Key:  " << tag.GetKey() << std::endl;
            std::cout << "Value: " << tag.GetValue()
                << std::endl << std::endl;
        }
    }
    else {
        std::cout << "No tags found." << std::endl;
    }

    return true;
}
}

```

- For API details, see [ListTagsForCertificate](#) in *AWS SDK for C++ API Reference*.

RemoveTagsFromCertificate

The following code example shows how to use `RemoveTagsFromCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Remove a tag from an ACM certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.

```

```

\param tagKey: The key for the tag.
\param tagValue: The value for the tag.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::ACM::removeTagsFromCertificate(const Aws::String &certificateArn,
                                           const Aws::String &tagKey,
                                           const Aws::String &tagValue,
                                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::Vector<Aws::ACM::Model::Tag> tags;

    Aws::ACM::Model::Tag tag;
    tag.SetKey(tagKey);

    tags.push_back(tag);

    Aws::ACM::Model::RemoveTagsFromCertificateRequest request;
    request.WithCertificateArn(certificateArn)
           .WithTags(tags);

    Aws::ACM::Model::RemoveTagsFromCertificateOutcome outcome =
        acmClient.RemoveTagsFromCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RemoveTagFromCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey << "' removed from "
            << "certificate with ARN '" << certificateArn << "'." <<
std::endl;

        return true;
    }
}

```

- For API details, see [RemoveTagsFromCertificate](#) in *AWS SDK for C++ API Reference*.

RenewCertificate

The following code example shows how to use `RenewCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Renew an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::renewCertificate(const Aws::String &certificateArn,
                                  const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RenewCertificateRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::RenewCertificateOutcome outcome =
        acmClient.RenewCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RenewCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Renewed certificate with ARN '"
            << certificateArn << "'." << std::endl;

        return true;
    }
}
```


- For API details, see [RenewCertificate](#) in *AWS SDK for C++ API Reference*.

RequestCertificate

The following code example shows how to use RequestCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Request an AWS Certificate Manager (ACM) certificate.
/*!
 \param domainName: A fully qualified domain name.
 \param idempotencyToken: Customer chosen string for idempotency.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::requestCertificate(const Aws::String &domainName,
                                    const Aws::String &idempotencyToken,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RequestCertificateRequest request;
    request.WithDomainName(domainName)
           .WithIdempotencyToken(idempotencyToken);

    Aws::ACM::Model::RequestCertificateOutcome outcome =
        acmClient.RequestCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "RequestCertificate error: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
}
```

```

    else {
        std::cout << "Success: The newly requested certificate's "
                  "ARN is '" <<
                  outcome.GetResult().GetCertificateArn() <<
                  "'." << std::endl;

        return true;
    }
}

```

- For API details, see [RequestCertificate](#) in *AWS SDK for C++ API Reference*.

ResendValidationEmail

The following code example shows how to use `ResendValidationEmail`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Resend the email that requests domain ownership validation.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param domainName: A fully qualified domain name.
 \param validationDomain: The base validation domain that will act as the suffix
                        of the email addresses.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::resendValidationEmail(const Aws::String &certificateArn,
                                       const Aws::String &domainName,
                                       const Aws::String &validationDomain,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ResendValidationEmailRequest request;

```

```

request.WithCertificateArn(certificateArn)
        .WithDomain(domainName)
        .WithValidationDomain(validationDomain);

Aws::ACM::Model::ResendValidationEmailOutcome outcome =
    acmClient.ResendValidationEmail(request);

if (!outcome.IsSuccess()) {
    std::cerr << "ResendValidationEmail error: " <<
        outcome.GetError().GetMessage() << std::endl;

    return false;
}
else {
    std::cout << "Success: The validation email has been resent."
        << std::endl;

    return true;
}
}

```

- For API details, see [ResendValidationEmail](#) in *AWS SDK for C++ API Reference*.

UpdateCertificateOptions

The following code example shows how to use UpdateCertificateOptions.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Update an AWS Certificate Manager (ACM) certificate option.
/!*
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param loggingEnabled: Boolean specifying logging enabled.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.

```

```

*/
bool AwsDoc::ACM::updateCertificateOption(const Aws::String &certificateArn,
                                          bool loggingEnabled,
                                          const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::UpdateCertificateOptionsRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::CertificateOptions options;

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(
            Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED);
    }
    else {
        options.SetCertificateTransparencyLoggingPreference(
            Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED);
    }

    request.SetOptions(options);

    Aws::ACM::Model::UpdateCertificateOptionsOutcome outcome =
        acmClient.UpdateCertificateOptions(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "UpdateCertificateOption error: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The option '"
            << (loggingEnabled ? "enabled" : "disabled") << "' has been set
for "
                                                    "the certificate
with the ARN '"
            << certificateArn << "'."
            << std::endl;

        return true;
    }
}

```

```
}
```

- For API details, see [UpdateCertificateOptions](#) in *AWS SDK for C++ API Reference*.

Aurora examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Aurora.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Aurora

The following code examples show how to get started using Aurora.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)
```

```
# Set this project's name.
project("hello_aurora")

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_aurora.cpp)

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

Code for the `hello_aurora.cpp` source file.

```
#include <aws/core/Aws.h>
```

```
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 * A "Hello Aurora" starter application which initializes an Amazon Relational
 * Database Service (Amazon RDS) client
 * and describes the Amazon Aurora (Aurora) clusters.
 *
 * main function
 *
 * Usage: 'hello_aurora'
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                    << outcome.GetError().GetMessage()

```

```
        << std::endl;
        break;
    }
} while (!marker.empty());

std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
for (auto &clusterId: clusterIds) {
    std::cout << "  clusterId " << clusterId << std::endl;
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateDBCluster

The following code example shows how to use CreateDBCluster.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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



```
Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBClusterRequest request;
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetEngine(engineName);
request.SetEngineVersion(engineVersionName);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBClusterOutcome outcome =
    client.CreateDBCluster(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster creation has started."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::CreateDBCluster. "
              << outcome.GetError().GetMessage()
              << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for C++ API Reference*.

CreateDBClusterParameterGroup

The following code example shows how to use CreateDBClusterParameterGroup.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetDBParameterGroupFamily(dbParameterGroupFamily);
request.SetDescription("Example cluster parameter group.");

Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
    client.CreateDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster parameter group was successfully created."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for C++ API Reference*.

CreateDBClusterSnapshot

The following code example shows how to use `CreateDBClusterSnapshot`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

```
Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
        client.CreateDBClusterSnapshot(request);

    if (outcome.IsSuccess()) {
        std::cout << "Snapshot creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for C++ API Reference*.

CreateDBInstance

The following code example shows how to use CreateDBInstance.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

```
Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetEngine(engineName);
    request.SetDBInstanceClass(dbInstanceClass);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
        client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
            "",
                        client);
        return false;
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for C++ API Reference*.

DeleteDBCluster

The following code example shows how to use DeleteDBCluster.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

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

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBClusterRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);
    request.SetSkipFinalSnapshot(true);

    Aws::RDS::Model::DeleteDBClusterOutcome outcome =
        client.DeleteDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB cluster deletion has started."
            << std::endl;
        clusterDeleting = true;
        std::cout
            << "Waiting for DB cluster to delete before deleting the
parameter group."
            << std::endl;
        std::cout << "This may take a while." << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBCluster. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for C++ API Reference*.

DeleteDBClusterParameterGroup

The following code example shows how to use `DeleteDBClusterParameterGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(parameterGroupName);

Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
    client.DeleteDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
    result = false;
}
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for C++ API Reference*.

DeleteDBInstance

The following code example shows how to use DeleteDBInstance.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

```
Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBInstanceRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);
    request.SetSkipFinalSnapshot(true);
    request.SetDeleteAutomatedBackups(true);

    Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
        client.DeleteDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB instance deletion has started."
                  << std::endl;
        instanceDeleting = true;
        std::cout
            << "Waiting for DB instance to delete before deleting the
parameter group."
            << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for C++ API Reference*.

DescribeDBClusterParameterGroups

The following code example shows how to use `DescribeDBClusterParameterGroups`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
    client.DescribeDBClusterParameterGroups(request);

if (outcome.IsSuccess()) {
    std::cout << "DB cluster parameter group named '" <<
        CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
    dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
}

else {
    std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for C++ API Reference*.

DescribeDBClusterParameters

The following code example shows how to use DescribeDBClusterParameters.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```

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

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
    /*!
    \sa getDBClusterParameters()
    \param parameterGroupName: The name of the cluster parameter group.
    \param namePrefix: Prefix string to filter results by parameter name.
    \param source: A source such as 'user', ignored if empty.
    \param parametersResult: Vector of 'Parameter' objects returned by the routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,
                                                const Aws::String &namePrefix,
                                                const Aws::String &source,
                                                Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                                const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
            client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {

```

```
        parametersResult.push_back(parameter);
    }
}
else {
    parametersResult.push_back(parameter);
}
}

marker = outcome.GetResult().GetMarker();
}
else {
    std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
} while (!marker.empty());

return true;
}
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for C++ API Reference*.

DescribeDBClusterSnapshots

The following code example shows how to use `DescribeDBClusterSnapshots`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);
```

```

    Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
        client.DescribeDBClusterSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for C++ API Reference*.

DescribeDBClusters

The following code example shows how to use DescribeDBClusters.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets a DB cluster description.

```

```

/!*
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
        client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
    // This example does not log an error if the DB cluster does not exist.
    // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;
}

```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for C++ API Reference*.

DescribeDBEngineVersions

The following code example shows how to use DescribeDBEngineVersions.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

                                         Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
    } while (true);
}
```

```
    }

    Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
        client.DescribeDBEngineVersions(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
            outcome.GetResult().GetDBEngineVersions();

        engineVersionsResult.insert(engineVersionsResult.end(),
            engineVersions.begin(),
engineVersions.end());
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
} while (!marker.empty());

return true;
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for C++ API Reference*.

DescribeDBInstances

The following code example shows how to use DescribeDBInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
```

```

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

        Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets a DB instance description.
    /*!
    \sa describeDBCluster()
    \param dbInstanceIdentifier: A DB instance identifier.
    \param instanceResult: The 'DBInstance' object containing the description.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                           Aws::RDS::Model::DBInstance &instanceResult,
                                           const Aws::RDS::RDSClient &client) {
        Aws::RDS::Model::DescribeDBInstancesRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);

        Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

        bool result = true;
        if (outcome.IsSuccess()) {
            instanceResult = outcome.GetResult().GetDBInstances()[0];
        }
        else if (outcome.GetError().GetErrorType() !=
                Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
            result = false;
            std::cerr << "Error with Aurora::DescribeDBInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
        else {
            instanceResult = Aws::RDS::Model::DBInstance();
        }

        return result;
    }
}

```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for C++ API Reference*.

DescribeOrderableDBInstanceOptions

The following code example shows how to use DescribeOrderableDBInstanceOptions.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
    }
```



```

    Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
        client.DescribeOrderableDBInstanceOptions(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
            outcome.GetResult().GetOrderableDBInstanceOptions();
        for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
        {
            const Aws::String &instanceClass = option.GetDBInstanceClass();
            if (std::find(instanceClasses.begin(), instanceClasses.end(),
                instanceClass) == instanceClasses.end()) {
                instanceClasses.push_back(instanceClass);
            }
        }
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

std::cout << "The available DB instance classes for your database engine are:"
    << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for C++ API Reference*.

ModifyDBClusterParameterGroup

The following code example shows how to use `ModifyDBClusterParameterGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetParameters(updateParameters);

Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
    client.ModifyDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster parameter group was successfully modified."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for C++ API Reference*.

Scenarios

Get started with DB clusters

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.

- Create a DB cluster that uses the parameter group.
- Create a DB instance that contains a database.
- Take a snapshot of the DB cluster, then clean up resources.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    //! Routine which creates an Amazon Aurora DB cluster and demonstrates several
    operations
    //! on that cluster.
    /*!
    \sa gettingStartedWithDBClusters()
    \param clientConfiguration: AWS client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
        Aws::RDS::RDSClient client(clientConfig);

        printAsterisksLine();
        std::cout << "Welcome to the Amazon Relational Database Service (Amazon Aurora)"
            << std::endl;
        std::cout << "get started with DB clusters demo." << std::endl;
        printAsterisksLine();

        std::cout << "Checking for an existing DB cluster parameter group named '" <<
            CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
        Aws::String dbParameterGroupFamily("Undefined");
        bool parameterGroupFound = true;
        {
            // 1. Check if the DB cluster parameter group already exists.
            Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;

```

```

request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
    client.DescribeDBClusterParameterGroups(request);

if (outcome.IsSuccess()) {
    std::cout << "DB cluster parameter group named '" <<
        CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
    dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
}
else if (outcome.GetError().GetErrorType() ==
    Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
    std::cout << "DB cluster parameter group named '" <<
        CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
    parameterGroupFound = false;
}
else {
    std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
}

if (!parameterGroupFound) {
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 2. Get available parameter group families for the specified engine.
    if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
        engineVersions, client)) {
        return false;
    }

    std::cout << "Getting available parameter group families for " << DB_ENGINE
        << "."
        << std::endl;
    std::vector<Aws::String> families;
    for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
        Aws::String family = version.GetDBParameterGroupFamily();
        if (std::find(families.begin(), families.end(), family) ==
            families.end()) {

```

```

        families.push_back(family);
        std::cout << " " << families.size() << ": " << family << std::endl;
    }
}

int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                    static_cast<int>(families.size()));
dbParameterGroupFamily = families[choice - 1];
}
if (!parameterGroupFound) {
    // 3. Create a DB cluster parameter group.
    Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example cluster parameter group.");

    Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
        client.CreateDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "Let's set some parameter values in your cluster parameter group."
          << std::endl;

Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
// 4. Get the parameters in the DB cluster parameter group.
if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX,
                             NO_SOURCE,
                             autoIncrementParameters,
                             client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}
}

```

```

Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
    if (autoIncParameter.GetIsModifiable() &&
        (autoIncParameter.GetDataTypes() == "integer")) {
        std::cout << "The " << autoIncParameter.GetParameterName()
            << " is described as: " <<
            autoIncParameter.GetDescription() << "." << std::endl;
        if (autoIncParameter.ParameterValueHasBeenSet()) {
            std::cout << "The current value is "
                << autoIncParameter.GetParameterValue()
                << "." << std::endl;
        }
        std::vector<int> splitValues = splitToInts(
            autoIncParameter.GetAllowedValues(), '-');
        if (splitValues.size() == 2) {
            int newValue = askQuestionForIntRange(
                Aws::String("Enter a new value between ") +
                autoIncParameter.GetAllowedValues() + ": ",
                splitValues[0], splitValues[1]);
            autoIncParameter.SetParameterValue(std::to_string(newValue));
            updateParameters.push_back(autoIncParameter);
        }
        else {
            std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                << std::endl;
        }
    }
}

{
    // 5. Modify the auto increment parameters in the DB cluster parameter
group.
    Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
        client.ModifyDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully modified."

```

```

        << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }
}

std::cout
of 'user'."
    << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6. Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
    "user",
        userParameters,
        client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << " " << userParameter.GetParameterName() << ", " <<
        userParameter.GetDescription() << ", parameter value - "
        << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7. Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
    }

```

```

    engineVersionName = dbCluster.GetEngineVersion();
    engineName = dbCluster.GetEngine();

}
else {
    std::cout << "Let's create a DB cluster." << std::endl;
    const Aws::String administratorName = askQuestion(
        "Enter an administrator username for the database: ");
    const Aws::String administratorPassword = askQuestion(
        "Enter a password for the administrator (at least 8 characters): ");
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 8. Get a list of engine versions for the parameter group family.
    if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
        client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    std::cout << "The available engines for your parameter group family are:"
        << std::endl;

    int index = 1;
    for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
    {
        std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
            << std::endl;
        ++index;
    }
    int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
    const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

    engineName = engineVersion.GetEngine();
    engineVersionName = engineVersion.GetEngineVersion();
    std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
        << "' and database '" << DB_NAME << "'.\n"
        << "The DB cluster is configured to use your custom cluster
parameter group '"
        << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
        << "selected engine version " << engineVersion.GetEngineVersion()

```



```
        << ".\nThis typically takes several minutes." << std::endl;

    Aws::RDS::Model::CreateDBClusterRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetEngine(engineName);
    request.SetEngineVersion(engineVersionName);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBClusterOutcome outcome =
        client.CreateDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster creation has started."
            << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBCluster. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }
}

std::cout << "Waiting for the DB cluster to become available." << std::endl;

int counter = 0;
// 11. Wait for the DB cluster to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for cluster to become available timed out after "
            << counter
            << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
            DB_CLUSTER_IDENTIFIER, "", client);
        return false;
    }

    dbCluster = Aws::RDS::Model::DBCluster();
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
```

```

        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, "", client);
    return false;
}

if ((counter % 20) == 0) {
    std::cout << "Current DB cluster status is '"
                << dbCluster.GetStatus()
                << "' after " << counter << " seconds." << std::endl;
}
} while (dbCluster.GetStatus() != "available");

if (dbCluster.GetStatus() == "available") {
    std::cout << "The DB cluster has been created." << std::endl;
}

printAsterisksLine();
Aws::RDS::Model::DBInstance dbInstance;
// 11. Check if the DB instance already exists.
if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                    client);
    return false;
}

if (dbInstance.DbInstancePortHasBeenSet()) {
    std::cout << "The DB instance already exists." << std::endl;
}
else {
    std::cout << "Let's create a DB instance." << std::endl;

    Aws::String dbInstanceClass;
    // 12. Get a list of instance classes.
    if (!chooseDBInstanceClass(engineName,
                               engineVersionName,
                               dbInstanceClass,
                               client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
            "",
                               client);
        return false;
    }

    std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER

```

```

        << "" with selected DB instance class "" << dbInstanceClass
        << "".\nThis typically takes several minutes." << std::endl;

// 13. Create a DB instance.
Aws::RDS::Model::CreateDBInstanceRequest request;
request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetEngine(engineName);
request.SetDBInstanceClass(dbInstanceClass);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
    << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
    << outcome.GetError().GetMessage()
    << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                    client);
    return false;
}
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

counter = 0;
// 14. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
        << counter
        << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }
}

```

```

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
                  << dbInstance.GetDBInstanceStatus()
                  << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}

// 15. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbCluster);

printAsterisksLine();

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
    Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                          Aws::String(Aws::Utils::UUID::RandomUUID()));
    {
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
        Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
            client.CreateDBClusterSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                    << std::endl;
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}

std::cout << "Waiting for the snapshot to become available." << std::endl;

Aws::RDS::Model::DBClusterSnapshot snapshot;
counter = 0;
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 600) {
        std::cerr << "Wait for snapshot to be available timed out after "
                  << counter
                  << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    // 17. Wait for the snapshot to become available.
    Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
        client.DescribeDBClusterSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,

```

```

        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current snapshot status is '"
            << snapshot.GetStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (snapshot.GetStatus() != "available");

if (snapshot.GetStatus() != "available") {
    std::cout << "A snapshot has been created." << std::endl;
}
}

printAsterisksLine();

bool result = true;
if (askYesNoQuestion(
    "Do you want to delete the DB cluster, DB instance, and parameter group
(y/n)? ")) {
    result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
        client);
}

return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
    Aws::RDS::Model::DBCluster &clusterResult,
    const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

```

```

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
        client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
        Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
    // This example does not log an error if the DB cluster does not exist.
    // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;
}

//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBClusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,
                                             Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {

```

```

    Aws::RDS::Model::DescribeDBClusterParametersRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    if (!marker.empty()) {
        request.SetMarker(marker);
    }
    if (!source.empty()) {
        request.SetSource(source);
    }

    Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
        client.DescribeDBClusterParameters(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
            outcome.GetResult().GetParameters();
        for (const Aws::RDS::Model::Parameter &parameter: parameters) {
            if (!namePrefix.empty()) {
                if (parameter.GetParameterName().find(namePrefix) == 0) {
                    parametersResult.push_back(parameter);
                }
            }
            else {
                parametersResult.push_back(parameter);
            }
        }

        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()

```



```

\param engineName: A DB engine name.
\param parameterGroupFamily: A parameter group family name, ignored if empty.
\param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
routine.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

                                         Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
            client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                       engineVersions.begin(),
                                       engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());
}

```

```

    return true;
}

//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                         Aws::RDS::Model::DBInstance &instanceResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
        client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
    // This example does not log an error if the DB instance does not exist.
    // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}

//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.

```

```

/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
            {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                             instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

```

} while (!marker.empty());

std::cout << "The available DB instance classes for your database engine are:"
          << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                     const Aws::String &dbClusterIdentifier,
                                     const Aws::String &dbInstanceIdentifier,
                                     const Aws::RDS::RDSClient &client) {

    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
        }
    }
}

```

```

        instanceDeleting = true;
        std::cout
            << "Waiting for DB instance to delete before deleting the
parameter group."
            << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBInstance. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}
}

if (!dbClusterIdentifier.empty()) {
    {
        // 19. Delete the DB cluster.
        Aws::RDS::Model::DeleteDBClusterRequest request;
        request.SetDBClusterIdentifier(dbClusterIdentifier);
        request.SetSkipFinalSnapshot(true);

        Aws::RDS::Model::DeleteDBClusterOutcome outcome =
            client.DeleteDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster deletion has started."
                << std::endl;
            clusterDeleting = true;
            std::cout
                << "Waiting for DB cluster to delete before deleting the
parameter group."
                << std::endl;
            std::cout << "This may take a while." << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBCluster. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}
}
int counter = 0;

```

```
while (clusterDeleting || instanceDeleting) {
    // 20. Wait for the DB cluster and instance to be deleted.
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 800) {
        std::cerr << "Wait for instance to delete timed out after " << counter
            << " seconds." << std::endl;
        return false;
    }

    Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
    if (instanceDeleting) {
        if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
            return false;
        }
        instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
    }

    Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
    if (clusterDeleting) {
        if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
            return false;
        }

        clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
    }

    if ((counter % 20) == 0) {
        if (instanceDeleting) {
            std::cout << "Current DB instance status is '"
                << dbInstance.GetDBInstanceStatus() << "'." << std::endl;
        }

        if (clusterDeleting) {
            std::cout << "Current DB cluster status is '"
                << dbCluster.GetStatus() << "'." << std::endl;
        }
    }
}

if (!parameterGroupName.empty()) {
    // 21. Delete the DB cluster parameter group.
    Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
```

```
request.SetDBClusterParameterGroupName(parameterGroupName);

Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
    client.DeleteDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
    result = false;
}
}

return result;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.

- [CreateDBCluster](#)
- [CreateDBClusterParameterGroup](#)
- [CreateDBClusterSnapshot](#)
- [CreateDBInstance](#)
- [DeleteDBCluster](#)
- [DeleteDBClusterParameterGroup](#)
- [DeleteDBInstance](#)
- [DescribeDBClusterParameterGroups](#)
- [DescribeDBClusterParameters](#)
- [DescribeDBClusterSnapshots](#)
- [DescribeDBClusters](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)

Auto Scaling examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Auto Scaling.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Auto Scaling

The following code examples show how to get started using Auto Scaling.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

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

# Set the C++ standard to use to build this target.
```



```

# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_autoscaling.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

```

```
/*
 * A "Hello Autoscaling" starter application which initializes an Amazon EC2 Auto
 * Scaling client and describes the
 * Amazon EC2 Auto Scaling groups.
 *
 * main function
 *
 * Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

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

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
&autoScalingGroups =
                    outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
            }
        }
    }
}
```

```
        nextToken = outcome.GetResult().GetNextToken();
    } else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = 1;
        break;
    }
} while (!nextToken.empty());

std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
for (auto &groupName: groupNames) {
    std::cout << "AutoScaling group: " << groupName << std::endl;
}

}

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

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateAutoScalingGroup

The following code example shows how to use `CreateAutoScalingGroup`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
Aws::Vector<Aws::String> availabilityGroupZones;
availabilityGroupZones.push_back(
    availabilityZones[availabilityZoneChoice - 1].GetZoneName());
request.SetAvailabilityZones(availabilityGroupZones);
request.SetMaxSize(1);
request.SetMinSize(1);

Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
launchTemplateSpecification.SetLaunchTemplateName(templateName);
request.SetLaunchTemplate(launchTemplateSpecification);

Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
    autoScalingClient.CreateAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Created Auto Scaling group '" << groupName << "'..."
        << std::endl;
}
else if (outcome.GetError().GetErrorType() ==
    Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
    std::cout << "Auto Scaling group '" << groupName << "' already exists."
        << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
```

```
        << outcome.GetError().GetMessage()  
        << std::endl;  
  
    }
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for C++ API Reference*.

DeleteAutoScalingGroup

The following code example shows how to use DeleteAutoScalingGroup.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
// Optional: Set to the AWS Region (overrides config file).  
// clientConfig.region = "us-east-1";  
  
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);  
  
    Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;  
    request.SetAutoScalingGroupName(groupName);  
  
    Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =  
        autoScalingClient.DeleteAutoScalingGroup(request);  
  
    if (outcome.IsSuccess()) {  
        std::cout << "Auto Scaling group '" << groupName << "' was deleted."  
        << std::endl;  
    }  
    else {  
        std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "  
        << outcome.GetError().GetMessage()  
        << std::endl;  
        result = false;  
    }
```

```
    }  
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for C++ API Reference*.

DescribeAutoScalingGroups

The following code example shows how to use `DescribeAutoScalingGroups`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
// Optional: Set to the AWS Region (overrides config file).  
// clientConfig.region = "us-east-1";  
  
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);  
  
Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;  
Aws::Vector<Aws::String> groupNames;  
groupNames.push_back(groupName);  
request.SetAutoScalingGroupNames(groupNames);  
  
Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =  
    client.DescribeAutoScalingGroups(request);  
  
if (outcome.IsSuccess()) {  
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();  
}  
else {  
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "  
                << outcome.GetError().GetMessage()  
                << std::endl;  
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for C++ API Reference*.

DescribeAutoScalingInstances

The following code example shows how to use `DescribeAutoScalingInstances`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for C++ API Reference*.

DescribeScalingActivities

The following code example shows how to use `DescribeScalingActivities`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
Aws::String nextToken; // Used for pagination;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }
    Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
        autoScalingClient.DescribeScalingActivities(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
            outcome.GetResult().GetActivities();
        allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}
```



```
    } while (!nextToken.empty());

    std::cout << "Found " << allActivities.size() << " activities."
              << std::endl;
    std::cout << "Activities are ordered with the most recent first."
              << std::endl;
    for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
        std::cout << activity.GetDescription() << std::endl;
        std::cout << activity.GetDetails() << std::endl;
    }
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for C++ API Reference*.

DisableMetricsCollection

The following code example shows how to use `DisableMetricsCollection`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
    autoScalingClient.DisableMetricsCollection(request);

if (outcome.IsSuccess()) {
    std::cout << "Metrics collection has been disabled." << std::endl;
}
else {
```

```
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "  
                << outcome.GetError().GetMessage()  
                << std::endl;  
    }
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for C++ API Reference*.

EnableMetricsCollection

The following code example shows how to use `EnableMetricsCollection`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
// Optional: Set to the AWS Region (overrides config file).  
// clientConfig.region = "us-east-1";  
  
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);  
  
Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;  
request.SetAutoScalingGroupName(groupName);  
  
request.AddMetrics("GroupMinSize");  
request.AddMetrics("GroupMaxSize");  
request.AddMetrics("GroupDesiredCapacity");  
request.AddMetrics("GroupInServiceInstances");  
request.AddMetrics("GroupTotalInstances");  
request.SetGranularity("1Minute");  
  
Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =  
    autoScalingClient.EnableMetricsCollection(request);  
if (outcome.IsSuccess()) {  
    std::cout << "Auto Scaling metrics have been enabled."  
}
```

```
        << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
}
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for C++ API Reference*.

SetDesiredCapacity

The following code example shows how to use SetDesiredCapacity.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetDesiredCapacity(2);

Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
    autoScalingClient.SetDesiredCapacity(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

```
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for C++ API Reference*.

TerminateInstanceInAutoScalingGroup

The following code example shows how to use `TerminateInstanceInAutoScalingGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
request.SetInstanceId(instanceIDs[instanceNumber - 1]);
request.SetShouldDecrementDesiredCapacity(false);

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
    autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Waiting for EC2 instance with ID '"
                << instanceIDs[instanceNumber - 1] << "' to terminate..."
                << std::endl;
}
else {
    std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage()
```

```
        << std::endl;
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for C++ API Reference*.

UpdateAutoScalingGroup

The following code example shows how to use `UpdateAutoScalingGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetMaxSize(3);

Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
    autoScalingClient.UpdateAutoScalingGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for C++ API Reference*.

Scenarios

Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group with a launch template and Availability Zones, and get information about running instances.
- Enable Amazon CloudWatch metrics collection.
- Update the group's desired capacity and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics, then clean up resources.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Routine which demonstrates using an Auto Scaling group
#!/ to manage Amazon EC2 instances.
/*!
 \sa groupsAndInstancesScenario()
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout
        << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
```

```
        << "demo for managing groups and instances." << std::endl;
std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
        << std::endl;

std::cout << "This example requires an EC2 launch template." << std::endl;
if (askYesNoQuestion(
    "Would you like to use an existing EC2 launch template (y/n)? ") {

    // 1. Specify the name of an existing EC2 launch template.
    templateName = askQuestion(
        "Enter the name of the existing EC2 launch template. ");

    Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
    request.AddLaunchTemplateName(templateName);
    Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
        ec2Client.DescribeLaunchTemplates(request);

    if (outcome.IsSuccess()) {
        std::cout << "Validated the EC2 launch template '" << templateName
            << "' exists by calling DescribeLaunchTemplate." << std::endl;
    }
    else {
        std::cerr << "Error validating the existence of the launch template. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}
else { // 2. Or create a new EC2 launch template.
    templateName = askQuestion("Enter the name for a new EC2 launch template:
");

    Aws::EC2::Model::CreateLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

    requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
    requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

    request.SetLaunchTemplateData(requestLaunchTemplateData);

    Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
        ec2Client.CreateLaunchTemplate(request);
```

```

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << "' was
created."
                << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
            "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already exists"
                << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
        "Enter a name for the Auto Scaling group: ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
            ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                << "EC2 instances can be created in the following Availability
Zones:"
                << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "    " << i + 1 << ". "
                    << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
}

```



```

        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

int availabilityZoneChoice = askQuestionForIntRange(
    "Choose an Availability Zone: ", 1,
    static_cast<int>(availabilityZones.size()));
// 4. Create an Auto Scaling group with the specified Availability Zone.
{
    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    Aws::Vector<Aws::String> availabilityGroupZones;
    availabilityGroupZones.push_back(
        availabilityZones[availabilityZoneChoice - 1].GetZoneName());
    request.SetAvailabilityZones(availabilityGroupZones);
    request.SetMaxSize(1);
    request.SetMinSize(1);

    Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
    launchTemplateSpecification.SetLaunchTemplateName(templateName);
    request.SetLaunchTemplate(launchTemplateSpecification);

    Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
        autoScalingClient.CreateAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Created Auto Scaling group '" << groupName << "'..."
            << std::endl;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
        std::cout << "Auto Scaling group '" << groupName << "' already exists."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}
}

```

```
Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                       autoScalingClient)) {
    std::cout << "Here is the Auto Scaling group description." << std::endl;
    if (!autoScalingGroups.empty()) {
        logAutoScalingGroupInfo(autoScalingGroups);
    }
}
else {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

std::cout
    << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
    << std::endl;
if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

bool enableMetrics = askYesNoQuestion(
    "Do you want to collect metrics about the A"
    "Auto Scaling group during this demo (y/n)? ");
// 7. Optionally enable metrics collection for the Auto Scaling group.
if (enableMetrics) {
    Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    request.AddMetrics("GroupMinSize");
    request.AddMetrics("GroupMaxSize");
    request.AddMetrics("GroupDesiredCapacity");
    request.AddMetrics("GroupInServiceInstances");
    request.AddMetrics("GroupTotalInstances");
    request.SetGranularity("1Minute");

    Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
        autoScalingClient.EnableMetricsCollection(request);
    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling metrics have been enabled."
            << std::endl;
    }
}
```

```
    else {
        std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
    "' from 1 to 3." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 8. Update the Auto Scaling group, setting a new maximum size.
{
    Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetMaxSize(3);

    Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
        autoScalingClient.UpdateAutoScalingGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                       autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        const auto &instances = autoScalingGroups[0].GetInstances();
        std::cout
            << "The group still has one running EC2 instance, but it can
have up to 3.\n"
            << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
```

```

        << "No EC2 launch groups were retrieved from DescribeGroup
request."
        << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;
std::cout << "Let's update the desired capacity in '" << groupName <<
    "' from 1 to 2." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 9. Update the Auto Scaling group, setting a new desired capacity.
{
    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
        autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        std::cout
            << "Here is the current state of the group." << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    }
}

```

```

        return false;
    }
}

std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;

std::cout << "Let's terminate one of the EC2 instances in " << groupName << "."
    << std::endl;
std::cout << "Because the desired capacity is 2, another EC2 instance will start
"
    << "to replace the terminated EC2 instance."
    << std::endl;
std::cout << "The currently running EC2 instances are:" << std::endl;

if (autoScalingGroups.empty()) {
    std::cerr << "Error describing groups. No groups returned." << std::endl;
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

int instanceNumber = 1;
Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
    autoScalingGroups[0].GetInstances());
for (const Aws::String &instanceID: instanceIDs) {
    std::cout << "    " << instanceNumber << ". " << instanceID << std::endl;
    ++instanceNumber;
}

instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to stop?
",
    1,
    static_cast<int>(instanceIDs.size()));

// 10. Terminate an EC2 instance in the Auto Scaling group.
{
    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
    request.SetInstanceId(instanceIDs[instanceNumber - 1]);
    request.SetShouldDecrementDesiredCapacity(false);
}

```

```

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
    =
        autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
            << instanceIDs[instanceNumber - 1] << "' to terminate..."
            << std::endl;
    }
    else {
        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;
std::cout << "Let's get a report of scaling activities for EC2 launch group '"
    << groupName << "'."
    << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 11. Get a description of activities for the Auto Scaling group.
{
    Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
    Aws::String nextToken; // Used for pagination;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
            autoScalingClient.DescribeScalingActivities(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                outcome.GetResult().GetActivities();

```

```

        allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                << outcome.GetError().GetMessage()
                << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
        return false;
    }
} while (!nextToken.empty());

std::cout << "Found " << allActivities.size() << " activities."
        << std::endl;
std::cout << "Activities are ordered with the most recent first."
        << std::endl;
for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
    std::cout << activity.GetDescription() << std::endl;
    std::cout << activity.GetDetails() << std::endl;
}
}

if (enableMetrics) {
    if (!logAutoScalingMetrics(groupName, clientConfig)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Let's clean up." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);

// 13. Disable metrics collection if enabled.
if (enableMetrics) {
    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
        autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
}

```

```

    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

return cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                             const Aws::AutoScaling::AutoScalingClient
&client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
    }
}

```



```

        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." << std::endl;
                }

                continue;
            }
        }
    }

    // 6. Check lifecycle state of the instances using
DescribeAutoScalingInstances.
    Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
    request.SetInstanceIds(instanceIDs);

    Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
        client.DescribeAutoScalingInstances(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
            outcome.GetResult().GetAutoScalingInstances();
        ready = instancesDetails.size() >= desiredCapacity;
        for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails &details:
instancesDetails) {
            if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                ready = false;
                break;
            }
        }
        // Log the status while waiting.
        if (((count % 5) == 1) || ready) {
            logInstancesLifecycleState(instancesDetails);
        }
    }

```

```

    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

if (!describeGroup(groupName, autoScalingGroups, client)) {
    return false;
}

return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const Aws::AutoScaling::AutoScalingClient
&autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client) {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
            Aws::String("Delete the Auto Scaling group '" + groupName +
                "' (y/n)?")))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =

```

```

        autoScalingClient.UpdateAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "The minimum size and desired capacity of the Auto
Scaling group "
            << "was set to zero before terminating the instances."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
            << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
        for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
    request.SetInstanceId(instanceID);
    request.SetShouldDecrementDesiredCapacity(true);

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
        autoScalingClient.TerminateInstanceInAutoScalingGroup(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "Initiating termination of EC2 instance '"
                << instanceID << "'." << std::endl;
        }
        else {
            std::cerr
                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }
}

```

```

    }
}

std::cout
    << "Waiting for the EC2 instances to terminate before deleting
the "
    << "Auto Scaling group..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);
}

{
    Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
        autoScalingClient.DeleteAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling group '" << groupName << "' was deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}
}

// 15. Delete the EC2 launch template.
if (!templateName.empty() && (askYesNoQuestion(
    Aws::String("Delete the EC2 launch template '" + templateName +
    "' (y/n)?"))) {
    Aws::EC2::Model::DeleteLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
        ec2Client.DeleteLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "EC2 launch template '" << templateName << "' was deleted."
            << std::endl;
    }
}

```

```

        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
        const Aws::AutoScaling::AutoScalingClient
&client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

CloudTrail examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with CloudTrail.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateTrail

The following code example shows how to use CreateTrail.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Routine which creates an AWS CloudTrail trail.
/#!
 \param trailName: The name of the CloudTrail trail.
 \param bucketName: The Amazon S3 bucket designate for publishing logs.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateTrail](#) in *AWS SDK for C++ API Reference*.

DeleteTrail

The following code example shows how to use DeleteTrail.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Routine which deletes an AWS CloudTrail trail.
/*!
 \param trailName: The name of the CloudTrail trail.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```


- For API details, see [DeleteTrail](#) in *AWS SDK for C++ API Reference*.

DescribeTrail

The following code example shows how to use DescribeTrail.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Routine which describes the AWS CloudTrail trails in an account.
/!*
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/

bool AwsDoc::CloudTrail::describeTrails(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;

    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." << outcome.GetError().GetMessage()
            << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see [DescribeTrail](#) in *AWS SDK for C++ API Reference*.

LookupEvents

The following code example shows how to use `LookupEvents`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Routine which looks up events captured by AWS CloudTrail.
/!*
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::lookupEvents(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;

    size_t count = 0;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
cloudtrail.LookupEvents(
    request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
outcome.GetResult().GetEvents();
```

```
        count += events.size();
        allEvents.insert(allEvents.end(), events.begin(), events.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} while (!nextToken.empty() && count <= 50); // Limit to 50 events.

std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

for (auto &event: allEvents) {
    std::cout << "Event name: " << event.GetEventName() << std::endl;
    std::cout << "Event source: " << event.GetEventSource() << std::endl;
    std::cout << "Event id: " << event.GetEventId() << std::endl;
    std::cout << "Resources: " << std::endl;
    for (auto &resource: event.GetResources()) {
        std::cout << " " << resource.GetResourceName() << std::endl;
    }
}

return true;
}
```

- For API details, see [LookupEvents](#) in *AWS SDK for C++ API Reference*.

CloudWatch examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with CloudWatch.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DeleteAlarms

The following code example shows how to use DeleteAlarms.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Delete the alarm.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to delete CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
```

```
        << std::endl;  
    }
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for C++ API Reference*.

DescribeAlarmsForMetric

The following code example shows how to use `DescribeAlarmsForMetric`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>  
#include <aws/monitoring/CloudWatchClient.h>  
#include <aws/monitoring/model/DescribeAlarmsRequest.h>  
#include <aws/monitoring/model/DescribeAlarmsResult.h>  
#include <iomanip>  
#include <iostream>
```

Describe the alarms.

```
Aws::CloudWatch::CloudWatchClient cw;  
Aws::CloudWatch::Model::DescribeAlarmsRequest request;  
request.SetMaxRecords(1);  
  
bool done = false;  
bool header = false;  
while (!done)  
{  
    auto outcome = cw.DescribeAlarms(request);  
    if (!outcome.IsSuccess())
```

```
{
    std::cout << "Failed to describe CloudWatch alarms:" <<
        outcome.GetError().GetMessage() << std::endl;
    break;
}

if (!header)
{
    std::cout << std::left <<
        std::setw(32) << "Name" <<
        std::setw(64) << "Arn" <<
        std::setw(64) << "Description" <<
        std::setw(20) << "LastUpdated" <<
        std::endl;
    header = true;
}

const auto &alarms = outcome.GetResult().GetMetricAlarms();
for (const auto &alarm : alarms)
{
    std::cout << std::left <<
        std::setw(32) << alarm.GetAlarmName() <<
        std::setw(64) << alarm.GetAlarmArn() <<
        std::setw(64) << alarm.GetAlarmDescription() <<
        std::setw(20) <<
        alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
            SIMPLE_DATE_FORMAT_STR) <<
        std::endl;
}

const auto &next_token = outcome.GetResult().GetNextToken();
request.SetNextToken(next_token);
done = next_token.empty();
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for C++ API Reference*.

DisableAlarmActions

The following code example shows how to use `DisableAlarmActions`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Disable the alarm actions.

```
    Aws::CloudWatch::CloudWatchClient cw;

    Aws::CloudWatch::Model::DisableAlarmActionsRequest
    disableAlarmActionsRequest;
    disableAlarmActionsRequest.AddAlarmNames(alarm_name);

    auto disableAlarmActionsOutcome =
    cw.DisableAlarmActions(disableAlarmActionsRequest);
    if (!disableAlarmActionsOutcome.IsSuccess())
    {
        std::cout << "Failed to disable actions for alarm " << alarm_name <<
            ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
            std::endl;
    }
    else
    {
        std::cout << "Successfully disabled actions for alarm " <<
            alarm_name << std::endl;
    }
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for C++ API Reference*.

EnableAlarmActions

The following code example shows how to use `EnableAlarmActions`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Enable the alarm actions.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
```



```
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for C++ API Reference*.

ListMetrics

The following code example shows how to use `ListMetrics`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
```

```
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

List the metrics.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}

if (argc > 2)
{
    request.SetNamespace(argv[2]);
}

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.ListMetrics(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list CloudWatch metrics:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left << std::setw(48) << "MetricName" <<
            std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
            std::endl;
        header = true;
    }

    const auto &metrics = outcome.GetResult().GetMetrics();
```

```

    for (const auto &metric : metrics)
    {
        std::cout << std::left << std::setw(48) <<
            metric.GetMetricName() << std::setw(32) <<
            metric.GetNamespace();
        const auto &dimensions = metric.GetDimensions();
        for (auto iter = dimensions.cbegin();
            iter != dimensions.cend(); ++iter)
        {
            const auto &dimkv = *iter;
            std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
            if (iter + 1 != dimensions.cend())
            {
                std::cout << ", ";
            }
        }
        std::cout << std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}

```

- For API details, see [ListMetrics](#) in *AWS SDK for C++ API Reference*.

PutMetricAlarm

The following code example shows how to use PutMetricAlarm.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
```

```
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Create the alarm to watch the metric.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch alarm " << alarm_name
        << std::endl;
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for C++ API Reference*.

PutMetricData

The following code example shows how to use PutMetricData.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Put data into the metric.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully put sample metric data" << std::endl;
    }
}
```

- For API details, see [PutMetricData](#) in *AWS SDK for C++ API Reference*.

CloudWatch Logs examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with CloudWatch Logs.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DeleteSubscriptionFilter

The following code example shows how to use DeleteSubscriptionFilter.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/Utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Delete the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetLogGroupName(log_group);

auto outcome = cwl.DeleteSubscriptionFilter(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
} else {
    std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for C++ API Reference*.

DescribeSubscriptionFilters

The following code example shows how to use `DescribeSubscriptionFilters`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

List the subscription filters.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
request.SetLogGroupName(log_group);
request.SetLimit(1);

bool done = false;
bool header = false;
while (!done) {
    auto outcome = cwl.DescribeSubscriptionFilters(
        request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to describe CloudWatch subscription filters "
            << "for log group " << log_group << ": " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
            std::setw(64) << "FilterPattern" << std::setw(64) <<
            "DestinationArn" << std::endl;
        header = true;
    }

    const auto &filters = outcome.GetResult().GetSubscriptionFilters();
    for (const auto &filter : filters) {
        std::cout << std::left << std::setw(32) <<
            filter.GetFilterName() << std::setw(64) <<
            filter.GetFilterPattern() << std::setw(64) <<
            filter.GetDestinationArn() << std::endl;
    }
}
```



```
    }  
  
    const auto &next_token = outcome.GetResult().GetNextToken();  
    request.SetNextToken(next_token);  
    done = next_token.empty();  
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for C++ API Reference*.

PutSubscriptionFilter

The following code example shows how to use `PutSubscriptionFilter`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>  
#include <aws/logs/CloudWatchLogsClient.h>  
#include <aws/logs/model/PutSubscriptionFilterRequest.h>  
#include <aws/core/Utils/Outcome.h>  
#include <iostream>
```

Create the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;  
Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;  
request.SetFilterName(filter_name);  
request.SetFilterPattern(filter_pattern);  
request.SetLogGroupName(log_group);  
request.SetDestinationArn(dest_arn);  
auto outcome = cwl.PutSubscriptionFilter(request);
```

```
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch logs subscription filter "
              << filter_name << ": " << outcome.GetError().GetMessage() <<
              std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch logs subscription " <<
              "filter " << filter_name << std::endl;
}
```

- For API details, see [PutSubscriptionFilter](#) in *AWS SDK for C++ API Reference*.

CodeBuild examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with CodeBuild.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics


- [Actions](#)

Actions

ListBuilds

The following code example shows how to use `ListBuilds`.

SDK for C++

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! List the CodeBuild builds.
/*!
 \param sortType: 'SortOrderType' type.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
            listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            std::cout << "Information about each build:" << std::endl;
            Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
            getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
            Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                getBuildsRequest);

            if (getBuildsOutcome.IsSuccess()) {
```

```
        const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
        std::cout << builds.size() << " build(s) found." << std::endl;
        for (auto val: builds) {
            std::cout << val.GetId() << std::endl;
        }
    }
    else {
        std::cout << "Error getting builds"
            << getBuildsOutcome.GetError().GetMessage() << std::endl;
        return false;
    }
    nextToken = listBuildsOutcome.GetResult().GetNextToken();
}

else {
    std::cerr << "Error listing builds"
        << listBuildsOutcome.GetError().GetMessage()
        << std::endl;
    return false;
}

} while (!nextToken.empty());

return true;
}
```

- For API details, see [ListBuilds](#) in *AWS SDK for C++ API Reference*.

ListProjects

The following code example shows how to use ListProjects.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ List the CodeBuild projects.
/*!
 \param sortType: 'SortOrderType' type.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
codeBuildClient.ListProjects(
            listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(), projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" << outcome.GetError().GetMessage()
                << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }
}
```

```

    }

    return true;
}

```

- For API details, see [ListProjects](#) in *AWS SDK for C++ API Reference*.

StartBuild

The following code example shows how to use StartBuild.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Start an AWS CodeBuild project build.
/*!
 \param projectName: A CodeBuild project name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome = codeBuildClient.StartBuild(
        startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
            << std::endl;
    }
}

```

```
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [StartBuild](#) in *AWS SDK for C++ API Reference*.

Amazon Cognito Identity Provider examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Cognito Identity Provider.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Cognito

The following code examples show how to get started using Amazon Cognito.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

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

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
  hello_cognito.cpp)
```



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

Code for the hello_cognito.cpp source file.

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

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

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

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

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

```
        listUserPoolsRequest.SetNextToken(nextToken);
    }

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

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

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

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

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

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

- For API details, see [ListUserPools](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

AdminGetUser

The following code example shows how to use AdminGetUser.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

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

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

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

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

- For API details, see [AdminGetUser](#) in *AWS SDK for C++ API Reference*.

AdminInitiateAuth

The following code example shows how to use AdminInitiateAuth.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

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

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

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

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

```
}

```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for C++ API Reference*.

AdminRespondToAuthChallenge

The following code example shows how to use `AdminRespondToAuthChallenge`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
request.AddChallengeResponses("USERNAME", userName);
request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
request.SetClientId(clientID);
request.SetUserPoolId(userPoolID);
request.SetSession(session);

Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =

    client.AdminRespondToAuthChallenge(request);

if (outcome.IsSuccess()) {
    std::cout << "Here is the response to the challenge.\n" <<

```

```

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
    << std::endl;

    accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
}
else {
    std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
    << outcome.GetError().GetMessage()
    << std::endl;
    return false;
}

```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for C++ API Reference*.

AssociateSoftwareToken

The following code example shows how to use AssociateSoftwareToken.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

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

Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
request.SetSession(session);

Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =

```

```

        client.AssociateSoftwareToken(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "Enter this setup key into an authenticator app, for example
Google Authenticator."
            << std::endl;
        std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
            << std::endl;
#ifdef USING_QR
        printAsterisksLine();
        std::cout << "\n0r scan the QR code in the file '" << QR_CODE_PATH <<
            ". "
            << std::endl;

        saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
            outcome.GetResult().GetSecretCode());
#endif // USING_QR
        session = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for C++ API Reference*.

ConfirmSignUp

The following code example shows how to use `ConfirmSignUp`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
request.SetClientId(clientID);
request.SetConfirmationCode(confirmationCode);
request.SetUsername(userName);

Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
    client.ConfirmSignUp(request);

if (outcome.IsSuccess()) {
    std::cout << "ConfirmSignup was Successful."
              << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for C++ API Reference*.

DeleteUser

The following code example shows how to use DeleteUser.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


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

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

Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
request.SetAccessToken(accessToken);

Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
    client.DeleteUser(request);

if (outcome.IsSuccess()) {
    std::cout << "The user " << userName << " was deleted."
              << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for C++ API Reference*.

ResendConfirmationCode

The following code example shows how to use ResendConfirmationCode.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

    Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
    request.SetUsername(userName);
    request.SetClientId(clientID);

    Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
        client.ResendConfirmationCode(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
            << std::endl;
    }
    else {
        std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for C++ API Reference*.

SignUp

The following code example shows how to use SignUp.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

```

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

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

Aws::CognitoIdentityProvider::Model::SignUpRequest request;
request.AddUserAttributes(
    Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
        "email").WithValue(email));
request.SetUsername(userName);
request.SetPassword(password);
request.SetClientId(clientID);
Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
    client.SignUp(request);

if (outcome.IsSuccess()) {
    std::cout << "The signup request for " << userName << " was successful."
        << std::endl;
}
else if (outcome.GetError().GetErrorType() ==
Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
    std::cout
        << "The username already exists. Please enter a different
username."
        << std::endl;
    userExists = true;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}

```

- For API details, see [SignUp](#) in *AWS SDK for C++ API Reference*.

VerifySoftwareToken

The following code example shows how to use `VerifySoftwareToken`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
request.SetUserCode(userCode);
request.SetSession(session);

Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
    client.VerifySoftwareToken(request);

if (outcome.IsSuccess()) {
    std::cout << "Verification of the code was successful."
              << std::endl;
    session = outcome.GetResult().GetSession();
}
else {
    std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for C++ API Reference*.

Scenarios

Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up and confirm a user with a username, password, and email address.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
 \sa gettingStartedWithUserPools()
 \param clientID: Client ID associated with an Amazon Cognito user pool.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param clientConfig: Aws client configuration.
 \return bool: Successful completion.
*/
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
                                                  Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
        << "Welcome to the Amazon Cognito example scenario."
        << std::endl;
    printAsterisksLine();

    std::cout
```

```
        << "This scenario will add a user to an Amazon Cognito user pool."
        << std::endl;
const Aws::String userName = askQuestion("Enter a new username: ");
const Aws::String password = askQuestion("Enter a new password: ");
const Aws::String email = askQuestion("Enter a valid email for the user: ");

std::cout << "Signing up " << userName << std::endl;

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
bool userExists = false;
do {
    // 1. Add a user with a username, password, and email address.
    Aws::CognitoIdentityProvider::Model::SignUpRequest request;
    request.AddUserAttributes(
        Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
            "email").WithValue(email));
    request.SetUsername(userName);
    request.SetPassword(password);
    request.SetClientId(clientID);
    Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
        client.SignUp(request);

    if (outcome.IsSuccess()) {
        std::cout << "The signup request for " << userName << " was successful."
            << std::endl;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
        std::cout
            << "The username already exists. Please enter a different
username."
            << std::endl;
        userExists = true;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (userExists);
```

```
printAsterisksLine();
std::cout << "Retrieving status of " << userName << " in the user pool."
    << std::endl;
// 2. Confirm that the user was added to the user pool.
if (!checkAdminUserStatus(userName, userPoolID, client)) {
    return false;
}

std::cout << "A confirmation code was sent to " << email << "." << std::endl;

bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
if (resend) {
    // Request a resend of the confirmation code to the email address.
    (ResendConfirmationCode)
    Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
    request.SetUsername(userName);
    request.SetClientId(clientID);

    Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
        client.ResendConfirmationCode(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
            << std::endl;
    }
    else {
        std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

printAsterisksLine();

{
    // 4. Send the confirmation code that's received in the email.
    (ConfirmSignUp)
    const Aws::String confirmationCode = askQuestion(
        "Enter the confirmation code that was emailed: ");
    Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
```

```
request.SetClientId(clientID);
request.SetConfirmationCode(confirmationCode);
request.SetUsername(userName);

Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
    client.ConfirmSignUp(request);

if (outcome.IsSuccess()) {
    std::cout << "ConfirmSignup was Successful."
              << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
}

std::cout << "Rechecking the status of " << userName << " in the user pool."
          << std::endl;
if (!checkAdminUserStatus(userName, userPoolID, client)) {
    return false;
}

printAsterisksLine();

std::cout << "Initiating authorization using the username and password."
          << std::endl;

Aws::String session;
// 5. Initiate authorization with username and password. (AdminInitiateAuth)
if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
    return false;
}

printAsterisksLine();

std::cout
    << "Starting setup of time-based one-time password (TOTP) multi-factor
authentication (MFA)."
    << std::endl;
```



```

{
    // 6. Request a setup key for one-time password (TOTP)
    //    multi-factor authentication (MFA). (AssociateSoftwareToken)
    Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
    request.SetSession(session);

    Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
        client.AssociateSoftwareToken(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "Enter this setup key into an authenticator app, for example
Google Authenticator."
            << std::endl;
        std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
            << std::endl;
#ifdef USING_QR
        printAsterisksLine();
        std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
"."
            << std::endl;

        saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
            outcome.GetResult().GetSecretCode());
#endif // USING_QR
        session = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}
askQuestion("Type enter to continue...", alwaysTrueTest);

printAsterisksLine();

{
    Aws::String userCode = askQuestion(
        "Enter the 6 digit code displayed in the authenticator app: ");
}

```

```
    // 7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
    Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
    request.SetUserCode(userCode);
    request.SetSession(session);

    Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
        client.VerifySoftwareToken(request);

    if (outcome.IsSuccess()) {
        std::cout << "Verification of the code was successful."
                  << std::endl;
        session = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "You have completed the MFA authentication setup." << std::endl;
std::cout << "Now, sign in." << std::endl;

// 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
            "Re-enter the 6 digit code displayed in the authenticator app: ");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```

        request.SetChallengeName(
            Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
            client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<
                outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                    << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                << outcome.GetError().GetMessage()
                    << std::endl;
            return false;
        }

        std::cout << "You have successfully added a user to Amazon Cognito."
            << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added? (y/
n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
            client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                << std::endl;
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
}

return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

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

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

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

    return outcome.IsSuccess();
}

```

```

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                const Aws::String &userPoolID,
                                                const Aws::String &userName,
                                                const Aws::String &password,
                                                Aws::String &sessionResult,
                                                const
Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

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

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

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

    return outcome.IsSuccess();
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

DynamoDB examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with DynamoDB.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello DynamoDB

The following code examples show how to get started using DynamoDB.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS dynamodb)

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

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

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

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

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

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

  # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
  need to uncomment this
```

```

                                # and set the proper subdirectory to the
executables' location.

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

add_executable(${PROJECT_NAME}
    hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_dynamodb.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
 * A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 * (DynamoDB) client and lists the
 * DynamoDB tables.
 *
 * main function
 *
 * Usage: 'hello_dynamodb'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
    }
}

```



```
Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
listTablesRequest.SetLimit(50);
do {
    const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamodbClient.ListTables(
        listTablesRequest);
    if (!outcome.IsSuccess()) {
        std::cout << "Error: " << outcome.GetError().GetMessage() <<
std::endl;
        result = 1;
        break;
    }

    for (const auto &tableName: outcome.GetResult().GetTableNames()) {
        std::cout << tableName << std::endl;
    }

    listTablesRequest.SetExclusiveStartTableName(
        outcome.GetResult().GetLastEvaluatedTableName());
} while (!listTablesRequest.GetExclusiveStartTableName().empty());

}

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

- For API details, see [ListTables](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

BatchExecuteStatement

The following code example shows how to use BatchExecuteStatement.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
// 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

std::vector<Aws::String> titles;
std::vector<float> ratings;
std::vector<int> years;
std::vector<Aws::String> plots;
Aws::String doAgain = "n";
do {
    Aws::String aTitle = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    titles.push_back(aTitle);
    int aYear = askQuestionForInt("What year was it released? ");
    years.push_back(aYear);
    float aRating = askQuestionForFloatRange(
        "On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    ratings.push_back(aRating);
    Aws::String aPlot = askQuestion("Summarize the plot for me: ");
    plots.push_back(aPlot);

    doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
} while (doAgain == "y");

std::cout << "Adding " << titles.size()
    << (titles.size() == 1 ? " movie " : " movies ")
```

```

        << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
            << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
            << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

            attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
            Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
            Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
    }

```

```

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
        << std::endl;
        return false;
    }
}
}

```

Use batches of SELECT statements to get items.

```

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \" << MOVIE_TABLE_NAME << "\" WHERE \"
        << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
    if (outcome.IsSuccess()) {

```

```

        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    }
else {
    std::cerr << "Failed to retrieve the movie information: "
              << outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}

```

Use batches of UPDATE statements to update items.

```

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"" + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i])
        + ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "

```

```

        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);
    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

Use batches of DELETE statements to delete items.

```

// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"\" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";
}

```

```

std::string sql(sqlStream.str());

for (size_t i = 0; i < statements.size(); ++i) {
    statements[i].SetStatement(sql);
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(
        Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
    statements[i].SetParameters(attributes);
}

Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

request.SetStatements(statements);

Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete the movies: "
        << outcome.GetError().GetMessage() << std::endl;
    return false;
}
}

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

BatchGetItem

The following code example shows how to use BatchGetItem.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Batch get items from different Amazon DynamoDB tables.
/*!
 \sa batchGetItem()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::batchGetItem(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages, #v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");
```



```

// Table2: Set key name, type, and value to search.
std::vector<Aws::String> idValues = {"102", "103", "201"};
for (const Aws::String &id: idValues) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
    Aws::DynamoDB::Model::AttributeValue key;
    key.SetN(id);
    keys.emplace("Id", key);
    table2KeysAndAttributes.AddKeys(keys);
}

requestItems.emplace(table2Name, table2KeysAndAttributes);

bool result = true;
do { // Use a do loop to handle pagination.
    request.SetRequestItems(requestItems);
    const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
    request);

    if (outcome.IsSuccess()) {
        for (const auto &responsesMapEntry: outcome.GetResult().GetResponses())
        {
            Aws::String tableName = responsesMapEntry.first;
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
            std::cout << "Retrieved " << tableResults.size()
                << " responses for table '" << tableName << "'.\n"
                << std::endl;
            if (tableName == "Forum") {

                std::cout << "Name | Category | Message | Views" << std::endl;
                for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                    std::cout << item.at("Name").GetS() << " | ";
                    std::cout << item.at("Category").GetS() << " | ";
                    std::cout << (item.count("Message") == 0 ? "" : item.at(
                        "Messages").GetN()) << " | ";
                    std::cout << (item.count("Views") == 0 ? "" : item.at(
                        "Views").GetN()) << std::endl;
                }
            }
            else {
                std::cout << "Title | Price | Color" << std::endl;

```

```

        for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
            std::cout << item.at("Title").GetS() << " | ";
            std::cout << (item.count("Price") == 0 ? "" : item.at(
                "Price").GetN());
            if (item.count("Color")) {
                std::cout << " | ";
                for (const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
                    "Color").GetL())
                    std::cout << listItem->GetS() << " ";
            }
            std::cout << std::endl;
        }
        std::cout << std::endl;
    }

    // If necessary, repeat request for remaining items.
    requestItems = outcome.GetResult().GetUnprocessedKeys();
}
else {
    std::cerr << "Batch get item failed: " <<
outcome.GetError().GetMessage()
        << std::endl;
    result = false;
    break;
}
} while (!requestItems.empty());

return result;
}

```

- For API details, see [BatchGetItem](#) in *AWS SDK for C++ API Reference*.

BatchWriteItem

The following code example shows how to use `BatchWriteItem`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Batch write items from a JSON file.
/*!
  \sa batchWriteItem()
  \param jsonFilePath: JSON file path.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/

/*
 * The input for this routine is a JSON file that you can download from the
 * following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 * generate
 * JSON strings when constructing the BatchWriteItem request, essentially outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
}
```

```
Aws::Utils::Json::JsonValue jsonValue(stringStream);

Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
jsonValue.View().GetAllObjects();
for (const auto &level1Entry: level1Map) {
    const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
    const Aws::String &tableName = level1Entry.first;
    // The JSON entries at this level are as follows:
    // key - table name
    // value - list of request objects
    if (!entriesView.IsListType()) {
        std::cerr << "Error: JSON file entry '"
            << tableName << "' is not a list." << std::endl;
        continue;
    }

    Aws::Utils::Array<Aws::Utils::Json::JsonView> entries =
entriesView.AsArray();

    Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
    if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
        writeRequests)) {
        batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
    }
}

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
dynamoClient.BatchWriteItem(
    batchWriteItemRequest);

if (outcome.IsSuccess()) {
    std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
}
else {
    std::cerr << "Error with DynamoDB::BatchWriteItem. "
        << outcome.GetError().GetMessage()
        << std::endl;
}

return true;
}
```

```

/*! Convert requests in JSON format to a vector of WriteRequest objects.
*/
\sa addWriteRequests()
\param tableName: Name of the table for the write operations.
\param requestsJson: Request data in JSON format.
\param writeRequests: Vector to receive the WriteRequest objects.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                         const
                                         Aws::Utils::Array<Aws::Utils::Json::JsonValue> &requestsJson,

                                         Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonValue &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                       << requestsEntry.WriteReadable() << std::endl;
            return false;
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> requestsMap =
requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonValue &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                                << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                                           attributes)) {
                    std::cerr << "Error getting attributes "
                                << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
            }
        }
    }
}

```

```

        Aws::DynamoDB::Model::PutRequest putRequest;
        putRequest.SetItem(attributes);
        writeRequests.push_back(
            Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                putRequest));
    }
    else {
        std::cerr << "Error: unimplemented request type '" << requestType
            << "'." << std::endl;
    }
}
}

return true;
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
 \sa getAttributeObjectsMap()
 \param jsonView: JSONView of attribute records.
 \param writeRequests: Map to receive the AttributeValue objects.
 \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView &jsonView,
                                           Aws::Map<Aws::String,
                                           Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
        jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,
            Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}

```

```
}

```

- For API details, see [BatchWriteItem](#) in *AWS SDK for C++ API Reference*.

CreateTable

The following code example shows how to use CreateTable.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

///! Create an Amazon DynamoDB table.
/*!
 \sa CreateTable()
 \param tableName: Name for the DynamoDB table.
 \param primaryKey: Primary key for the DynamoDB table.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
        " with a simple primary key: \"" << primaryKey << "\"." << std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;

```

```

keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(keySchemaElement);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
request.SetProvisionedThroughput(throughput);
request.SetTableName(tableName);

const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
dynamoClient.CreateTable(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Table \""
        << outcome.GetResult().GetTableDescription().GetTableName() <<
        " created!" << std::endl;
}
else {
    std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
        << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [CreateTable](#) in *AWS SDK for C++ API Reference*.

DeleteItem

The following code example shows how to use DeleteItem.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Delete an item from an Amazon DynamoDB table.

```



```

/ *!
 \sa deleteItem()
 \param tableName: The table name.
 \param partitionKey: The partition key.
 \param partitionValue: The value for the partition key.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
dynamoClient.DeleteItem(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"\" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
<< std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteItem](#) in *AWS SDK for C++ API Reference*.

DeleteTable

The following code example shows how to use DeleteTable.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Delete an Amazon DynamoDB table.
/*!
 \sa deleteTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                    << result.GetResult().GetTableDescription().GetTableName()
                    << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                    << std::endl;
    }

    return result.IsSuccess();
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for C++ API Reference*.

DescribeTable

The following code example shows how to use DescribeTable.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Describe an Amazon DynamoDB table.
/#!
 \sa describeTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
dynamoClient.DescribeTable(
    request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
outcome.GetResult().GetTable();
        std::cout << "Table name   : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN    : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
            << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;
    }
}
```

```

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
            a.GetAttributeType()) <<
            ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DescribeTable](#) in *AWS SDK for C++ API Reference*.

ExecuteStatement

The following code example shows how to use ExecuteStatement.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

// 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
Aws::String title;
float rating;
int year;
Aws::String plot;
{
    title = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    year = askQuestionForInt("What year was it released? ");
    rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                     1, 10);
    plot = askQuestion("Summarize the plot for me: ");

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {" <<
        << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
        << INFO_KEY << "': ?}";

    request.SetStatement(sqlStream.str());

    // Create the parameter attributes.
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(rating);
    infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    plotAttribute->SetS(plot);
    infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
    attributes.push_back(infoMapAttribute);
}

```

```

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

Use a SELECT statement to get an item.

```

// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.

```

```

        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}

```

Use an UPDATE statement to update an item.

```

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);
}

```

```

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

```

Use a DELETE statement to delete an item.

```

// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"\" << MOVIE_TABLE_NAME << "\" WHERE \"
        << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movie: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
}

```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

GetItem

The following code example shows how to use `GetItem`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Get an item from an Amazon DynamoDB table.
/*!
  \sa getItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                  Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
outcome.GetResult().GetItem();
        if (!item.empty()) {
```

```

        // Output each retrieved field and its value.
        for (const auto &i: item)
            std::cout << "Values: " << i.first << ": " << i.second.GetS()
                << std::endl;
    }
    else {
        std::cout << "No item found with the key " << partitionKey << std::endl;
    }
}
else {
    std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
}

return outcome.IsSuccess();
}

```

- For API details, see [GetItem](#) in *AWS SDK for C++ API Reference*.

ListTables

The following code example shows how to use ListTables.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! List the Amazon DynamoDB tables for the current AWS account.
 *!
 * \sa listTables()
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

```

```

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamoClient.ListTables(
            listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
            outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}

```

- For API details, see [ListTables](#) in *AWS SDK for C++ API Reference*.

PutItem

The following code example shows how to use PutItem.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Put an item in an Amazon DynamoDB table.
/*!
    \sa putItem()
    \param tableName: The table name.
    \param artistKey: The artist key. This is the partition key for the table.
    \param artistValue: The artist value.

```

```

\param albumTitleKey: The album title key.
\param albumTitleValue: The album title value.
\param awardsKey: The awards key.
\param awardsValue: The awards value.
\param songTitleKey: The song title key.
\param songTitleValue: The song title value.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
                               const Aws::String &awardsKey,
                               const Aws::String &awardsValue,
                               const Aws::String &songTitleKey,
                               const Aws::String &songTitleValue,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
        artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
    Aws::DynamoDB::Model::AttributeValue().SetS(
        albumTitleValue));
    putItemRequest.AddItem(awardsKey,

    Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

    Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
        putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```
    }  
  
    return outcome.IsSuccess();  
}
```

- For API details, see [PutItem](#) in *AWS SDK for C++ API Reference*.

Query

The following code example shows how to use Query.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Perform a query on an Amazon DynamoDB Table and retrieve items.  
/*!  
  \sa queryItem()  
  \param tableName: The table name.  
  \param partitionKey: The partition key.  
  \param partitionValue: The value for the partition key.  
  \param projectionExpression: The projections expression, which is ignored if  
  empty.  
  \param clientConfiguration: AWS client configuration.  
  \return bool: Function succeeded.  
*/  
  
/*  
 * The partition key attribute is searched with the specified value. By default, all  
 * fields and values  
 * contained in the item are returned. If an optional projection expression is  
 * specified on the command line, only the specified fields and values are  
 * returned.  
 */  
  
bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
```

```
        const Aws::String &partitionKey,
        const Aws::String &partitionValue,
        const Aws::String &projectionExpression,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);

    if (!projectionExpression.empty()) {
        request.SetProjectionExpression(projectionExpression);
    }

    // Set query key condition expression.
    request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

    // Set Expression AttributeValues.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch", partitionValue);

    request.SetExpressionAttributeValues(attributeValues);

    bool result = true;

    // "exclusiveStartKey" is used for pagination.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> exclusiveStartKey;
    do {
        if (!exclusiveStartKey.empty()) {
            request.SetExclusiveStartKey(exclusiveStartKey);
            exclusiveStartKey.clear();
        }
        // Perform Query operation.
        const Aws::DynamoDB::Model::QueryOutcome &outcome =
dynamoClient.Query(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "Number of items retrieved from Query: " <<
items.size()
                    << std::endl;
                // Iterate each item and print.
            }
        }
    } while (result);
}
```

```

        for (const auto &item: items) {
            std::cout
                <<
                "*****"
                << std::endl;
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << i.first << ": " << i.second.GetS() <<
std::endl;
        }
    }
    else {
        std::cout << "No item found in table: " << tableName << std::endl;
    }

    exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
}
else {
    std::cerr << "Failed to Query items: " <<
outcome.GetError().GetMessage();
    result = false;
    break;
}
} while (!exclusiveStartKey.empty());

return result;
}

```

- For API details, see [Query](#) in *AWS SDK for C++ API Reference*.

Scan

The following code example shows how to use Scan.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Scan an Amazon DynamoDB table.
/*!
  \sa scanTable()
  \param tableName: Name for the DynamoDB table.
  \param projectionExpression: An optional projection expression, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/

bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                 const Aws::String &projectionExpression,
                                 const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            all_items.insert(all_items.end(), items.begin(), items.end());

            last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Scan items: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```



```

    } while (!last_evaluated_key.empty());

    if (!all_items.empty()) {
        std::cout << "Number of items retrieved from scan: " << all_items.size()
                  << std::endl;
        // Iterate each item and print.
        for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
            &itemMap: all_items) {
            std::cout << "*****"
                      << std::endl;
            // Output each retrieved field and its value.
            for (const auto &itemEntry: itemMap)
                std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                          << std::endl;
        }
    }

    else {
        std::cout << "No items found in table: " << tableName << std::endl;
    }

    return true;
}

```

- For API details, see [Scan](#) in *AWS SDK for C++ API Reference*.

UpdateItem

The following code example shows how to use UpdateItem.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Update an Amazon DynamoDB table item.

```

```
/*!
 \sa updateItem()
 \param tableName: The table name.
 \param partitionKey: The partition key.
 \param partitionValue: The value for the partition key.
 \param attributeKey: The key for the attribute to be updated.
 \param attributeValue: The value for the attribute to be updated.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

/*
 * The example code only sets/updates an attribute value. It processes
 * the attribute value as a string, even if the value could be interpreted
 * as a number. Also, the example code does not remove an existing attribute
 * from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::String &attributeKey,
                                   const Aws::String &attributeValue,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
}
```

```
request.SetExpressionAttributeNames(expressionAttributeNames);

// Construct attribute value argument.
Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
attributeUpdatedValue.SetS(attributeValue);
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
expressionAttributeValues[":valueA"] = attributeUpdatedValue;
request.SetExpressionAttributeValues(expressionAttributeValues);

// Update the item.
const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
dynamoClient.UpdateItem(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Item was updated" << std::endl;
}
else {
    std::cerr << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for C++ API Reference*.

Scenarios

Get started with tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table, then delete the table.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

{
    Aws::Client::ClientConfiguration clientConfig;
    // 1. Create a table with partition: year (N) and sort: title (S).
(CreateTable)
    if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

        AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

        // 9. Delete the table. (DeleteTable)
        AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
    }
}

//! Scenario to modify and query a DynamoDB table.
/*!
 \sa dynamodbGettingStartedScenario()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;

```

```

    Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                        1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
            outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

```

std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

// 3. Update the rating and plot of the movie by using an update expression.
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);
    plot = askQuestion(Aws::String("You summarized the plot as '" + plot +
        "'.\nWhat would you say now? "));

    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);
    request.AddKey(TITLE_KEY,
        Aws::DynamoDB::Model::AttributeValue().SetS(title));
    request.AddKey(YEAR_KEY, Aws::DynamoDB::Model::AttributeValue().SetN(year));
    std::stringstream expressionStream;
    expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
        << INFO_KEY << "." << PLOT_KEY << " =:p";
    request.SetUpdateExpression(expressionStream.str());
    request.SetExpressionAttributeValues({
        {":r",
        Aws::DynamoDB::Model::AttributeValue().SetN(
            rating)},
        {":p",
        Aws::DynamoDB::Model::AttributeValue().SetS(
            plot)}});

    request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

    const Aws::DynamoDB::Model::UpdateItemOutcome &result =
    dynamoClient.UpdateItem(
        request);
    if (!result.IsSuccess()) {
        std::cerr << "Error updating movie " + result.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

```

```

// 4. Put 250 movies in the table from moviedata.json.
{
    std::cout << "Adding movies from a json file to the database." << std::endl;
    const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
    const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
    Aws::String jsonString = getMovieJSON();
    if (!jsonString.empty()) {
        Aws::Utils::Json::JsonValue json(jsonString);
        Aws::Utils::Array<Aws::Utils::Json::JsonValue> movieJsons =
json.View().AsArray();
        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

        // To add movies with a cross-section of years, use an appropriate
increment
        // value for iterating through the database.
        size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
        for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
            writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> putItems
= movieJsonValueToAttributeMap(
                movieJsons[i]);
            Aws::DynamoDB::Model::PutRequest putRequest;
            putRequest.SetItem(putItems);
            writeRequests.back().SetPutRequest(putRequest);
            if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                Aws::DynamoDB::Model::BatchWriteItemRequest request;
                request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
                const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                    request);
                if (!outcome.IsSuccess()) {
                    std::cerr << "Unable to batch write movie data: "
                        << outcome.GetError().GetMessage()
                        << std::endl;
                    writeRequests.clear();
                    break;
                }
            }
            else {
                std::cout << "Added batch of " << writeRequests.size()
                    << " movies to the database."
                    << std::endl;
            }
            writeRequests.clear();

```

```

    }
    }
}

std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
    << std::endl;

// 5. Get a movie by Key (partition + sort).
{
    Aws::String titleToGet("King Kong");
    Aws::String answer = askQuestion(Aws::String(
        "Let's move on...Would you like to get info about '" + titleToGet +
        "'? (y/n) "));
    if (answer == "y") {
        Aws::DynamoDB::Model::GetItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
        request.AddKey(YEAR_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetN(1933));

        const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
            request);
        if (!result.IsSuccess()) {
            std::cerr << "Error " << result.GetError().GetMessage();
        }
        else {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
            if (!item.empty()) {
                std::cout << "\nHere's what I found:" << std::endl;
                printMovieInfo(item);
            }
            else {
                std::cout << "\nThe movie was not found in the database."
                    << std::endl;
            }
        }
    }
}

// 6. Use Query with a key condition expression to return all movies

```



```
// released in a given year.
Aws::String doAgain = "n";
do {
    Aws::DynamoDB::Model::QueryRequest req;

    req.SetTableName(MOVIE_TABLE_NAME);

    // "year" is a DynamoDB reserved keyword and must be replaced with an
    // expression attribute name.
    req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
    req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

    int yearToMatch = askQuestionForIntRange(
        "\nLet's get a list of movies released in"
        " a given year. Enter a year between 1972 and 2018 ",
        1972, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch",
        Aws::DynamoDB::Model::AttributeValue().SetN(
            yearToMatch));
    req.SetExpressionAttributeValues(attributeValues);

    const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
    if (result.IsSuccess()) {
        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
        if (!items.empty()) {
            std::cout << "\nThere were " << items.size()
                << " movies in the database from "
                << yearToMatch << "." << std::endl;
            for (const auto &item: items) {
                printMovieInfo(item);
            }
            doAgain = "n";
        }
        else {
            std::cout << "\nNo movies from " << yearToMatch
                << " were found in the database"
                << std::endl;
            doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
        }
    }
    else {
        std::cerr << "Failed to Query items: " << result.GetError().GetMessage()
```

```

        << std::endl;
    }

    } while (doAgain == "y");

    // 7. Use Scan to return movies released within a range of years.
    // Show how to paginate data using ExclusiveStartKey. (Scan +
    FilterExpression)
    {
        int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
            "for movies in the database. Enter a
start year: ",
            1972, 2018);
        int endYear = askQuestionForIntRange("\nEnter an end year: ",
            startYear, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
        do {
            Aws::DynamoDB::Model::ScanRequest scanRequest;
            scanRequest.SetTableName(MOVIE_TABLE_NAME);
            scanRequest.SetFilterExpression(
                "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
            scanRequest.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
            attributeValues.emplace(":startYear",
                Aws::DynamoDB::Model::AttributeValue().SetN(
                    startYear));
            attributeValues.emplace(":endYear",
                Aws::DynamoDB::Model::AttributeValue().SetN(
                    endYear));
            scanRequest.SetExpressionAttributeValues(attributeValues);

            if (!exclusiveStartKey.empty()) {
                scanRequest.SetExclusiveStartKey(exclusiveStartKey);
            }

            const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
                scanRequest);
            if (result.IsSuccess()) {
                const Aws::Vector<Aws::Map<Aws::String,
                Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
                if (!items.empty()) {

```

```

        std::stringstream stringstream;
        stringstream << "\nFound " << items.size() << " movies in one
scan."
                << " How many would you like to see? ";
        size_t count = askQuestionForInt(stringStream.str());
        for (size_t i = 0; i < count && i < items.size(); ++i) {
            printMovieInfo(items[i]);
        }
    }
    else {
        std::cout << "\nNo movies in the database between " << startYear
<<
                " and " << endYear << "." << std::endl;
    }

    exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
    if (!exclusiveStartKey.empty()) {
        std::cout << "Not all movies were retrieved. Scanning for more."
                << std::endl;
    }
    else {
        std::cout << "All movies were retrieved with this scan."
                << std::endl;
    }
}
else {
    std::cerr << "Failed to Scan movies: "
                << result.GetError().GetMessage() << std::endl;
}
} while (!exclusiveStartKey.empty());
}

// 8. Delete a movie. (DeleteItem)
{
    std::stringstream stringstream;
    stringstream << "\nWould you like to delete the movie " << title
                << " from the database? (y/n) ";
    Aws::String answer = askQuestion(stringStream.str());
    if (answer == "y") {
        Aws::DynamoDB::Model::DeleteItemRequest request;
        request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.AddKey(TITLE_KEY,
                Aws::DynamoDB::Model::AttributeValue().SetS(title));
    }
}

```

```

        request.SetTableName(MOVIE_TABLE_NAME);

        const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(
            request);
        if (result.IsSuccess()) {
            std::cout << "\nRemoved \"" << title << "\" from the database."
                << std::endl;
        }
        else {
            std::cerr << "Failed to delete the movie: "
                << result.GetError().GetMessage()
                << std::endl;
        }
    }
}

return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
 \sa movieJsonViewToAttributeMap()
 \param jsonView: Json view object.
 \return map: Map that can be used in a DynamoDB request.
 */
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
    const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();

```

```

        attributeValue->SetN(infoView.GetDouble(RATING_KEY));
        infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
    }
    if (infoView.KeyExists(PLOT_KEY)) {
        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
        attributeValue->SetS(infoView.GetString(PLOT_KEY));
        infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
    }

    result[INFO_KEY].SetM(infoMap);
}

return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
 \sa createMoviesDynamoDBTable()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;

```

```

yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorType() ==
        Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
            << result.GetError().GetMessage();
        return false;
    }
}
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}

```

```

    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
 \sa deleteMoviesDynamoDBTable()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
 \sa waitTableActive()
 \param waitTableActive: The DynamoDB table's name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                        const Aws::Client::ClientConfiguration
&clientConfiguration) {

```

```

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
// Repeatedly call DescribeTable until table is ACTIVE.
const int MAX_QUERIES = 20;
Aws::DynamoDB::Model::DescribeTableRequest request;
request.SetTableName(tableName);

int count = 0;
while (count < MAX_QUERIES) {
    const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
        request);
    if (result.IsSuccess()) {
        Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

        if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            return true;
        }
    }
    else {
        std::cerr << "Error DynamoDB::waitTableActive "
            << result.GetError().GetMessage() << std::endl;
        return false;
    }
    count++;
}
return false;
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)

- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

    AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
}

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
    \sa partiqlBatchExecuteScenario()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
```

```

bool AwsDoc::DynamoDB::partIqlBatchExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
            "On a scale of 1 - 10, how do you rate it? ",
            1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
    "));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
        << (titles.size() == 1 ? " movie " : " movies ")
        << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {"
            << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
            << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {

```

```

        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(ratings[i]);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plots[i]);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
        << std::endl;
        return false;
    }
}

    std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
        << std::endl;

```

```

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
        outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
        &responses = result.GetResponse();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
        responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
            &item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {

```

```

        std::cerr << "Failed to retrieve the movie information: "
                << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"" + titles[i] +
            ".\nYou rated it " + std::to_string(ratings[i])
            + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

            attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }
    }
}

```

```

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);
    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

    std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
        << std::endl;

    // 5. Get the updated data for multiple movies using "Select" statements.
    (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM \"\" << MOVIE_TABLE_NAME << "\" WHERE "
            << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(

```

```

        request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {
        std::cerr << "Failed to retrieve the movies information: "
                << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
        << std::endl;

// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"\" << MOVIE_TABLE_NAME << "\" WHERE \"
        << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }
}

```

```

    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movies: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
 \sa createMoviesDynamoDBTable()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(

```



```

        Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(yearAttributeDefinition);

    Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
    yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
    yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::RANGE);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(MOVIE_TABLE_NAME);

    std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
    const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
        request);
    if (!result.IsSuccess()) {
        if (result.GetError().GetErrorType() ==
            Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
            std::cout << "Table already exists." << std::endl;
            movieTableAlreadyExisted = true;
        }
        else {
            std::cerr << "Failed to create table: "
                << result.GetError().GetMessage();
            return false;
        }
    }
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {

```

```

        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}

return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
 \sa deleteMoviesDynamoDBTable()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
 \sa waitTableActive()
 \param waitTableActive: The DynamoDB table's name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/

```

```

*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

    AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
}

/*! Scenario to modify and query a DynamoDB table using single PartiQL statements.
 *!
 * \sa partiqlExecuteScenario()
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
```

```

        "Enter the title of a movie you want to add to the table: ");
year = askQuestionForInt("What year was it released? ");
rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                1, 10);
plot = askQuestion("Summarize the plot for me: ");

Aws::DynamoDB::Model::ExecuteStatementRequest request;
std::stringstream sqlStream;
sqlStream << "INSERT INTO \" << MOVIE_TABLE_NAME << "\" VALUE {'"
            << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
            << INFO_KEY << "': ?}";

request.SetStatement(sqlStream.str());

// Create the parameter attributes.
Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
ratingAttribute->SetN(rating);
infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
attributes.push_back(infoMapAttribute);
request.SetParameters(attributes);

Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
    << std::endl;
}

```

```
        return false;
    }
}

std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}
```

```

    }
  }
}

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
    dynamoClient.ExecuteStatement(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

// 5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;

```

```

std::stringstream sqlStream;
sqlStream << "SELECT * FROM \" << MOVIE_TABLE_NAME << "\" WHERE \"
    << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?";

request.SetStatement(sqlStream.str());

Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
request.SetParameters(attributes);

Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to retrieve the movie information: "
        << outcome.GetError().GetMessage() << std::endl;
    return false;
}
else {
    const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

    const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

    if (items.size() == 1) {
        printMovieInfo(items[0]);
    }
    else {
        std::cerr << "Error: " << items.size() << " movies were retrieved. "
            << " There should be only one movie." << std::endl;
    }
}
}

std::cout << "Deleting the movie" << std::endl;

// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \" << MOVIE_TABLE_NAME << "\" WHERE \"
        << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?";

```



```

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movie: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

std::cout << "Movie successfully deleted." << std::endl;
return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
 \sa createMoviesDynamoDBTable()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;

```

```

yearAttributeDefinition.SetAttributeName(TITLE_KEY);
yearAttributeDefinition.SetAttributeType(
    Aws::DynamoDB::Model::ScalarAttributeType::S);
request.AddAttributeDefinitions(yearAttributeDefinition);

Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorType() ==
        Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
            << result.GetError().GetMessage();
        return false;
    }
}
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
}

```

```

        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
            << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
    \sa deleteMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.

```

```

    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

Amazon EC2 examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon EC2.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon EC2

The following code examples show how to get started using Amazon EC2.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
project("hello_ec2")

# Set the C++ standard to use to build this target.
```

```

# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_ec2.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

```

```
/*
 * A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 * Cloud (Amazon EC2) client and describes
 * the Amazon EC2 instances.
 *
 * main function
 *
 * Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            auto outcome = ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                        std::setw(48) << "Name" <<
                        std::setw(20) << "ID" <<
                        std::setw(25) << "Ami" <<
                        std::setw(15) << "Type" <<
                        std::setw(15) << "State" <<
                        std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }

                const std::vector<Aws::EC2::Model::Reservation> &reservations =
                    outcome.GetResult().GetReservations();
            }
        }
    }
}
```

```

        for (const auto &reservation: reservations) {
            const std::vector<Aws::EC2::Model::Instance> &instances =
                reservation.GetInstances();
            for (const auto &instance: instances) {
                Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                    instance.GetState().GetName());

                Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                    instance.GetInstanceType());

                Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                    instance.GetMonitoring().GetState());
                Aws::String name = "Unknown";

                const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                    [](const Aws::EC2::Model::Tag
&tag) {
                            return tag.GetKey() ==
"Name";
                        });
                if (nameIter != tags.cend()) {
                    name = nameIter->GetValue();
                }
                std::cout <<
                    std::setw(48) << name <<
                    std::setw(20) << instance.GetInstanceId() <<
                    std::setw(25) << instance.GetImageId() <<
                    std::setw(15) << typeString <<
                    std::setw(15) << instanceStateString <<
                    std::setw(15) << monitorString << std::endl;
            }
        }

        if (!outcome.GetResult().GetNextToken().empty()) {
            request.SetNextToken(outcome.GetResult().GetNextToken());
        } else {

```



```
        done = true;
    }
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = 1;
    break;
}
}
}

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

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)

Actions

AllocateAddress

The following code example shows how to use `AllocateAddress`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::AllocateAddressRequest request;
request.SetDomain(Aws::EC2::Model::DomainType::vpc);
```

```
const Aws::EC2::Model::AllocateAddressOutcome outcome =
    ec2Client.AllocateAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to allocate Elastic IP address:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

allocationId = outcome.GetResult().GetAllocationId();
```

- For API details, see [AllocateAddress](#) in *AWS SDK for C++ API Reference*.

AssociateAddress

The following code example shows how to use AssociateAddress.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::AssociateAddressRequest associate_request;
associate_request.SetInstanceId(instanceId);
associate_request.SetAllocationId(allocationId);

const Aws::EC2::Model::AssociateAddressOutcome associate_outcome =
    ec2Client.AssociateAddress(associate_request);
if (!associate_outcome.IsSuccess()) {
    std::cerr << "Failed to associate Elastic IP address " << allocationId
        << " with instance " << instanceId << ":" <<
        associate_outcome.GetError().GetMessage() << std::endl;
    return false;
}
```

```
std::cout << "Successfully associated Elastic IP address " << allocationId
          << " with instance " << instanceId << std::endl;
```

- For API details, see [AssociateAddress](#) in *AWS SDK for C++ API Reference*.

AuthorizeSecurityGroupIngress

The following code example shows how to use `AuthorizeSecurityGroupIngress`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::IpRange ip_range;
ip_range.SetCidrIp("0.0.0.0/0");

Aws::EC2::Model::IpPermission permission1;
permission1.SetIpProtocol("tcp");
permission1.SetToPort(80);
permission1.SetFromPort(80);
permission1.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission1);

Aws::EC2::Model::IpPermission permission2;
permission2.SetIpProtocol("tcp");
permission2.SetToPort(22);
permission2.SetFromPort(22);
permission2.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission2);

const Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome authorizeOutcome =
    ec2Client.AuthorizeSecurityGroupIngress(authorizeRequest);
```

```

if (!authorizeOutcome.IsSuccess()) {
    std::cerr << "Failed to set ingress policy for security group " <<
        groupName << ":" << authorizeOutcome.GetError().GetMessage() <<
        std::endl;
    return false;
}

std::cout << "Successfully added ingress policy to security group " <<
    groupName << std::endl;

```

- For API details, see [AuthorizeSecurityGroupIngress](#) in *AWS SDK for C++ API Reference*.

CreateKeyPair

The following code example shows how to use `CreateKeyPair`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::CreateKeyPairRequest request;
request.SetKeyName(keyPairName);

Aws::EC2::Model::CreateKeyPairOutcome outcome =
ec2Client.CreateKeyPair(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create key pair:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created key pair named " <<
        keyPairName << std::endl;
}

```

- For API details, see [CreateKeyPair](#) in *AWS SDK for C++ API Reference*.

CreateSecurityGroup

The following code example shows how to use `CreateSecurityGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::CreateSecurityGroupRequest request;

request.SetGroupName(groupName);
request.SetDescription(description);
request.SetVpcId(vpcID);

const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
    ec2Client.CreateSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create security group:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully created security group named " << groupName <<
    std::endl;
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for C++ API Reference*.

CreateTags

The following code example shows how to use `CreateTags`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::Tag nameTag;
nameTag.SetKey("Name");
nameTag.SetValue(instanceName);

Aws::EC2::Model::CreateTagsRequest createRequest;
createRequest.AddResources(instanceID);
createRequest.AddTags(nameTag);

Aws::EC2::Model::CreateTagsOutcome createOutcome = ec2Client.CreateTags(
    createRequest);
if (!createOutcome.IsSuccess()) {
    std::cerr << "Failed to tag ec2 instance " << instanceID <<
        " with name " << instanceName << ":" <<
        createOutcome.GetError().GetMessage() << std::endl;
    return false;
}
```

- For API details, see [CreateTags](#) in *AWS SDK for C++ API Reference*.

DeleteKeyPair

The following code example shows how to use DeleteKeyPair.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteKeyPairRequest request;

request.SetKeyName(keyPairName);
const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete key pair " << keyPairName <<
        ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted key pair named " << keyPairName <<
        std::endl;
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for C++ API Reference*.

DeleteSecurityGroup

The following code example shows how to use DeleteSecurityGroup.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteSecurityGroupRequest request;

request.SetGroupId(securityGroupID);
auto outcome = ec2Client.DeleteSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete security group " << securityGroupID <<
        ":" << outcome.GetError().GetMessage() << std::endl;
}
```

```
else {
    std::cout << "Successfully deleted security group " << securityGroupID <<
        std::endl;
}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for C++ API Reference*.

DescribeAddresses

The following code example shows how to use DescribeAddresses.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeAddressesRequest request;
auto outcome = ec2Client.DescribeAddresses(request);
if (outcome.IsSuccess()) {
    std::cout << std::left << std::setw(20) << "InstanceId" <<
        std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
        std::setw(30) << "Allocation ID" << std::setw(25) <<
        "NIC ID" << std::endl;

    const auto &addresses = outcome.GetResult().GetAddresses();
    for (const auto &address: addresses) {
        Aws::String domainString =
            Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                address.GetDomain());

        std::cout << std::left << std::setw(20) <<
            address.GetInstanceId() << std::setw(15) <<
            address.GetPublicIp() << std::setw(10) << domainString <<
            std::setw(30) << address.GetAllocationId() << std::setw(25)
            << address.GetNetworkInterfaceId() << std::endl;
    }
}
```



```
else {
    std::cerr << "Failed to describe Elastic IP addresses:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

- For API details, see [DescribeAddresses](#) in *AWS SDK for C++ API Reference*.

DescribeAvailabilityZones

The following code example shows how to use `DescribeAvailabilityZones`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::DescribeAvailabilityZonesRequest describe_request;
auto describe_outcome = ec2Client.DescribeAvailabilityZones(describe_request);

if (describe_outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "ZoneName" <<
        std::setw(20) << "State" <<
        std::setw(32) << "Region" << std::endl;

    const auto &zones =
        describe_outcome.GetResult().GetAvailabilityZones();

    for (const auto &zone: zones) {
        Aws::String stateString =
            Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                zone.GetState());
        std::cout << std::left <<
            std::setw(32) << zone.GetZoneName() <<
            std::setw(20) << stateString <<
```

```
        std::setw(32) << zone.GetRegionName() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe availability zones:" <<
        describe_outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

- For API details, see [DescribeAvailabilityZones](#) in *AWS SDK for C++ API Reference*.

DescribeInstances

The following code example shows how to use DescribeInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeInstancesRequest request;
bool header = false;
bool done = false;
while (!done) {
    auto outcome = ec2Client.DescribeInstances(request);
    if (outcome.IsSuccess()) {
        if (!header) {
            std::cout << std::left <<
                std::setw(48) << "Name" <<
                std::setw(20) << "ID" <<
                std::setw(25) << "Ami" <<
                std::setw(15) << "Type" <<
                std::setw(15) << "State" <<
                std::setw(15) << "Monitoring" << std::endl;
            header = true;
        }
    }
}
```

```

const std::vector<Aws::EC2::Model::Reservation> &reservations =
    outcome.GetResult().GetReservations();

for (const auto &reservation: reservations) {
    const std::vector<Aws::EC2::Model::Instance> &instances =
        reservation.GetInstances();
    for (const auto &instance: instances) {
        Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
            instance.GetState().GetName());

        Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
            instance.GetInstanceType());

        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
            instance.GetMonitoring().GetState());
        Aws::String name = "Unknown";

        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
            [](const Aws::EC2::Model::Tag &tag)
{
            return tag.GetKey() ==

            "Name";

        });
        if (nameIter != tags.cend()) {
            name = nameIter->GetValue();
        }
        std::cout <<
            std::setw(48) << name <<
            std::setw(20) << instance.GetInstanceId() <<
            std::setw(25) << instance.GetImageId() <<
            std::setw(15) << typeString <<
            std::setw(15) << instanceStateString <<
            std::setw(15) << monitorString << std::endl;
    }
}

```

```

        if (!outcome.GetResult().GetNextToken().empty()) {
            request.SetNextToken(outcome.GetResult().GetNextToken());
        }
        else {
            done = true;
        }
    }
    else {
        std::cerr << "Failed to describe EC2 instances:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

- For API details, see [DescribeInstances](#) in *AWS SDK for C++ API Reference*.

DescribeKeyPairs

The following code example shows how to use DescribeKeyPairs.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeKeyPairsRequest request;

auto outcome = ec2Client.DescribeKeyPairs(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "Name" <<
        std::setw(64) << "Fingerprint" << std::endl;

    const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
        outcome.GetResult().GetKeyPairs();
    for (const auto &key_pair: key_pairs) {
        std::cout << std::left <<

```

```

        std::setw(32) << key_pair.GetKeyName() <<
        std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe key pairs:" <<
        outcome.GetError().GetMessage() << std::endl;
}

```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for C++ API Reference*.

DescribeRegions

The following code example shows how to use DescribeRegions.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::DescribeRegionsRequest request;
auto outcome = ec2Client.DescribeRegions(request);
bool result = true;
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "RegionName" <<
        std::setw(64) << "Endpoint" << std::endl;

    const auto &regions = outcome.GetResult().GetRegions();
    for (const auto &region: regions) {
        std::cout << std::left <<
            std::setw(32) << region.GetRegionName() <<
            std::setw(64) << region.GetEndpoint() << std::endl;
    }
}
else {

```

```
std::cerr << "Failed to describe regions:" <<
    outcome.GetError().GetMessage() << std::endl;
result = false;
}
```

- For API details, see [DescribeRegions](#) in *AWS SDK for C++ API Reference*.

DescribeSecurityGroups

The following code example shows how to use DescribeSecurityGroups.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeSecurityGroupsRequest request;

if (!groupID.empty()) {
    request.AddGroupIds(groupID);
}

Aws::String nextToken;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    auto outcome = ec2Client.DescribeSecurityGroups(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(30) << "GroupId" <<
            std::setw(30) << "VpcId" <<
            std::setw(64) << "Description" << std::endl;

        const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
```

```
        outcome.GetResult().GetSecurityGroups();

    for (const auto &securityGroup: securityGroups) {
        std::cout << std::left <<
            std::setw(32) << securityGroup.GetGroupName() <<
            std::setw(30) << securityGroup.GetGroupId() <<
            std::setw(30) << securityGroup.GetVpcId() <<
            std::setw(64) << securityGroup.GetDescription() <<
            std::endl;
    }
}
else {
    std::cerr << "Failed to describe security groups:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for C++ API Reference*.

MonitorInstances

The following code example shows how to use MonitorInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::MonitorInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.MonitorInstances(request);
```

```
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to enable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    }
    else if (dry_run_outcome.GetError().GetErrorType()
        != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
            instanceId << ": " << dry_run_outcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    request.SetDryRun(false);
    auto monitorInstancesOutcome = ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
            instanceId << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully enabled monitoring on instance " <<
            instanceId << std::endl;
    }
}
```

- For API details, see [MonitorInstances](#) in *AWS SDK for C++ API Reference*.

RebootInstances

The following code example shows how to use `RebootInstances`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```

Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::RebootInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.RebootInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to reboot on instance. A dry run should trigger
an error."
        <<
        std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to reboot instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
auto outcome = ec2Client.RebootInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to reboot instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully rebooted instance " << instanceId <<
        std::endl;
}
}

```

- For API details, see [RebootInstances](#) in *AWS SDK for C++ API Reference*.

ReleaseAddress

The following code example shows how to use `ReleaseAddress`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2(clientConfiguration);

Aws::EC2::Model::ReleaseAddressRequest request;
request.SetAllocationId(allocationID);

auto outcome = ec2.ReleaseAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to release Elastic IP address " <<
        allocationID << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully released Elastic IP address " <<
        allocationID << std::endl;
}
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for C++ API Reference*.

RunInstances

The following code example shows how to use RunInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
```

```
Aws::EC2::Model::RunInstancesRequest runRequest;
runRequest.SetImageId(amiId);
runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
runRequest.SetMinCount(1);
runRequest.SetMaxCount(1);

Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
    runRequest);
if (!runOutcome.IsSuccess()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();
if (instances.empty()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

instanceID = instances[0].GetInstanceId();
```

- For API details, see [RunInstances](#) in *AWS SDK for C++ API Reference*.

StartInstances

The following code example shows how to use StartInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::StartInstancesRequest start_request;
start_request.AddInstanceIds(instanceId);
start_request.SetDryRun(true);

auto dry_run_outcome = ec2Client.StartInstances(start_request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to start instance. A dry run should trigger an
error."
        << std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to start instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

start_request.SetDryRun(false);
auto start_instancesOutcome = ec2Client.StartInstances(start_request);

if (!start_instancesOutcome.IsSuccess()) {
    std::cout << "Failed to start instance " << instanceId << ": " <<
        start_instancesOutcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully started instance " << instanceId <<
        std::endl;
}
}
```

- For API details, see [StartInstances](#) in *AWS SDK for C++ API Reference*.

StopInstances

The following code example shows how to use StopInstances.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::StopInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.StopInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to stop instance. A dry run should trigger an
error."
        << std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to stop instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
auto outcome = ec2Client.StopInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to stop instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully stopped instance " << instanceId <<
        std::endl;
}
}
```

- For API details, see [StopInstances](#) in *AWS SDK for C++ API Reference*.

TerminateInstances

The following code example shows how to use `TerminateInstances`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::TerminateInstancesRequest request;
request.SetInstanceIds({instanceID});

Aws::EC2::Model::TerminateInstancesOutcome outcome =
    ec2Client.TerminateInstances(request);
if (outcome.IsSuccess()) {
    std::cout << "Ec2 instance '" << instanceID <<
        "' was terminated." << std::endl;
}
else {
    std::cerr << "Failed to terminate ec2 instance " << instanceID <<
        ", " <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
```

- For API details, see [TerminatelInstances](#) in *AWS SDK for C++ API Reference*.

UnmonitorInstances

The following code example shows how to use `UnmonitorInstances`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
unrequest.AddInstanceIds(instanceId);
unrequest.SetDryRun(true);

auto undryRunOutcome = ec2Client.UnmonitorInstances(unrequest);
if (undryRunOutcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
        <<
        std::endl;
    return false;
}
else if (undryRunOutcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to disable monitoring on instance " <<
        instanceId << ": " << undryRunOutcome.GetError().GetMessage() <<
        std::endl;
    return false;
}

unrequest.SetDryRun(false);
auto unmonitorInstancesOutcome = ec2Client.UnmonitorInstances(unrequest);
if (!unmonitorInstancesOutcome.IsSuccess()) {
    std::cout << "Failed to disable monitoring on instance " << instanceId
        << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully disable monitoring on instance " <<
        instanceId << std::endl;
}
}
```

- For API details, see [UnmonitorInstances](#) in *AWS SDK for C++ API Reference*.

EventBridge examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with EventBridge.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

PutEvents

The following code example shows how to use PutEvents.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
```



```
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Send the event.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}
```

- For API details, see [PutEvents](#) in *AWS SDK for C++ API Reference*.

PutRule

The following code example shows how to use PutRule.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>
```

Create the rule.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;
request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}
```

- For API details, see [PutRule](#) in *AWS SDK for C++ API Reference*.

PutTargets

The following code example shows how to use PutTargets.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Add the target.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);
target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
              << rule_name << ": " <<
              putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
        "Successfully created CloudWatch events target for rule "
        << rule_name << std::endl;
}
```

```
}
```

- For API details, see [PutTargets](#) in *AWS SDK for C++ API Reference*.

AWS Glue examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS Glue.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS Glue

The following code examples show how to get started using AWS Glue.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)
```

```

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the `hello_glue.cpp` source file.

```
#include <aws/core/Aws.h>
```

```
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 * A "Hello Glue" starter application which initializes an AWS Glue client and
 * lists the
 * AWS Glue job definitions.
 *
 * main function
 *
 * Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = glueClient.ListJobs(
                listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());
            }
        } while (listRunsOutcome.IsSuccess() && !listJobsRequest.IsLastPage());
    }
}
```

```
        nextToken = listRunsOutcome.GetResult().GetNextToken();
    } else {
        std::cerr << "Error listing jobs. "
                  << listRunsOutcome.GetError().GetMessage()
                  << std::endl;
        result = 1;
        break;
    }
} while (!nextToken.empty());

std::cout << "Your account has " << jobs.size() << " jobs."
          << std::endl;
for (size_t i = 0; i < jobs.size(); ++i) {
    std::cout << "    " << i + 1 << ". " << jobs[i] << std::endl;
}
}
Aws::ShutdownAPI(options); // Should only be called once.
return result;
}
```

- For API details, see [ListJobs](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateCrawler

The following code example shows how to use `CreateCrawler`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    Aws::Glue::GlueClient client(clientConfig);

    Aws::Glue::Model::S3Target s3Target;
    s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
    Aws::Glue::Model::CrawlerTargets crawlerTargets;
    crawlerTargets.AddS3Targets(s3Target);

    Aws::Glue::Model::CreateCrawlerRequest request;
    request.SetTargets(crawlerTargets);
    request.SetName(CRAWLER_NAME);
    request.SetDatabaseName(CRAWLER_DATABASE_NAME);
    request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
    request.SetRole(roleArn);

    Aws::Glue::Model::CreateCrawlerOutcome outcome =
    client.CreateCrawler(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created the crawler." << std::endl;
    }
    else {
        std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
        << std::endl;
        deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
        return false;
    }
}

```

- For API details, see [CreateCrawler](#) in *AWS SDK for C++ API Reference*.

CreateJob

The following code example shows how to use CreateJob.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::CreateJobRequest request;
request.SetName(JOB_NAME);
request.SetRole(roleArn);
request.SetGlueVersion(GLUE_VERSION);

Aws::Glue::Model::JobCommand command;
command.SetName(JOB_COMMAND_NAME);
command.SetPythonVersion(JOB_PYTHON_VERSION);
command.SetScriptLocation(
    Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
request.SetCommand(command);

Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created the job." << std::endl;
}
else {
    std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
        << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
```

- For API details, see [CreateJob](#) in *AWS SDK for C++ API Reference*.

DeleteCrawler

The following code example shows how to use DeleteCrawler.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::DeleteCrawlerRequest request;
request.SetName(crawler);

Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted the crawler." << std::endl;
}
else {
    std::cerr << "Error deleting the crawler. "
              << outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for C++ API Reference*.

DeleteDatabase

The following code example shows how to use DeleteDatabase.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::DeleteDatabaseRequest request;
request.SetName(database);

Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
    request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted the database." << std::endl;
}
else {
    std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
        << std::endl;
    result = false;
}
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for C++ API Reference*.

DeleteJob

The following code example shows how to use DeleteJob.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::DeleteJobRequest request;
request.SetJobName(job);

Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted the job." << std::endl;
}
else {
    std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
        << std::endl;
    result = false;
}
```

- For API details, see [DeleteJob](#) in *AWS SDK for C++ API Reference*.

GetCrawler

The following code example shows how to use `GetCrawler`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetCrawlerRequest request;
request.SetName(CRAWLER_NAME);

Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

if (outcome.IsSuccess()) {
    Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
    std::cout << "Retrieved crawler with state " <<
        Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
            crawlerState)
        << "." << std::endl;
}
else {
    std::cerr << "Error retrieving a crawler. "
        << outcome.GetError().GetMessage() << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
```

- For API details, see [GetCrawler](#) in *AWS SDK for C++ API Reference*.

GetDatabase

The following code example shows how to use GetDatabase.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetDatabaseRequest request;
request.SetName(CRAWLER_DATABASE_NAME);

Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

if (outcome.IsSuccess()) {
    const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

    std::cout << "Successfully retrieve the database\n" <<
        database.Jsonize().View().WriteReadable() << "." <<
std::endl;
}
else {
    std::cerr << "Error getting the database. "
        << outcome.GetError().GetMessage() << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for C++ API Reference*.

GetJobRun

The following code example shows how to use GetJobRun.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetJobRunRequest jobRunRequest;
jobRunRequest.SetJobName(jobName);
jobRunRequest.SetRunId(jobRunID);

Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
    jobRunRequest);

if (jobRunOutcome.IsSuccess()) {
    std::cout << "Displaying the job run JSON description." << std::endl;
    std::cout
        <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
        << std::endl;
}
else {
    std::cerr << "Error get a job run. "
        << jobRunOutcome.GetError().GetMessage()
        << std::endl;
}
```

- For API details, see [GetJobRun](#) in *AWS SDK for C++ API Reference*.

GetJobRuns

The following code example shows how to use GetJobRuns.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
getJobRunsRequest.SetJobName(jobName);

Aws::String nextToken; // Used for pagination.
std::vector<Aws::Glue::Model::JobRun> allJobRuns;
do {
    if (!nextToken.empty()) {
        getJobRunsRequest.SetNextToken(nextToken);
    }
    Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
        getJobRunsRequest);

    if (jobRunsOutcome.IsSuccess()) {
        const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
        allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

        nextToken = jobRunsOutcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error getting job runs. "
            << jobRunsOutcome.GetError().GetMessage()
            << std::endl;
        break;
    }
}
```



```
    }  
    } while (!nextToken.empty());
```

- For API details, see [GetJobRuns](#) in *AWS SDK for C++ API Reference*.

GetTables

The following code example shows how to use `GetTables`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
    // Optional: Set to the AWS Region in which the bucket was created  
(overrides config file).  
    // clientConfig.region = "us-east-1";  
  
Aws::Glue::GlueClient client(clientConfig);  
  
Aws::Glue::Model::GetTablesRequest request;  
request.SetDatabaseName(CRAWLER_DATABASE_NAME);  
std::vector<Aws::Glue::Model::Table> all_tables;  
Aws::String nextToken; // Used for pagination.  
do {  
    Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);  
  
    if (outcome.IsSuccess()) {  
        const std::vector<Aws::Glue::Model::Table> &tables =  
outcome.GetResult().GetTableList();  
        all_tables.insert(all_tables.end(), tables.begin(), tables.end());  
        nextToken = outcome.GetResult().GetNextToken();  
    }  
    else {  
        std::cerr << "Error getting the tables. "  
        << outcome.GetError().GetMessage()  
        << std::endl;
```

```

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                    clientConfig);
        return false;
    }
} while (!nextToken.empty());

std::cout << "The database contains " << all_tables.size()
          << (all_tables.size() == 1 ?
            " table." : "tables.") << std::endl;
std::cout << "Here is a list of the tables in the database.";
for (size_t index = 0; index < all_tables.size(); ++index) {
    std::cout << "    " << index + 1 << ": " << all_tables[index].GetName()
              << std::endl;
}

if (!all_tables.empty()) {
    int tableIndex = askQuestionForIntRange(
        "Enter an index to display the database detail ",
        1, static_cast<int>(all_tables.size()));
    std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
              << std::endl;

    tableName = all_tables[tableIndex - 1].GetName();
}

```

- For API details, see [GetTables](#) in *AWS SDK for C++ API Reference*.

ListJobs

The following code example shows how to use ListJobs.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

```
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::ListJobsRequest listJobsRequest;

Aws::String nextToken;
std::vector<Aws::String> allJobNames;

do {
    if (!nextToken.empty()) {
        listJobsRequest.SetNextToken(nextToken);
    }
    Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
        listJobsRequest);

    if (listRunsOutcome.IsSuccess()) {
        const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
        allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
        nextToken = listRunsOutcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error listing jobs. "
            << listRunsOutcome.GetError().GetMessage()
            << std::endl;
    }
} while (!nextToken.empty());
```

- For API details, see [ListJobs](#) in *AWS SDK for C++ API Reference*.

StartCrawler

The following code example shows how to use StartCrawler.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::StartCrawlerRequest request;
request.SetName(CRAWLER_NAME);

Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);

if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                           outcome.GetError().GetErrorType())) {
    if (!outcome.IsSuccess()) {
        std::cout << "Crawler was already started." << std::endl;
    }
    else {
        std::cout << "Successfully started crawler." << std::endl;
    }

    std::cout << "This may take a while to run." << std::endl;

    Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
    int iterations = 0;
    while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++iterations;
        if ((iterations % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Crawler status " <<
```

```

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
    crawlerState)
    << ". After " << iterations
    << " seconds elapsed."
    << std::endl;
}
Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
getCrawlerRequest.SetName(CRAWLER_NAME);

Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
    getCrawlerRequest);

if (getCrawlerOutcome.IsSuccess()) {
    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
}
else {
    std::cerr << "Error getting crawler.  "
    << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
    break;
}

if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
    std::cout << "Crawler finished running after " << iterations
    << " seconds."
    << std::endl;
}
}
else {
    std::cerr << "Error starting a crawler.  "
    << outcome.GetError().GetMessage()
    << std::endl;

    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
}

```

- For API details, see [StartCrawler](#) in *AWS SDK for C++ API Reference*.

StartJobRun

The following code example shows how to use StartJobRun.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::StartJobRunRequest request;
request.SetJobName(JOB_NAME);

Aws::Map<Aws::String, Aws::String> arguments;
arguments["--input_database"] = CRAWLER_DATABASE_NAME;
arguments["--input_table"] = tableName;
arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
request.SetArguments(arguments);

Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully started the job." << std::endl;

    Aws::String jobRunId = outcome.GetResult().GetJobRunId();

    int iterator = 0;
    bool done = false;
    while (!done) {
        ++iterator;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(JOB_NAME);
        jobRunRequest.SetRunId(jobRunId);
```

```

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
            jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
            Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

            if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                std::cerr << "Error running job. "
                    << jobRun.GetErrorMessage()
                    << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                    bucketName,
                    clientConfig);
                return false;
            }
            else if (jobRunState ==
                Aws::Glue::Model::JobRunState::SUCCEEDED) {
                std::cout << "Job run succeeded after " << iterator <<
                    " seconds elapsed." << std::endl;
                done = true;
            }
            else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                jobRunState) <<
                    ". " << iterator <<
                    " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error retrieving job run state. "
                << jobRunOutcome.GetError().GetMessage()
                << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                bucketName, clientConfig);
            return false;
        }
    }
}

```

```
    }  
  }  
  else {  
    std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()  
              << std::endl;  
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,  
                 clientConfig);  
    return false;  
  }  
}
```

- For API details, see [StartJobRun](#) in *AWS SDK for C++ API Reference*.

Scenarios

Get started with crawlers and jobs

The following code example shows how to:

- Create a crawler that crawls a public Amazon S3 bucket and generates a database of CSV-formatted metadata.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create a job to extract CSV data from the S3 bucket, transform the data, and load JSON-formatted output into another S3 bucket.
- List information about job runs, view transformed data, and clean up resources.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Scenario which demonstrates using AWS Glue to add a crawler and run a job.  
/*!  
\\sa runGettingStartedWithGlueScenario()
```



```

\param bucketName: An S3 bucket created in the setup.
\param roleName: An AWS Identity and Access Management (IAM) role created in the
setup.
\param clientConfig: AWS client configuration.
\return bool: Successful completion.
*/

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String &bucketName,
                                                    const Aws::String &roleName,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
                                      clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
    }
}

```

```
request.SetDatabaseName(CRAWLER_DATABASE_NAME);
request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
request.SetRole(roleArn);

Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created the crawler." << std::endl;
}
else {
    std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
    << std::endl;
    deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
    return false;
}
}

// 3. Get a crawler.
{
    Aws::Glue::Model::GetCrawlerRequest request;
    request.SetName(CRAWLER_NAME);

    Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

    if (outcome.IsSuccess()) {
        Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
        std::cout << "Retrieved crawler with state " <<
            Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                crawlerState)
            << "." << std::endl;
    }
    else {
        std::cerr << "Error retrieving a crawler. "
            << outcome.GetError().GetMessage() << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 4. Start a crawler.
```

```

{
    Aws::Glue::Model::StartCrawlerRequest request;
    request.SetName(CRAWLER_NAME);

    Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);

    if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                outcome.GetError().GetErrorType())) {
        if (!outcome.IsSuccess()) {
            std::cout << "Crawler was already started." << std::endl;
        }
        else {
            std::cout << "Successfully started crawler." << std::endl;
        }

        std::cout << "This may take a while to run." << std::endl;

        Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
        int iterations = 0;
        while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++iterations;
            if ((iterations % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                crawlerState)
                << ". After " << iterations
                << " seconds elapsed."
                << std::endl;
            }
            Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
            getCrawlerRequest.SetName(CRAWLER_NAME);

            Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                                getCrawlerRequest);

            if (getCrawlerOutcome.IsSuccess()) {
                crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();

```

```

        }
        else {
            std::cerr << "Error getting crawler.  "
                << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;

            break;
        }
    }

    if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
        std::cout << "Crawler finished running after " << iterations
            << " seconds."
            << std::endl;
    }
}
else {
    std::cerr << "Error starting a crawler.  "
        << outcome.GetError().GetMessage()
        << std::endl;

    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
}

// 5. Get a database.
{
    Aws::Glue::Model::GetDatabaseRequest request;
    request.SetName(CRAWLER_DATABASE_NAME);

    Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

    if (outcome.IsSuccess()) {
        const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

        std::cout << "Successfully retrieve the database\n" <<
            database.Jsonize().View().WriteReadable() << "." <<
std::endl;
    }
    else {
        std::cerr << "Error getting the database.  "
            << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                    clientConfig);
        return false;
    }
}

// 6. Get tables.
Aws::String tableName;
{
    Aws::Glue::Model::GetTablesRequest request;
    request.SetDatabaseName(CRAWLER_DATABASE_NAME);
    std::vector<Aws::Glue::Model::Table> all_tables;
    Aws::String nextToken; // Used for pagination.
    do {
        Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

        if (outcome.IsSuccess()) {
            const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
            all_tables.insert(all_tables.end(), tables.begin(), tables.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error getting the tables. "
                << outcome.GetError().GetMessage()
                << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                clientConfig);
            return false;
        }
    } while (!nextToken.empty());

    std::cout << "The database contains " << all_tables.size()
        << (all_tables.size() == 1 ?
            " table." : " tables.") << std::endl;
    std::cout << "Here is a list of the tables in the database.";
    for (size_t index = 0; index < all_tables.size(); ++index) {
        std::cout << "    " << index + 1 << ": " << all_tables[index].GetName()
            << std::endl;
    }

    if (!all_tables.empty()) {
        int tableIndex = askQuestionForIntRange(
            "Enter an index to display the database detail ",

```

```
        1, static_cast<int>(all_tables.size()));
        std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
        << std::endl;

        tableName = all_tables[tableIndex - 1].GetName();
    }
}

// 7. Create a job.
{
    Aws::Glue::Model::CreateJobRequest request;
    request.SetName(JOB_NAME);
    request.SetRole(roleArn);
    request.SetGlueVersion(GLUE_VERSION);

    Aws::Glue::Model::JobCommand command;
    command.SetName(JOB_COMMAND_NAME);
    command.SetPythonVersion(JOB_PYTHON_VERSION);
    command.SetScriptLocation(
        Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
    request.SetCommand(command);

    Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created the job." << std::endl;
    }
    else {
        std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
        << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 8. Start a job run.
{
    Aws::Glue::Model::StartJobRunRequest request;
    request.SetJobName(JOB_NAME);

    Aws::Map<Aws::String, Aws::String> arguments;
    arguments["--input_database"] = CRAWLER_DATABASE_NAME;
```

```

arguments["--input_table"] = tableName;
arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
request.SetArguments(arguments);

Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully started the job." << std::endl;

    Aws::String jobRunId = outcome.GetResult().GetJobRunId();

    int iterator = 0;
    bool done = false;
    while (!done) {
        ++iterator;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(JOB_NAME);
        jobRunRequest.SetRunId(jobRunId);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
            jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
            Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

            if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                std::cerr << "Error running job. "
                    << jobRun.GetErrorMessage()
                    << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                    bucketName,
                    clientConfig);
                return false;
            }
            else if (jobRunState ==
                Aws::Glue::Model::JobRunState::SUCCEEDED) {
                std::cout << "Job run succeeded after " << iterator <<
                    " seconds elapsed." << std::endl;
            }
        }
    }
}

```

```

        done = true;
    }
    else if ((iterator % 10) == 0) { // Log status every 10 seconds.
        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
            jobRunState) <<
        ". " << iterator <<
        " seconds elapsed." << std::endl;
    }
}
else {
    std::cerr << "Error retrieving job run state. "
        << jobRunOutcome.GetError().GetMessage()
        << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
        bucketName, clientConfig);
    return false;
}
}
}
else {
    std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
        << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
        clientConfig);
    return false;
}
}

// 9. List the output data stored in the S3 bucket.
{
    Aws::S3::S3Client s3Client;
    Aws::S3::Model::ListObjectsV2Request request;
    request.SetBucket(bucketName);
    request.SetPrefix(OUTPUT_FILE_PREFIX);

    Aws::String continuationToken; // Used for pagination.
    std::vector<Aws::S3::Model::Object> allObjects;
    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = s3Client.ListObjectsV2(

```



```

        request);

    if (outcome.IsSuccess()) {
        const std::vector<Aws::S3::Model::Object> &objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetNextContinuationToken();
    }
    else {
        std::cerr << "Error listing objects. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        break;
    }
} while (!continuationToken.empty());

std::cout << "Data from your job is in " << allObjects.size() <<
          " files in the S3 bucket, " << bucketName << "." << std::endl;

for (size_t i = 0; i < allObjects.size(); ++i) {
    std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
              << std::endl;
}

int objectIndex = askQuestionForIntRange(
    std::string(
        "Enter the number of a block to download it and see the
first ") +
    std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
    " lines of JSON output in the block: ", 1,
    static_cast<int>(allObjects.size()));

Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

std::stringstream stringStream;
if (getObjectFromBucket(bucketName, objectKey, stringStream,
    clientConfig)) {
    for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; ++i) {
        std::string line;
        std::getline(stringStream, line);
        std::cout << "    " << line << std::endl;
    }
}
else {

```

```

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                    clientConfig);
        return false;
    }
}

// 10. List all the jobs.
Aws::String jobName;
{
    Aws::Glue::Model::ListJobsRequest listJobsRequest;

    Aws::String nextToken;
    std::vector<Aws::String> allJobNames;

    do {
        if (!nextToken.empty()) {
            listJobsRequest.SetNextToken(nextToken);
        }
        Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
            listJobsRequest);

        if (listRunsOutcome.IsSuccess()) {
            const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
            allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
            nextToken = listRunsOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing jobs. "
                << listRunsOutcome.GetError().GetMessage()
                << std::endl;
        }
    } while (!nextToken.empty());
    std::cout << "Your account has " << allJobNames.size() << " jobs."
        << std::endl;
    for (size_t i = 0; i < allJobNames.size(); ++i) {
        std::cout << "    " << i + 1 << ". " << allJobNames[i] << std::endl;
    }
    int jobIndex = askQuestionForIntRange(
        Aws::String("Enter a number between 1 and ") +
        std::to_string(allJobNames.size()) +
        " to see the list of runs for a job: ",
        1, static_cast<int>(allJobNames.size()));
}

```

```

    jobName = allJobNames[jobIndex - 1];
}

// 11. Get the job runs for a job.
Aws::String jobRunID;
if (!jobName.empty()) {
    Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
    getJobRunsRequest.SetJobName(jobName);

    Aws::String nextToken; // Used for pagination.
    std::vector<Aws::Glue::Model::JobRun> allJobRuns;
    do {
        if (!nextToken.empty()) {
            getJobRunsRequest.SetNextToken(nextToken);
        }
        Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
            getJobRunsRequest);

        if (jobRunsOutcome.IsSuccess()) {
            const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
            allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

            nextToken = jobRunsOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error getting job runs. "
                << jobRunsOutcome.GetError().GetMessage()
                << std::endl;

            break;
        }
    } while (!nextToken.empty());

    std::cout << "There are " << allJobRuns.size() << " runs in the job '"
        <<
        jobName << "'." << std::endl;

    for (size_t i = 0; i < allJobRuns.size(); ++i) {
        std::cout << "    " << i + 1 << ". " << allJobRuns[i].GetJobName()
            << std::endl;
    }

    int runIndex = askQuestionForIntRange(

```

```

        Aws::String("Enter a number between 1 and ") +
        std::to_string(allJobRuns.size()) +
        " to see details for a run: ",
        1, static_cast<int>(allJobRuns.size()));
    jobRunID = allJobRuns[runIndex - 1].GetId();
}

// 12. Get a single job run.
if (!jobRunID.empty()) {
    Aws::Glue::Model::GetJobRunRequest jobRunRequest;
    jobRunRequest.SetJobName(jobName);
    jobRunRequest.SetRunId(jobRunID);

    Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
        jobRunRequest);

    if (jobRunOutcome.IsSuccess()) {
        std::cout << "Displaying the job run JSON description." << std::endl;
        std::cout
            <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
            << std::endl;
    }
    else {
        std::cerr << "Error get a job run. "
            << jobRunOutcome.GetError().GetMessage()
            << std::endl;
    }
}

return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
    clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
  \\sa deleteAssets()
  \\param crawler: Name of an AWS Glue crawler.
  \\param database: The name of an AWS Glue database.
  \\param job: The name of an AWS Glue job.
  \\param bucketName: The name of an S3 bucket.
  \\param clientConfig: AWS client configuration.
  \\return bool: Successful completion.
*/

```

```
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
&database,
                                const Aws::String &job, const Aws::String
&bucketName,
                                const Aws::Client::ClientConfiguration
&clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;

    // 13. Delete a job.
    if (!job.empty()) {
        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }

    // 14. Delete a database.
    if (!database.empty()) {
        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}
```

```

    }
}

// 15. Delete a crawler.
if (!crawler.empty()) {
    Aws::Glue::Model::DeleteCrawlerRequest request;
    request.SetName(crawler);

    Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the crawler." << std::endl;
    }
    else {
        std::cerr << "Error deleting the crawler. "
            << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}

// 16. Delete the job script and run data from the S3 bucket.
result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                    clientConfig);

return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
  \sa uploadFile()
  \param bucketName: An S3 bucket created in the setup.
  \param filePath: The path of the file to upload.
  \param fileName The name for the uploaded file.
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
*/
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                        const Aws::String &filePath,
                        const Aws::String &fileName,
                        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;

```

```

request.SetBucket(bucketName);
request.SetKey(fileName);

std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
        filePath.c_str(),
        std::ios_base::in |
std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << filePath << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3_client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutObject: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Added object '" << filePath << "' to bucket '"
        << bucketName << "'." << std::endl;
}

return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

```

```
Aws::String continuationToken; // Used for pagination.
bool result = true;
do {
    if (!continuationToken.empty()) {
        listObjectsRequest.SetContinuationToken(continuationToken);
    }

    Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
client.ListObjectsV2(
    listObjectsRequest);

    if (listObjectsOutcome.IsSuccess()) {
        const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
        if (!objects.empty()) {
            Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
            deleteObjectsRequest.SetBucket(bucketName);

            std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
            for (const Aws::S3::Model::Object &object: objects) {
                objectIdentifiers.push_back(
                    Aws::S3::Model::ObjectIdentifier().WithKey(
                        object.GetKey()));
            }
            Aws::S3::Model::Delete objectsDelete;
            objectsDelete.SetObjects(objectIdentifiers);
            objectsDelete.SetQuiet(true);
            deleteObjectsRequest.SetDelete(objectsDelete);

            Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                client.DeleteObjects(deleteObjectsRequest);

            if (!deleteObjectsOutcome.IsSuccess()) {
                std::cerr << "Error deleting objects. " <<
                    deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;

                result = false;
                break;
            }
            else {
                std::cout << "Successfully deleted the objects." << std::endl;
            }
        }
    }
}
```



```

    }
    else {
        std::cout << "No objects to delete in '" << bucketName << "'."
            << std::endl;
    }

    continuationToken =
listObjectsOutcome.GetResult().GetNextContinuationToken();
    }
    else {
        std::cerr << "Error listing objects. "
            << listObjectsOutcome.GetError().GetMessage() << std::endl;
        result = false;
        break;
    }
} while (!continuationToken.empty());

return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
  \sa getObjectFromBucket()
  \param bucketName: The S3 bucket name.
  \param objectKey: The object's name.
  \param objectStream: A stream to receive the retrieved data.
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
  */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." << std::endl;
    }
}

```

```
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

HealthImaging examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with HealthImaging.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello HealthImaging

The following code examples show how to get started using HealthImaging.

SDK for C++

Code for the CMakeLists.txt CMake file.

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

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

# Set this project's name.
project("hello_health-imaging")

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

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

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

```

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_health_imaging.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the `hello_health_imaging.cpp` source file.

```

#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 * A "Hello HealthImaging" starter application which initializes an AWS
 * HealthImaging (HealthImaging) client
 * and lists the HealthImaging data stores in the current account.
 *
 * main function
 *
 * Usage: 'hello_health-imaging'
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

```

```

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;

        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome listDatastoresOutcome
=
                medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
&dataStoreSummaries =
                    listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                    dataStoreSummaries.cbegin(),
                    dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "ListDatastores error: "
                    << listDatastoresOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());
    }
}

```

```
std::cout << allDataStoreSummaries.size() << " HealthImaging data "
    << ((allDataStoreSummaries.size() == 1) ?
        "store was retrieved." : "stores were retrieved.") <<
std::endl;

for (auto const &dataStoreSummary: allDataStoreSummaries) {
    std::cout << " Datastore: " << dataStoreSummary.GetDatastoreName()
        << std::endl;
    std::cout << " Datastore ID: " << dataStoreSummary.GetDatastoreId()
        << std::endl;
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}
```

- For API details, see [ListDatastores](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Topics

- [Actions](#)
- [Scenarios](#)

Actions

DeleteImageSet

The following code example shows how to use DeleteImageSet.

SDK for C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
```

```

\param datastoreID: The HealthImaging data store ID.
\param imageSetID: The image set ID.
\param clientConfig: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::deleteImageSet(
    const Aws::String &dataStoreID, const Aws::String &imageSetID,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
client.DeleteImageSet(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
            << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data store
"
            << dataStoreID << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteImageSet](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

GetDICOMImportJob

The following code example shows how to use `GetDICOMImportJob`.

SDK for C++

```

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                             const Aws::String &importJobID,
                                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
client.GetDICOMImportJob(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}

```

- For API details, see [GetDICOMImportJob](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

GetImageFrame

The following code example shows how to use `GetImageFrame`.

SDK for C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
 \param datastoreID: The HealthImaging data store ID.
 \param imageSetID: The image set ID.
 \param frameID: The image frame ID.
 \param jphFile: File to store the downloaded frame.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                             const Aws::String &imageSetID,
                                             const Aws::String &frameID,
                                             const Aws::String &jphFile,
                                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome = client.GetImageFrame(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
outcome.GetError().GetMessage()
        << std::endl;
    }
}
```

```
    return outcome.IsSuccess();
}
```

- For API details, see [GetImageFrame](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

GetImageSetMetadata

The following code example shows how to use `GetImageSetMetadata`.

SDK for C++

Utility function to get image set metadata.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
 \param datastoreID: The HealthImaging data store ID.
 \param imageSetID: The HealthImaging image set ID.
 \param versionID: The HealthImaging image set version ID, ignored if empty.
 \param outputPath: The path where the metadata will be stored as gzipped json.
 \param clientConfig: Aws client configuration.
 \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
                                                  Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
```

```

    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
client.GetImageSetMetadata(
    request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

Get image set metadata without version.

```

    if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
"", outputFilePath, clientConfig))
    {
        std::cout << "Successfully retrieved image set metadata." << std::endl;
        std::cout << "Metadata stored in: " << outputFilePath << std::endl;
    }

```

Get image set metadata with version.

```

    if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
versionID, outputFilePath, clientConfig))
    {
        std::cout << "Successfully retrieved image set metadata." << std::endl;
        std::cout << "Metadata stored in: " << outputFilePath << std::endl;
    }

```

- For API details, see [GetImageSetMetadata](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

SearchImageSets

The following code example shows how to use SearchImageSets.

SDK for C++

The utility function for searching image sets.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param datastoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
                                              Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
                                              &imageSetResults,
                                              const Aws::Client::ClientConfiguration
                                              &clientConfig) {
  Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
  Aws::MedicalImaging::Model::SearchImageSetsRequest request;
  request.SetDatastoreId(dataStoreID);
  request.SetSearchCriteria(searchCriteria);

  Aws::String nextToken; // Used for paginated results.
  bool result = true;
  do {
    if (!nextToken.empty()) {
      request.SetNextToken(nextToken);
    }

    Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
    client.SearchImageSets(
```

```

        request);
    if (outcome.IsSuccess()) {
        for (auto &imageSetMetadataSummary:
outcome.GetResult().GetImageSetsMetadataSummaries()) {
            imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
        }

        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
} while (!nextToken.empty());

return result;
}

```

Use case #1: EQUAL operator.

```

    Aws::Vector<Aws::String> imageIDsForPatientID;
    Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
    Aws::Vector<Aws::MedicalImaging::Model::SearchFilter> patientIDSearchFilters
= {

    Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Operator
    .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patient
        });

    searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
    bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

searchCriteriaEqualsPatientID,

                                imageIDsForPatientID,
                                clientConfig);

    if (result) {
        std::cout << imageIDsForPatientID.size() << " image sets found for the
patient with ID '"
        << patientID << "'." << std::endl;
        for (auto &imageSetResult : imageIDsForPatientID) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;

```

```

    }
}

```

Use case #2: BETWEEN operator using DICOMStudyDate and DICOMStudyTime.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

    useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000")));

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

    useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
        .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeStrin
        "%m%d"))
        .WithDICOMStudyTime("000000.000"));

    Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
    useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

    useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
    useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

    Aws::Vector<Aws::String> usesCase2Results;
    result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                        useCase2SearchCriteria,
                                                        usesCase2Results,
                                                        clientConfig);

    if (result) {
        std::cout << usesCase2Results.size() << " image sets found for between
1999/01/01 and present."
                << std::endl;
        for (auto &imageSetResult : usesCase2Results) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;
        }
    }
}

```

Use case #3: BETWEEN operator using createdAt. Time studies were previously persisted.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;
    useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z", Aws::Utils::DateF

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;
    useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

    Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
    useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

    useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
    useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

    Aws::Vector<Aws::String> usesCase3Results;
    result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                    useCase3SearchCriteria,
                                                    usesCase3Results,
                                                    clientConfig);

    if (result) {
        std::cout << usesCase3Results.size() << " image sets found for created
between 2023/11/30 and present."
                << std::endl;
        for (auto &imageSetResult : usesCase3Results) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;
        }
    }
}

```

Use case #4: EQUAL operator on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response in ASC order on updatedAt field.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;
    useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z", Aws::Utils::DateF

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;
    useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

```

```

    Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
    useCase4SearchFilterBetween.SetValues({useCase4StartDate, useCase4EndDate});

    useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
    seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

    Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
    useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

    useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

    Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
    useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
    useCase4SearchFilterEqual});

    Aws::MedicalImaging::Model::Sort useCase4Sort;
    useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
    useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

    useCase4SearchCriteria.SetSort(useCase4Sort);

    Aws::Vector<Aws::String> usesCase4Results;
    result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                    useCase4SearchCriteria,
                                                    usesCase4Results,
                                                    clientConfig);

    if (result) {
        std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
        << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response
\n"
        << "in ASC order on updatedAt field." << std::endl;
        for (auto &imageSetResult : usesCase4Results) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;
        }
    }
}

```

- For API details, see [SearchImageSets](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

StartDICOMImportJob

The following code example shows how to use StartDICOMImportJob.

SDK for C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
    const Aws::String &dataStoreID, const Aws::String &inputBucketName,
    const Aws::String &inputDirectory, const Aws::String &outputBucketName,
    const Aws::String &outputDirectory, const Aws::String &roleArn,
    Aws::String &importJobId,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
"/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);
```

```
Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
    startDICOMImportJobRequest);

if (startDICOMImportJobOutcome.IsSuccess()) {
    importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
}
else {
    std::cerr << "Failed to start DICOM import job because "
        << startDICOMImportJobOutcome.GetError().GetMessage() <<
std::endl;
}

return startDICOMImportJobOutcome.IsSuccess();
}
```

- For API details, see [StartDICOMImportJob](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Scenarios

Get started with image sets and image frames

The following code example shows how to import DICOM files and download image frames in HealthImaging.

The implementation is structured as a workflow command-line application.

- Set up resources for a DICOM import.
- Import DICOM files into a data store.
- Retrieve the image set IDs for the import job.
- Retrieve the image frame IDs for the image sets.
- Download, decode and verify the image frames.

- Clean up resources.

SDK for C++

Create an AWS CloudFormation stack with the necessary resources.

```
Aws::String inputBucketName;
Aws::String outputBucketName;
Aws::String dataStoreId;
Aws::String roleArn;
Aws::String stackName;

if (askYesNoQuestion(
    "Would you like to let this workflow create the resources for you? (y/n)
")) {
    stackName = askQuestion(
        "Enter a name for the AWS CloudFormation stack to create. ");
    Aws::String dataStoreName = askQuestion(
        "Enter a name for the HealthImaging datastore to create. ");

    Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
        stackName,
        dataStoreName,
        clientConfiguration);

    if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
        outputBucketName,
            roleArn)) {
        return false;
    }

    std::cout << "The following resources have been created." << std::endl;
    std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
        << std::endl;
    std::cout << "An Amazon S3 input bucket named: " << inputBucketName << "."
        << std::endl;
    std::cout << "An Amazon S3 output bucket named: " << outputBucketName << "."
        << std::endl;
    std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
    askQuestion("Enter return to continue.", alwaysTrueTest);
}
else {
    std::cout << "You have chosen to use preexisting resources:" << std::endl;
```

```

        dataStoreId = askQuestion(
            "Enter the data store ID of the HealthImaging datastore you wish to
use: ");
        inputBucketName = askQuestion(
            "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
            "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
            "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }

```

Copy DICOM files to the Amazon S3 import bucket.

```

std::cout
    << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
    << "Commons (IDC) Collections." << std::endl;
std::cout << "Here is the link to their website." << std::endl;
std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
    << std::endl;
std::cout
    << "First one of the DICOM folders in the IDC collection must be copied
to your\n"
    "input S3 bucket."
    << std::endl;
std::cout << "You have the choice of one of the following "
    << IDC_ImageChoices.size() << " folders to copy." << std::endl;

int index = 1;
for (auto &idcChoice: IDC_ImageChoices) {
    std::cout << index << " - " << idcChoice.mDescription << std::endl;
    index++;
}
int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
Aws::String inputDirectory = "input";

```

```

std::cout << "The files in the directory '" << fromDirectory << "' in the bucket
'"
    << IDC_S3_BucketName << "' will be copied " << std::endl;
std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
    << "' in the bucket '" << inputBucketName << "'." << std::endl;
askQuestion("Enter return to start the copy.", alwaysTrueTest);

if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
    IDC_S3_BucketName,
    fromDirectory,
    inputBucketName,
    inputDirectory, clientConfiguration)) {
    std::cerr << "This workflow will exit because of an error." << std::endl;
    cleanup(stackName, dataStoreId, clientConfiguration);
    return false;
}

```

Import the DICOM files to the Amazon S3 data store.

```

bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                                const Aws::String &inputBucketName,
                                                const Aws::String &inputDirectory,
                                                const Aws::String &outputBucketName,
                                                const Aws::String &outputDirectory,
                                                const Aws::String &roleArn,
                                                Aws::String &importJobId,
                                                const
    Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = false;
    if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
        outputBucketName, outputDirectory, roleArn, importJobId,
        clientConfiguration)) {
        std::cout << "DICOM import job started with job ID " << importJobId << "."
            << std::endl;
        result = waitImportJobCompleted(dataStoreID, importJobId,
            clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;
        }
    }
}

```

```

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
 \param dataStoreID: The HealthImaging data store ID.
 \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
 \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
 \param outputBucketName: The name of the S3 bucket for the output.
 \param outputDirectory: The directory in the S3 bucket to store the output.
 \param roleArn: The ARN of the IAM role with permissions for the import.
 \param importJobId: A string to receive the import job ID.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
    const Aws::String &dataStoreID, const Aws::String &inputBucketName,
    const Aws::String &inputDirectory, const Aws::String &outputBucketName,
    const Aws::String &outputDirectory, const Aws::String &roleArn,
    Aws::String &importJobId,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
"/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
    startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
        << startDICOMImportJobOutcome.GetError().GetMessage() <<
std::endl;

```

```

    }

    return startDICOMImportJobOutcome.IsSuccess();
}

//! Routine which waits for a DICOM import job to complete.
/*!
 * @param datastoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String &datastoreID,
                                                    const Aws::String &importJobId,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
    Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
        getDicomImportJobOutcome = getDICOMImportJob(
            datastoreID, importJobId,
            clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
            getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

            std::cout << "DICOM import job status: " <<

            Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                jobStatus) << std::endl;
        }
        else {
            std::cerr << "Failed to get import job status because "
                << getDicomImportJobOutcome.GetError().GetMessage() <<
            std::endl;
            return false;
        }
    }
}

```

```

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
 \param datastoreID: The HealthImaging data store ID.
 \param importJobID: The DICOM import job ID
 \param clientConfig: Aws client configuration.
 \return GetDICOMImportJobOutcome: The import job outcome.
 */
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
client.GetDICOMImportJob(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}

```

Get image sets created by the DICOM import job.

```

bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
&datastoreID,
                                                         const Aws::String
&importJobId,
                                                         Aws::Vector<Aws::String>
&imageSets,
                                                         const
Aws::Client::ClientConfiguration &clientConfiguration) {

```



```

    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome =
getDICOMImportJob(
    datastoreID, importJobId, clientConfiguration);
    bool result = false;
    if (getDicomImportJobOutcome.IsSuccess()) {
        auto outputURI =
getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
        Aws::Http::URI uri(outputURI);
        const Aws::String &bucket = uri.GetAuthority();
        Aws::String key = uri.GetPath();

        Aws::S3::S3Client s3Client(clientConfiguration);
        Aws::S3::Model::GetObjectRequest objectRequest;
        objectRequest.SetBucket(bucket);
        objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

        auto getObjectOutcome = s3Client.GetObject(objectRequest);
        if (getObjectOutcome.IsSuccess()) {
            auto &data = getObjectOutcome.GetResult().GetBody();

            std::stringstream stringStream;
            stringStream << data.rdbuf();

            try {
                // Use JMESPath to extract the image set IDs.
                // https://jmespath.org/specification.html
                std::string jmesPathExpression =
"jobSummary.imageSetsSummary[].imageSetId";
                jsoncons::json doc = jsoncons::json::parse(stringStream.str());

                jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,
jmesPathExpression);\
                for (auto &imageSet: imageSetsJson.array_range()) {
                    imageSets.push_back(imageSet.as_string());
                }

                result = true;
            }
            catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }
        }
    }
}

```

```

        else {
            std::cerr << "Failed to get object because "
                << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
            << getDicomImportJobOutcome.GetError().GetMessage() << std::endl;
    }

    return result;
}

```

Get image frame information for image sets.

```

bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
&dataStoreID,
                                                    const Aws::String
&imageSetID,
                                                    const Aws::String
&outDirectory,
                                                    Aws::Vector<ImageFrameInfo>
&imageFrames,
                                                    const
Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID + "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for version
ID.
                                fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {
                    throw std::runtime_error("Failed to open file " + fileName);
                }

                std::stringstream stringStream;
                stringStream << inFileStream.rdbuf();
                metadataGZip = stringStream.str();
            }
        }
    }
}

```

```

    }
    std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                metadataGZip.size());

    // Use JMESPath to extract the image set IDs.
    // https://jmespath.org/specification.html
    jsoncons::json doc = jsoncons::json::parse(metadataJson);
    std::string jmesPathExpression = "Study.Series.*.Instances[].[*]";
    jsoncons::json instances = jsoncons::jmespath::search(doc,

jmesPathExpression);
    for (auto &instance: instances.array_range()) {
        jmesPathExpression = "DICOM.RescaleSlope";
        std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
        jmesPathExpression = "DICOM.RescaleIntercept";
        std::string rescaleIntercept = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

        jmesPathExpression = "ImageFrames[].[*]";
        jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);

        for (auto &imageFrame: imageFramesJson.array_range()) {
            ImageFrameInfo imageFrameIDs;
            imageFrameIDs.mImageSetId = imageSetID;
            imageFrameIDs.mImageFrameId = imageFrame.find(
                "ID")->value().as_string();
            imageFrameIDs.mRescaleIntercept = rescaleIntercept;
            imageFrameIDs.mRescaleSlope = rescaleSlope;
            imageFrameIDs.MinPixelValue = imageFrame.find(
                "MinPixelValue")->value().as_string();
            imageFrameIDs.MaxPixelValue = imageFrame.find(
                "MaxPixelValue")->value().as_string();

            jmesPathExpression =
"max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
            jsoncons::json checksumJson =
jsoncons::jmespath::search(imageFrame,

jmesPathExpression);

```

```

        imageFrameIDs.mFullResolutionChecksum =
checksumJson.as_integer<uint32_t>();

        imageFrames.emplace_back(imageFrameIDs);
    }
}

    result = true;
}
catch (const std::exception &e) {
    std::cerr << "getImageFramesForImageSet failed because " << e.what()
        << std::endl;
}
}

return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
 \param dataStoreID: The HealthImaging data store ID.
 \param imageSetID: The HealthImaging image set ID.
 \param versionID: The HealthImaging image set version ID, ignored if empty.
 \param outputPath: The path where the metadata will be stored as gzipped json.
 \param clientConfig: Aws client configuration.
 \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
client.GetImageSetMetadata(
    request);
    if (outcome.IsSuccess()) {

```

```

        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

Download, decode and verify image frames.

```

bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
    const Aws::String &dataStoreID,
    const Aws::Vector<ImageFrameInfo> &imageFrames,
    const Aws::String &outDirectory,
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
        clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
outDirectory](

```

```

        const Aws::MedicalImaging::MedicalImagingClient *client,
        const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
        Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext>
&context) {

    if (!handleGetImageFrameResult(outcome, outDirectory, imageFrame)) {
        std::cerr << "Failed to download and convert image frame: "
            << imageFrame.mImageFrameId << " from image set: "
            << imageFrame.mImageSetId << std::endl;
        result = false;
    }

    count--;
    if (count <= 0) {

        semaphore.ReleaseAll();
    }
}; // End of 'getImageFrameAsyncLambda' lambda.

medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                        getImageFrameAsyncLambda);
}

if (count > 0) {
    semaphore.WaitOne();
}

if (result) {
    std::cout << imageFrames.size() << " image files were downloaded."
        << std::endl;
}

return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
    const Aws::String &jphFile,
    uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }
}

```

```

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected value."
            << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
            OPJ_PATH_LEN);

        inFileStream = opj_stream_create_default_file_stream(
            decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                "Unable to create input file stream for file '" + jphFile +
                "'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }
    }
}

```

```
int decodeMessageLevel = 1;
if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
    std::cerr << "Failed to setup codec logging." << std::endl;
}

if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
    throw std::runtime_error("Failed to setup decompression codec.");
}
if (!opj_codec_set_threads(decompressorCodec, 4)) {
    throw std::runtime_error("Failed to set decompression codec threads.");
}

if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
    throw std::runtime_error("Failed to read header.");
}

if (!opj_decode(decompressorCodec, inFileStream,
                outputImage)) {
    throw std::runtime_error("Failed to decode.");
}

if (DEBUGGING) {
    std::cout << "image width : " << outputImage->x1 - outputImage->x0
              << std::endl;
    std::cout << "image height : " << outputImage->y1 - outputImage->y0
              << std::endl;
    std::cout << "number of channels: " << outputImage->numcomps
              << std::endl;
    std::cout << "colorspace : " << outputImage->color_space << std::endl;
}

} catch (const std::exception &e) {
    std::cerr << e.what() << std::endl;
    if (outputImage) {
        opj_image_destroy(outputImage);
        outputImage = nullptr;
    }
}
if (inFileStream) {
    opj_stream_destroy(inFileStream);
}
if (decompressorCodec) {
    opj_destroy_codec(decompressorCodec);
}
}
```



```

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved image
    bitmap and
    //! then verifies the checksum of the bitmap.
    /*!
    * @param image: The OpenJPEG image struct.
    * @param crc32Checksum: The CRC32 checksum.
    * @return bool: Function succeeded.
    */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                image->comps[channel].dx;
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
                uint32_t fromIndex = fromRowStart + col / image->comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
        buffer.size() * sizeof(myType));
}

```

```

    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                    << crc32Checksum << ", actual - " << crc32.checksum()
                    << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                       uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;

        if (signedData) {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,
                                                                    crc32Checksum);

                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,
                                                                    crc32Checksum);

                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,
                                                                    crc32Checksum);

                    break;
                default:
                    std::cerr
                        << "verifyChecksumForImage, unsupported data type,
signed bytes - "

```

```

        << bytes << std::endl;
        break;
    }
}
else {
    switch (bytes) {
        case 1 :
            result = verifyChecksumForImageForType<uint8_t>(image,
                                                            crc32Checksum);
            break;
        case 2 :
            result = verifyChecksumForImageForType<uint16_t>(image,
                                                            crc32Checksum);
            break;
        case 4 :
            result = verifyChecksumForImageForType<uint32_t>(image,
                                                            crc32Checksum);
            break;
        default:
            std::cerr
                << "verifyChecksumForImage, unsupported data type,
unsigned bytes - "
                << bytes << std::endl;
            break;
    }
}

if (!result) {
    std::cerr << "verifyChecksumForImage, error bytes " << bytes
              << " signed "
              << signedData << std::endl;
}
}
else {
    std::cerr << "'verifyChecksumForImage', no channels in the image."
              << std::endl;
}
return result;
}

```

Clean up resources.

```

bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                       const Aws::String &dataStoreId,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
        "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const Aws::Client::ClientConfiguration
&clientConfiguration) {


    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID, clientConfiguration);
        }
    }

    return result;
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [DeleteImageSet](#)
 - [GetDICOMImportJob](#)
 - [GetImageFrame](#)
 - [GetImageSetMetadata](#)

- [SearchImageSets](#)
- [StartDICOMImportJob](#)

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

IAM examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with IAM.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.


Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello IAM

The following code examples show how to get started using IAM.

SDK for C++

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

```
# Set the minimum required version of CMake for this project.
```

```
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
  hello_iam.cpp)

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

Code for the iam.cpp source file.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 * A "Hello IAM" starter application which initializes an AWS Identity and Access
 * Management (IAM) client
 * and lists the IAM policies.
 *
 * main function
 *
 * Usage: 'hello_iam'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }
        }
    }
}
```

```

    if (!header) {
        std::cout << std::left << std::setw(55) << "Name" <<
            std::setw(30) << "ID" << std::setw(80) << "Arn" <<
            std::setw(64) << "Description" << std::setw(12) <<
            "CreateDate" << std::endl;
        header = true;
    }

    const auto &policies = outcome.GetResult().GetPolicies();
    for (const auto &policy: policies) {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) << policy.GetArn()
<<
<<
            std::setw(64) << policy.GetDescription() << std::setw(12)
            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    } else {
        done = true;
    }
}

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

```

- For API details, see [ListPolicies](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

AttachRolePolicy

The following code example shows how to use `AttachRolePolicy`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                roleName << ": " << list_outcome.GetError().GetMessage() <<
                std::endl;
            return false;
        }
    }

    const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
    if (std::any_of(policies.cbegin(), policies.cend(),
        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
            return policy.GetPolicyArn() == policyArn;
        })) {
        std::cout << "Policy " << policyArn <<
            " is already attached to role " << roleName << std::endl;
        return true;
    }
}
```

```
        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
            roleName << ": " << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role " <<
            roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for C++ API Reference*.

CreateAccessKey

The following code example shows how to use CreateAccessKey.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
```

```

Aws::IAM::Model::CreateAccessKeyRequest request;
request.SetUserName(userName);

Aws::String result;
Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating access key for IAM user " << userName
              << ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    const auto &accessKey = outcome.GetResult().GetAccessKey();
    std::cout << "Successfully created access key for IAM user " <<
              userName << std::endl << "  aws_access_key_id = " <<
              accessKey.GetAccessKeyId() << std::endl <<
              "  aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
              std::endl;
    result = accessKey.GetAccessKeyId();
}

return result;
}

```

- For API details, see [CreateAccessKey](#) in *AWS SDK for C++ API Reference*.

CreateAccountAlias

The following code example shows how to use `CreateAccountAlias`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;

```

```

request.SetAccountAlias(aliasName);

Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating account alias " << aliasName << ": "
              << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created account alias " << aliasName <<
              << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for C++ API Reference*.

CreatePolicy

The following code example shows how to use CreatePolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                     const Aws::String &rsrcArn,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);

```

```

    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
            std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
        << "  \"Version\": \"2012-10-17\","
        << "  \"Statement\": ["
        << "    {"
        << "      \"Effect\": \"Allow\","
        << "      \"Action\": \"logs:CreateLogGroup\","
        << "      \"Resource\": \""
        << rsrc_arn
        << "\"\"
        << "    },"
        << "    {"
        << "      \"Effect\": \"Allow\","
        << "      \"Action\": ["
        << "        \"dynamodb:DeleteItem\","
        << "        \"dynamodb:GetItem\","
        << "        \"dynamodb:PutItem\","
        << "        \"dynamodb:Scan\","
        << "        \"dynamodb:UpdateItem\"
        << "      ],"
        << "      \"Resource\": \""
        << rsrc_arn
        << "\"\"
        << "    }"
        << "  ]"
        << "}";

    return stringStream.str();
}

```

- For API details, see [CreatePolicy](#) in *AWS SDK for C++ API Reference*.

CreateRole

The following code example shows how to use CreateRole.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::createIamRole(
    const Aws::String &roleName,
    const Aws::String &policy,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateRole](#) in *AWS SDK for C++ API Reference*.

CreateUser

The following code example shows how to use CreateUser.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
        create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

return create_outcome.IsSuccess();
```

- For API details, see [CreateUser](#) in *AWS SDK for C++ API Reference*.

DeleteAccessKey

The following code example shows how to use DeleteAccessKey.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for C++ API Reference*.

DeleteAccountAlias

The following code example shows how to use `DeleteAccountAlias`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for C++ API Reference*.

DeletePolicy

The following code example shows how to use DeletePolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for C++ API Reference*.

DeleteServerCertificate

The following code example shows how to use DeleteServerCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                         const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName <<
                ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
            << std::endl;
    }

    return result;
}
```

- For API details, see [DeleteServerCertificate](#) in *AWS SDK for C++ API Reference*.

DeleteUser

The following code example shows how to use DeleteUser.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
```

- For API details, see [DeleteUser](#) in *AWS SDK for C++ API Reference*.

DetachRolePolicy

The following code example shows how to use DetachRolePolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
detachRequest.SetRoleName(roleName);
detachRequest.SetPolicyArn(policyArn);

auto detachOutcome = iam.DetachRolePolicy(detachRequest);
if (!detachOutcome.IsSuccess()) {
    std::cerr << "Failed to detach policy " << policyArn << " from role "
              << roleName << ": " << detachOutcome.GetError().GetMessage() <<
              std::endl;
}
else {
    std::cout << "Successfully detached policy " << policyArn << " from role "
              << roleName << std::endl;
}

return detachOutcome.IsSuccess();
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for C++ API Reference*.

GetAccessKeyLastUsed

The following code example shows how to use `GetAccessKeyLastUsed`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);
```

```

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome = iam.GetAccessKeyLastUsed(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
            secretKeyID << ":" << outcome.GetError().GetMessage() <<
std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
            outcome.GetResult()
                .GetAccessKeyLastUsed()
                .GetLastUsedDate()
                .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
            lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for C++ API Reference*.

GetPolicy

The following code example shows how to use `GetPolicy`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);
}

```

```

auto outcome = iam.GetPolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error getting policy " << policyArn << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    const auto &policy = outcome.GetResult().GetPolicy();
    std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
        "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
        policy.GetArn() << std::endl << "Description: " <<
        policy.GetDescription() << std::endl << "CreateDate: " <<
        policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [GetPolicy](#) in *AWS SDK for C++ API Reference*.

GetServerCertificate

The following code example shows how to use `GetServerCertificate`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);
}

```

```
auto outcome = iam.GetServerCertificate(request);
bool result = true;
if (!outcome.IsSuccess()) {
    if (outcome.GetError().GetErrorType() !=
        Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
        std::cerr << "Error getting server certificate " << certificateName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Certificate '" << certificateName
            << "' not found." << std::endl;
    }
}
else {
    const auto &certificate = outcome.GetResult().GetServerCertificate();
    std::cout << "Name: " <<
        certificate.GetServerCertificateMetadata().GetServerCertificateName()
            << std::endl << "Body: " << certificate.GetCertificateBody() <<
                std::endl << "Chain: " << certificate.GetCertificateChain() <<
                    std::endl;
}

return result;
}
```

- For API details, see [GetServerCertificate](#) in *AWS SDK for C++ API Reference*.

ListAccessKeys

The following code example shows how to use ListAccessKeys.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```

bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                    key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                statusString << std::setw(20) <<
                key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
}

```

```
    }  
  }  
  
  return true;  
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for C++ API Reference*.

ListAccountAliases

The following code example shows how to use `ListAccountAliases`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool  
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::IAM::IAMClient iam(clientConfig);  
    Aws::IAM::Model::ListAccountAliasesRequest request;  
  
    bool done = false;  
    bool header = false;  
    while (!done) {  
        auto outcome = iam.ListAccountAliases(request);  
        if (!outcome.IsSuccess()) {  
            std::cerr << "Failed to list account aliases: " <<  
                outcome.GetError().GetMessage() << std::endl;  
            return false;  
        }  
  
        const auto &aliases = outcome.GetResult().GetAccountAliases();  
        if (!header) {  
            if (aliases.size() == 0) {  
                std::cout << "Account has no aliases" << std::endl;  
            }  
        }  
    }  
}
```

```
        break;
    }
    std::cout << std::left << std::setw(32) << "Alias" << std::endl;
    header = true;
}

for (const auto &alias: aliases) {
    std::cout << std::left << std::setw(32) << alias << std::endl;
}

if (outcome.GetResult().GetIsTruncated()) {
    request.SetMarker(outcome.GetResult().GetMarker());
}
else {
    done = true;
}
}

return true;
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for C++ API Reference*.

ListPolicies

The following code example shows how to use `ListPolicies`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration &clientConfig)
{
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;
```

```
bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListPolicies(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list iam policies: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (!header) {
        std::cout << std::left << std::setw(55) << "Name" <<
            std::setw(30) << "ID" << std::setw(80) << "Arn" <<
            std::setw(64) << "Description" << std::setw(12) <<
            "CreateDate" << std::endl;
        header = true;
    }

    const auto &policies = outcome.GetResult().GetPolicies();
    for (const auto &policy: policies) {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
            std::setw(64) << policy.GetDescription() << std::setw(12) <<
            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for C++ API Reference*.

ListServerCertificates

The following code example shows how to use ListServerCertificates.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listServerCertificates(
    const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(14) << "UploadDate" << std::setw(14) <<
                "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
            outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                certificate.GetServerCertificateName() << std::setw(30) <<
```

```
        certificate.GetServerCertificateId() << std::setw(80) <<
        certificate.GetArn() << std::setw(14) <<
        certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str())
<<
        std::setw(14) <<
        certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str())
<<
        std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListServerCertificates](#) in *AWS SDK for C++ API Reference*.

ListUsers

The following code example shows how to use ListUsers.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;
```

```
bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListUsers(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list iam users:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
            std::setw(30) << "ID" << std::setw(64) << "Arn" <<
            std::setw(20) << "CreateDate" << std::endl;
        header = true;
    }

    const auto &users = outcome.GetResult().GetUsers();
    for (const auto &user: users) {
        std::cout << std::left << std::setw(32) << user.GetUserName() <<
            std::setw(30) << user.GetUserId() << std::setw(64) <<
            user.GetArn() << std::setw(20) <<
            user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
            << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListUsers](#) in *AWS SDK for C++ API Reference*.

PutRolePolicy

The following code example shows how to use PutRolePolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::putRolePolicy(
    const Aws::String &roleName,
    const Aws::String &policyName,
    const Aws::String &policyDocument,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [PutRolePolicy](#) in *AWS SDK for C++ API Reference*.

UpdateAccessKey

The following code example shows how to use UpdateAccessKey.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for C++ API Reference*.

UpdateServerCertificate

The following code example shows how to use `UpdateServerCertificate`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String &currentCertificateName,
                                          const Aws::String &newCertificateName,
                                          const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                    currentCertificateName << " to " << newCertificateName << ":"
<<
                    outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                    << "' not found." << std::endl;
        }
    }
    return result;
}
```

- For API details, see [UpdateServerCertificate](#) in *AWS SDK for C++ API Reference*.

UpdateUser

The following code example shows how to use UpdateUser.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
            " successfully updated with new user name " << newUserName <<
            std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateUser](#) in *AWS SDK for C++ API Reference*.

Scenarios

Create a user and assume a role

The following code example shows how to create a user and assume a role.

Warning

To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as [AWS IAM Identity Center](#).

- Create a user with no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list S3 buckets using temporary credentials, then clean up resources.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
         \sa DeleteCreatedEntities
         \param client: IAM client.
         \param role: IAM role.
         \param user: IAM user.
         \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
```

```

        const Aws::IAM::Model::User &user,
        const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

/*! Scenario to create an IAM user, create an IAM role, and apply the role to the
    user.
    // "IAM access" permissions are needed to run this code.
    // "STS assume role" permissions are needed to run this code. (Note: It might be
    necessary to
    //   create a custom policy).
    /*!
    \sa iamCreateUserAssumeRoleScenario
    \param clientConfig: Aws client configuration.
    \return bool: Successful completion.
    */
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }
}

```

```
    }

    user = outcome.GetResult().GetUser();
}

// 2. Create a role.
{
    // Get the IAM user for the current client in order to access its ARN.
    Aws::String iamUserArn;
    {
        Aws::IAM::Model::GetUserRequest request;
        Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error getting Iam user. " <<
                outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully retrieved Iam user "
                << outcome.GetResult().GetUser().GetUserName()
                << std::endl;
        }

        iamUserArn = outcome.GetResult().GetUser().GetArn();
    }

    Aws::IAM::Model::CreateRoleRequest request;

    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleName = "iam-demo-role-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleName(roleName);

    // Build policy document for role.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");

    Aws::Utils::Document jsonPrincipal;
    jsonPrincipal.WithString("AWS", iamUserArn);
    jsonStatement.WithObject("Principal", jsonPrincipal);
    jsonStatement.WithString("Action", "sts:AssumeRole");
    jsonStatement.WithObject("Condition", Aws::Utils::Document());
}
```

```
Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Setting policy for role\n  "
           << policyDocument.View().WriteCompact() << std::endl;

// Set role policy document as JSON string.
request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
              outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a role with name " << roleName
              << std::endl;
}

role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3::*");
```

```

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array<Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    std::cout << "Creating a policy.\n " <<
policyDocument.View().WriteCompact()
        << std::endl;

    // Set IAM policy document as JSON string.
    request.SetPolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreatePolicyOutcome outcome = client.CreatePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully created a policy with name, " << policyName
<<
            "." << std::endl;
    }

    policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSCliient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +

    Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);

```



```

    Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

    // Repeatedly call AssumeRole, because there is often a delay
    // before the role is available to be assumed.
    // Repeat at most 20 times when access is denied.
    int count = 0;
    while (true) {
        assumeRoleOutcome = stsClient.AssumeRole(request);
        if (!assumeRoleOutcome.IsSuccess()) {
            if (count > 20 ||
                assumeRoleOutcome.GetError().GetErrorType() !=
                Aws::STS::STSErrors::ACCESS_DENIED) {
                std::cerr << "Error assuming role after 20 tries. " <<
                    assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            std::cout << "Successfully assumed the role after " << count
                << " seconds." << std::endl;
            break;
        }
        count++;
    }

    credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {

```

```
        if (listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout
                << "Access to list buckets denied because privileges have
not been applied."
                << std::endl;
        }
    }
    else {
        std::cerr
            << "Successfully retrieved bucket lists when this should not
happen."
            << std::endl;
    }
}

// 6. Attach the policy to the role.
{
    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(role.GetRoleName());
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully attached the policy with name, "
            << policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." << std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
```

```

// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the role.
// Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
while (true) {
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                   credentials.GetSecretAccessKey(),
                                   credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if ((count > LIST_BUCKETS_WAIT_SEC) ||
            listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    else {

        std::cout << "Successfully retrieved bucket lists after " << count
            << " seconds." << std::endl;

        break;
    }
    count++;
}

// 8. Delete all the created resources.
return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                       const Aws::IAM::Model::Role &role,
                                       const Aws::IAM::Model::User &user,
                                       const Aws::IAM::Model::Policy &policy) {

    bool result = true;
    if (policy.ArnHasBeenSet()) {

```

```
    // Detach the policy from the role.
    {
        Aws::IAM::Model::DetachRolePolicyRequest request;
        request.SetPolicyArn(policy.GetArn());
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
    request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error Detaching policy from roles. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully detached the policy with arn "
                << policy.GetArn()
                << " from role " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    // Delete the policy.
    {
        Aws::IAM::Model::DeletePolicyRequest request;
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting policy. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the policy with arn "
                << policy.GetArn() << std::endl;
        }
    }
}

if (role.RoleIdHasBeenSet()) {
    // Delete the role.
```

```
Aws::IAM::Model::DeleteRoleRequest request;
request.SetRoleName(role.GetRoleName());

Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting role. " <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
else {
    std::cout << "Successfully deleted the role with name "
        << role.GetRoleName() << std::endl;
}
}

if (user.ArnHasBeenSet()) {
    // Delete the user.
    Aws::IAM::Model::DeleteUserRequest request;
    request.WithUserName(user.GetUserName());

    Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting user. " <<
            outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Successfully deleted the user with name "
            << user.GetUserName() << std::endl;
    }
}

return result;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)

- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

AWS IoT examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS IoT.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS IoT

The following code examples show how to get started using AWS IoT.

SDK for C++

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)
```

```
# Set this project's name.
project("hello_iot")

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
  hello_iot.cpp)

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

Code for the `hello_iot.cpp` source file.

```
#include <aws/core/Aws.h>
```

```
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 * A "Hello IoT" starter application which initializes an AWS IoT client and
 * lists the AWS IoT topics in the current account.
 *
 * main function
 *
 * Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
            iotClient.ListThings(
                listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
            listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
        } while (listThingsOutcome.IsSuccess());
    }
}
```



```
    }
    else {
        std::cerr << "List things failed"
                << listThingsOutcome.GetError().GetMessage() << std::endl;
        break;
    }
} while (!nextToken.empty());

std::cout << allThings.size() << " thing(s) found." << std::endl;
for (auto const &thing: allThings) {
    std::cout << thing.GetThingName() << std::endl;
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}
```

- For API details, see [listThings](#) in *AWS SDK for C++ API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Topics

- [Actions](#)
- [Scenarios](#)

Actions

AttachThingPrincipal

The following code example shows how to use `AttachThingPrincipal`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Attach a principal to an AWS IoT thing.
/#!
 \param principal: A principal to attach.
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [AttachThingPrincipal](#) in *AWS SDK for C++ API Reference*.

CreateKeysAndCertificate

The following code example shows how to use CreateKeysAndCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create keys and certificate for an Aws IoT device.
#!/ This routine will save certificates and keys to an output folder, if provided.
/*!
 \param outputFolder: Location for storing output in files, ignored when string is
 empty.
 \param certificateARNResult: A string to receive the ARN of the created
 certificate.
 \param certificateID: A string to receive the ID of the created certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                         Aws::String &certificateARNResult,
                                         Aws::String &certificateID,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
        client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
ID: "
                << certificateID << std::endl;

        if (!outputFolder.empty()) {
```

```
        std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
            << "'." << std::endl;
        std::cout << "Be sure these files are stored securely." << std::endl;

        Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
        std::ofstream certificateFile(certificateFilePath);
        if (!certificateFile.is_open()) {
            std::cerr << "Error opening certificate file, '" <<
certificateFilePath
                << "'."
                << std::endl;
            return false;
        }
        certificateFile << outcome.GetResult().GetCertificatePem();
        certificateFile.close();

        const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

        Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
        std::ofstream privateKeyFile(privateKeyFilePath);
        if (!privateKeyFile.is_open()) {
            std::cerr << "Error opening private key file, '" <<
privateKeyFilePath
                << "'."
                << std::endl;
            return false;
        }
        privateKeyFile << keyPair.GetPrivateKey();
        privateKeyFile.close();

        Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
        std::ofstream publicKeyFile(publicKeyFilePath);
        if (!publicKeyFile.is_open()) {
            std::cerr << "Error opening public key file, '" << publicKeyFilePath
                << "'."
                << std::endl;
            return false;
        }
        publicKeyFile << keyPair.GetPublicKey();
    }
}
else {
```

```
        std::cerr << "Error creating keys and certificate: "  
                << outcome.GetError().GetMessage() << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- For API details, see [CreateKeysAndCertificate](#) in *AWS SDK for C++ API Reference*.

CreateThing

The following code example shows how to use CreateThing.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create an AWS IoT thing.  
/*!  
  \param thingName: The name for the thing.  
  \param clientConfiguration: AWS client configuration.  
  \return bool: Function succeeded.  
*/  
bool AwsDoc::IoT::createThing(const Aws::String &thingName,  
                             const Aws::Client::ClientConfiguration  
&clientConfiguration) {  
    Aws::IoT::IoTClient iotClient(clientConfiguration);  
    Aws::IoT::Model::CreateThingRequest createThingRequest;  
    createThingRequest.SetThingName(thingName);  
  
    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(  
        createThingRequest);  
    if (outcome.IsSuccess()) {  
        std::cout << "Successfully created thing " << thingName << std::endl;  
    }  
    else {
```

```

        std::cerr << "Failed to create thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [CreateThing](#) in *AWS SDK for C++ API Reference*.

CreateTopicRule

The following code example shows how to use `CreateTopicRule`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Create an AWS IoT rule with an SNS topic as the target.
 *!
 * \param ruleName: The name for the rule.
 * \param snsTopic: The SNS topic ARN for the action.
 * \param sql: The SQL statement used to query the topic.
 * \param roleARN: The IAM role ARN for the action.
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                            const Aws::String &snsTopicARN, const Aws::String &sql,
                            const Aws::String &roleARN,
                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);
}

```

```

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

```

- For API details, see [CreateTopicRule](#) in *AWS SDK for C++ API Reference*.

DeleteCertificate

The following code example shows how to use DeleteCertificate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Delete a certificate.
/*!
    \param certificateID: The ID of a certificate.

```

```
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteCertificate](#) in *AWS SDK for C++ API Reference*.

DeleteThing

The following code example shows how to use DeleteThing.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Delete an AWS IoT thing.
```



```

/#!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteThing](#) in *AWS SDK for C++ API Reference*.

DeleteTopicRule

The following code example shows how to use DeleteTopicRule.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Delete an AWS IoT rule.
/#!
 \param ruleName: The name for the rule.
 \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteTopicRule](#) in *AWS SDK for C++ API Reference*.

DescribeEndpoint

The following code example shows how to use DescribeEndpoint.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Describe the endpoint specific to the AWS account making the call.
    */
    \param endpointResult: String to receive the endpoint result.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.

```

```
*/
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
        describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DescribeEndpoint](#) in *AWS SDK for C++ API Reference*.

DescribeThing

The following code example shows how to use DescribeThing.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Describe an AWS IoT thing.
```

```

/ *!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
    iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing " << result.GetThingName() << " " <<
std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
        << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << " attribute: " << attribute.first << "=" <<
attribute.second
            << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DescribeThing](#) in *AWS SDK for C++ API Reference*.

DetachThingPrincipal

The following code example shows how to use `DetachThingPrincipal`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Detach a principal from an AWS IoT thing.
/#!
\param principal: A principal to detach.
\param thingName: The name for the thing.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing "
"
                << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                << thingName << ": "
                << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DetachThingPrincipal](#) in *AWS SDK for C++ API Reference*.

ListCertificates

The following code example shows how to use ListCertificates.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ List certificates registered in the AWS account making the call.
/#!/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
        iotClient.ListCertificates(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
        outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
```

```

        result.GetCertificates().begin(),
        result.GetCertificates().end());
    }
    else {
        std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} while (!marker.empty());

std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

for (auto &certificate: allCertificates) {
    std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
std::endl;
    std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
        << std::endl;
    std::cout << std::endl;
}

return true;
}

```

- For API details, see [ListCertificates](#) in *AWS SDK for C++ API Reference*.

SearchIndex

The following code example shows how to use SearchIndex.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Query the AWS IoT fleet index.
/*! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
    \param query: The query string.

```

```
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                    result.GetThings().cbegin(),
                                    result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << " Thing name: " << thingDocument.GetThingName() << "."
            << std::endl;
    }
    return true;
}
```


- For API details, see [SearchIndex](#) in *AWS SDK for C++ API Reference*.

UpdateIndexingConfiguration

The following code example shows how to use `UpdateIndexingConfiguration`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Update the indexing configuration.
/#!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
  ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration object
  which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::IoT::updateIndexingConfiguration(
    const Aws::IoT::Model::ThingIndexingConfiguration
&thingIndexingConfiguration,
    const Aws::IoT::Model::ThingGroupIndexingConfiguration
&thingGroupIndexingConfiguration,
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {
        request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }
}
```

```

    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
    iotClient.UpdateIndexingConfiguration(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [UpdateIndexingConfiguration](#) in *AWS SDK for C++ API Reference*.

UpdateThing

The following code example shows how to use UpdateThing.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Update an AWS IoT thing with attributes.
/*!
 \param thingName: The name for the thing.
 \param attributeMap: A map of key/value attributes/
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                             const std::map<Aws::String, Aws::String>
                             &attributeMap,

```

```
const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateThing](#) in *AWS SDK for C++ API Reference*.

Scenarios

Work with device management use cases

The following code example shows how to work with AWS IoT device management use cases using AWS IoT SDK

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an AWS IoT thing.

```

    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }

```

```

//! Create an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
        createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

Generate and attach a device certificate.

```

    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing? (y/
n) ")) {

```

```

    Aws::String outputFolder;
    if (askYesNoQuestion(
        "Would you like to save the certificate and keys to file? (y/n) ")
    {
        outputFolder = std::filesystem::current_path();
        outputFolder += "/device_keys_and_certificates";

        std::filesystem::create_directories(outputFolder);

        std::cout << "The certificate and keys will be saved to the folder: "
            << outputFolder << std::endl;
    }

    if (!createKeysAndCertificate(outputFolder, certificateARN, certificateID,
        clientConfiguration)) {
        std::cerr << "Exiting because createKeysAndCertificate failed."
            << std::endl;
        cleanup(thingName, "", "", "", "", false, clientConfiguration);
        return false;
    }

    std::cout << "\nNext, the certificate will be attached to the thing.\n"
        << std::endl;
    if (!attachThingPrincipal(certificateARN, thingName, clientConfiguration)) {
        std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "",
            false,
            clientConfiguration);
        return false;
    }
}

```

```

/*! Create keys and certificate for an Aws IoT device.
    /*! This routine will save certificates and keys to an output folder, if provided.
    /*!
    \param outputFolder: Location for storing output in files, ignored when string is
    empty.
    \param certificateARNResult: A string to receive the ARN of the created
    certificate.
    \param certificateID: A string to receive the ID of the created certificate.
    \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                          Aws::String &certificateARNResult,
                                          Aws::String &certificateID,
                                          const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
        client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
ID: "
                << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
                << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
certificateFilePath
                    << "'."
                    << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);

```



```

    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

Perform various operations on the AWS IoT thing.

```

    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
"v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String endpoint;
    if (!describeEndpoint(endpoint, clientConfiguration)) {
        std::cerr << "Exiting because getEndpoint failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
        return false;
    }
}

```



```
std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
printAsterisksLine();

std::cout << "Now the certificates in your account will be listed." <<
std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

if (!listCertificates(clientConfiguration)) {
    std::cerr << "Exiting because listCertificates failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
<< "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
<< "the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow." << std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

if (!updateThingShadow(thingName, R("{\"state\":{\"reported\":
{\"temperature\":25,\"humidity\":50}}}")", clientConfiguration)) {
    std::cerr << "Exiting because updateThingShadow failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now, the state information for the shadow will be retrieved.\n" <<
std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

Aws::String shadowState;
if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
    std::cerr << "Exiting because getThingShadow failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
```

```

        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to access
data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation template."
<< std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
clientConfiguration);
        return false;
    }

    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
false,
clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM ";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += """;

```

```

printAsterisksLine();

std::cout << "Now a rule will be created.\n" << std::endl;
std::cout << "Rules are an administrator-level action. Any user who has
permission\n"
        << "to create rules will be able to access data processed by the
rule." << std::endl;
std::cout << "In this case, the rule will use an SNS topic" << std::endl;
std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
std::endl;
std::cout << "For more information on IoT SQL, see https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" << std::endl;
Aws::String ruleName = askQuestion("Enter a rule name: ");
if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
clientConfiguration)) {
    std::cerr << "Exiting because createRule failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
            false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now your rules will be listed.\n" << std::endl;
askQuestion("Press Enter to continue: ", alwaysTrueTest);
if (!listTopicRules(clientConfiguration)) {
    std::cerr << "Exiting because listRules failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
            false,
            clientConfiguration);
    return false;
}

printAsterisksLine();
Aws::String queryString = "thingName:" + thingName;
std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
<< queryString << "'.\n" << std::endl;
std::cout << "For query information, see https://docs.aws.amazon.com/iot/latest/developerguide/query-syntax.html" << std::endl;

std::cout << "For this query to work, thing indexing must be enabled in your
account.\n"

```

```

    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
    << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/latest/
developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account? (y/
n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGISTR

thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnectiv
    // The ThingGroupIndexingConfiguration object is ignored if not set.
    Aws::IoT::Model::ThingGroupIndexingConfiguration
thingGroupIndexingConfiguration;
    if (!updateIndexingConfiguration(thingIndexingConfiguration,
thingGroupIndexingConfiguration, clientConfiguration)) {
        std::cerr << "Exiting because updateIndexingConfiguration failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME,
            ruleName, false,
            clientConfiguration);
        return false;
    }
}

if (!searchIndex(queryString, clientConfiguration)) {

    std::cerr << "Exiting because searchIndex failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
        false,
        clientConfiguration);
    return false;
}
}

```

```

//! Update an AWS IoT thing with attributes.
/*!
 \param thingName: The name for the thing.
 \param attributeMap: A map of key/value attributes/
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.

```

```

*/
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                             const std::map<Aws::String, Aws::String>
                             &attributeMap,
                             const Aws::Client::ClientConfiguration
                             &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
 \param endpointResult: String to receive the endpoint result.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
        describeEndpointRequest);

```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
        iotClient.ListCertificates(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
            outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                result.GetCertificates().begin(),
                result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
        }
    }
}
```

```

        return false;
    }
} while (!marker.empty());

std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

for (auto &certificate: allCertificates) {
    std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
std::endl;
    std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
        << std::endl;
    std::cout << std::endl;
}

return true;
}

///  

/*! Update the shadow of an AWS IoT thing.  

 *!  

 * \param thingName: The name for the thing.  

 * \param document: The state information, in JSON format.  

 * \param clientConfiguration: AWS client configuration.  

 * \return bool: Function succeeded.  

 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                     const Aws::String &document,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
std::make_shared<std::stringstream>(
        document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
iotDataPlaneClient.UpdateThingShadow(
        updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
            << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param documentResult: String to receive the state information, in JSON format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                Aws::String &documentResult,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
 \param ruleName: The name for the rule.
 \param snsTopic: The SNS topic ARN for the action.
 \param sql: The SQL statement used to query the topic.
 \param roleARN: The IAM role ARN for the action.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,

```



```

        const Aws::String &snsTopicARN, const Aws::String &sql,
        const Aws::String &roleARN,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;

```

```

    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome = iotClient.ListTopicRules(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
outcome.GetResult();
            allRules.insert(allRules.end(),
                result.GetRules().cbegin(),
                result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
        << std::endl;
    for (auto &rule: allRules) {
        std::cout << " Rule name: " << rule.GetRuleName() << ", rule ARN: "
            << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/developerguide/query-syntax.html
/*!
    \param query: The query string.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

```

```
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                    result.GetThings().cbegin(),
                                    result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << " Thing name: " << thingDocument.GetThingName() << "."
            << std::endl;
    }
    return true;
}
```

Clean up resources.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
&certificateARN,
                    const Aws::String &certificateID, const Aws::String &stackName,
                    const Aws::String &ruleName, bool askForConfirmation,
                    const Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the rule '" + ruleName +
            "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
        askYesNoQuestion(
            "Delete the CloudFormation stack '" +
stackName +
            "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the certificate '" +
            certificateARN + "'? (y/n) ")))
    {
        result &= detachThingPrincipal(certificateARN, thingName,
clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the thing '" + thingName +
            "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```

//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing "
"
                << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                << thingName << ": "
                << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

```

```

bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
    \param ruleName: The name for the rule.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
        ": " << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```
    }

    return outcome.IsSuccess();
}

///  
Delete an AWS IoT thing.  
/*!  
 \param thingName: The name for the thing.  
 \param clientConfiguration: AWS client configuration.  
 \return bool: Function succeeded.  
*/  
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,  
                               const Aws::Client::ClientConfiguration  
&clientConfiguration) {  
    Aws::IoT::IoTClient iotClient(clientConfiguration);  
    Aws::IoT::Model::DeleteThingRequest request;  
    request.SetThingName(thingName);  
    const auto outcome = iotClient.DeleteThing(request);  
    if (outcome.IsSuccess()) {  
        std::cout << "Successfully deleted thing " << thingName << std::endl;  
    }  
    else {  
        std::cerr << "Error deleting thing " << thingName << ": " <<  
            outcome.GetError().GetMessage() << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

AWS IoT data examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS IoT data.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

GetThingShadow

The following code example shows how to use `GetThingShadow`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Get the shadow of an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param documentResult: String to receive the state information, in JSON format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                Aws::String &documentResult,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
```



```

        std::cerr << "Error getting thing shadow: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetThingShadow](#) in *AWS SDK for C++ API Reference*.

UpdateThingShadow

The following code example shows how to use UpdateThingShadow.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Update the shadow of an AWS IoT thing.
 *!
 * \param thingName: The name for the thing.
 * \param document: The state information, in JSON format.
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                     const Aws::String &document,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
std::make_shared<std::stringstream>(
        document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
iotDataPlaneClient.UpdateThingShadow(

```

```
        updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateThingShadow](#) in *AWS SDK for C++ API Reference*.

Lambda examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Lambda.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Lambda

The following code examples show how to get started using Lambda.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

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

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

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

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

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

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

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
need to uncomment this
```

```

                                # and set the proper subdirectory to the
executables' location.

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

add_executable(${PROJECT_NAME}
    hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_lambda.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>

/*
 * A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
 * client and lists the Lambda functions.
 *
 * main function
 *
 * Usage: 'hello_lambda'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);

```

```

std::vector<Aws::String> functions;
Aws::String marker; // Used for pagination.

do {
    Aws::Lambda::Model::ListFunctionsRequest request;
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::Lambda::Model::ListFunctionsOutcome outcome =
lambdaClient.ListFunctions(
    request);

    if (outcome.IsSuccess()) {
        const Aws::Lambda::Model::ListFunctionsResult &listFunctionsResult =
outcome.GetResult();
        std::cout << listFunctionsResult.GetFunctions().size()
<< " lambda functions were retrieved." << std::endl;

        for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: listFunctionsResult.GetFunctions()) {
            functions.push_back(functionConfiguration.GetFunctionName());
            std::cout << functions.size() << " "
<< functionConfiguration.GetDescription() <<
std::endl;

            std::cout << " "
<<
Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                functionConfiguration.GetRuntime()) << ": "
<< functionConfiguration.GetHandler()
<< std::endl;

        }
        marker = listFunctionsResult.GetNextMarker();
    } else {
        std::cerr << "Error with Lambda::ListFunctions. "
<< outcome.GetError().GetMessage()
<< std::endl;
        result = 1;
        break;
    }
} while (!marker.empty());
}

```

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

- For API details, see [ListFunctions](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateFunction

The following code example shows how to use CreateFunction.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
// Optional: Set to the AWS Region in which the bucket was created  
(overrides config file).  
// clientConfig.region = "us-east-1";  
  
Aws::Lambda::LambdaClient client(clientConfig);  
  
Aws::Lambda::Model::CreateFunctionRequest request;  
request.SetFunctionName(LAMBDA_NAME);  
request.SetDescription(LAMBDA_DESCRIPTION); // Optional.  
#if USE_CPP_LAMBDA_FUNCTION  
request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);  
request.SetTimeout(15);  
request.SetMemorySize(128);
```

```

        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
    #if defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
    #elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
    #else
    #error "Unimplemented architecture"
    #endif // defined(architecture)
    #else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_8);
    #endif

    request.SetRole(roleArn);
    request.SetHandler(LAMBDA_HANDLER_NAME);
    request.SetPublish(true);
    Aws::Lambda::Model::FunctionCode code;
    std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                          std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;
    }

    #if USE_CPP_LAMBDA_FUNCTION
        std::cerr
            << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
            << std::endl;
    #endif

    deleteIamRole(clientConfig);
    return false;
}

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();

    code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
buffer.str().c_str(),
                                          buffer.str().length()));
    request.SetCode(code);

    Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
        request);

    if (outcome.IsSuccess()) {

```

```
        std::cout << "The lambda function was successfully created. " << seconds
                << " seconds elapsed." << std::endl;
        break;
    }

    else {
        std::cerr << "Error with CreateFunction. "
                << outcome.GetError().GetMessage()
                << std::endl;
        deleteIamRole(clientConfig);
        return false;
    }
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for C++ API Reference*.

DeleteFunction

The following code example shows how to use DeleteFunction.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Lambda::LambdaClient client(clientConfig);

Aws::Lambda::Model::DeleteFunctionRequest request;
request.SetFunctionName(LAMBDA_NAME);

Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
    request);

if (outcome.IsSuccess()) {
```



```
        std::cout << "The lambda function was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for C++ API Reference*.

GetFunction

The following code example shows how to use `GetFunction`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Lambda::LambdaClient client(clientConfig);

Aws::Lambda::Model::GetFunctionRequest request;
request.SetFunctionName(functionName);

Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

if (outcome.IsSuccess()) {
    std::cout << "Function retrieve.\n" <<
outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
    << std::endl;
}
```

```
else {
    std::cerr << "Error with Lambda::GetFunction. "
               << outcome.GetError().GetMessage()
               << std::endl;
}
```

- For API details, see [GetFunction](#) in *AWS SDK for C++ API Reference*.

Invoke

The following code example shows how to use Invoke.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Lambda::LambdaClient client(clientConfig);

Aws::Lambda::Model::InvokeRequest request;
request.SetFunctionName(LAMBDA_NAME);
request.SetLogType(logType);
std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
    "FunctionTest");
*payload << jsonPayload.View().WriteReadable();
request.SetBody(payload);
request.SetContentType("application/json");
Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

if (outcome.IsSuccess()) {
    invokeResult = std::move(outcome.GetResult());
    result = true;
    break;
}
```

```
    }

    else {
        std::cerr << "Error with Lambda::InvokeRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        break;
    }
}
```

- For API details, see [Invoke](#) in *AWS SDK for C++ API Reference*.

ListFunctions

The following code example shows how to use `ListFunctions`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Lambda::LambdaClient client(clientConfig);

std::vector<Aws::String> functions;
Aws::String marker;

do {
    Aws::Lambda::Model::ListFunctionsRequest request;
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
```

```
        request);

    if (outcome.IsSuccess()) {
        const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
        std::cout << result.GetFunctions().size()
            << " lambda functions were retrieved." << std::endl;

        for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
            functions.push_back(functionConfiguration.GetFunctionName());
            std::cout << functions.size() << " "
                << functionConfiguration.GetDescription() << std::endl;
            std::cout << " "
                << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                    functionConfiguration.GetRuntime()) << ": "
                << functionConfiguration.GetHandler()
                << std::endl;
        }
        marker = result.GetNextMarker();
    }
    else {
        std::cerr << "Error with Lambda::ListFunctions. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
} while (!marker.empty());
```

- For API details, see [ListFunctions](#) in *AWS SDK for C++ API Reference*.

UpdateFunctionCode

The following code example shows how to use UpdateFunctionCode.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;
    }

#ifdef USE_CPP_LAMBDA_FUNCTION
    std::cerr
        << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
        << std::endl;
#endif

    deleteLambdaFunction(client);
    deleteIamRole(clientConfig);
    return false;
}

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();
    request.SetZipFile(
        Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                buffer.str().length()));
    request.SetPublish(true);

    Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda code was successfully updated." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionCode. "
        << outcome.GetError().GetMessage()

```

```
        << std::endl;
    }
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for C++ API Reference*.

UpdateFunctionConfiguration

The following code example shows how to use UpdateFunctionConfiguration.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::Lambda::LambdaClient client(clientConfig);

Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
request.SetFunctionName(LAMBDA_NAME);
Aws::Lambda::Model::Environment environment;
environment.AddVariables("LOG_LEVEL", "DEBUG");
request.SetEnvironment(environment);

Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
    request);

if (outcome.IsSuccess()) {
    std::cout << "The lambda configuration was successfully updated."
        << std::endl;
    break;
}

else {
```

```
        std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "  
                << outcome.GetError().GetMessage()  
                << std::endl;  
    }
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for C++ API Reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.
- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see [Create a Lambda function with the console](#).

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///  
/*!  
 \param clientConfig: AWS client configuration.  
 \return bool: Successful completion.  
 */  
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(  
    const Aws::Client::ClientConfiguration &clientConfig) {  
  
    Aws::Lambda::LambdaClient client(clientConfig);
```

```
// 1. Create an AWS Identity and Access Management (IAM) role for Lambda
function.
Aws::String roleArn;
if (!getIamRoleArn(roleArn, clientConfig)) {
    return false;
}

// 2. Create a Lambda function.
int seconds = 0;
do {
    Aws::Lambda::Model::CreateFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#ifdef USE_CPP_LAMBDA_FUNCTION
    request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
    request.SetTimeout(15);
    request.SetMemorySize(128);

    // Assume the AWS Lambda function was built in Docker with same architecture
    // as this code.
#ifdef __x86_64__
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
    request.SetRuntime(Aws::Lambda::Model::Runtime::python3_8);
#endif

    request.SetRole(roleArn);
    request.SetHandler(LAMBDA_HANDLER_NAME);
    request.SetPublish(true);
    Aws::Lambda::Model::FunctionCode code;
    std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;
    }

#ifdef USE_CPP_LAMBDA_FUNCTION
    std::cerr
```



```

        << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
        << std::endl;
#endif
    deleteIamRole(clientConfig);
    return false;
}

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();

    code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
buffer.str().c_str(),
                                           buffer.str().length()));
    request.SetCode(code);

    Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully created. " << seconds
        << " seconds elapsed." << std::endl;
        break;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&
        outcome.GetError().GetMessage().find("role") >= 0) {
        if ((seconds % 5) == 0) { // Log status every 10 seconds.
            std::cout
                << "Waiting for the IAM role to become available as a
CreateFunction parameter. "
                << seconds
                << " seconds elapsed." << std::endl;

            std::cout << outcome.GetError().GetMessage() << std::endl;
        }
    }
    else {
        std::cerr << "Error with CreateFunction. "
        << outcome.GetError().GetMessage()
        << std::endl;
        deleteIamRole(clientConfig);
        return false;
    }
}

```

```

        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (60 > seconds);

    std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

// 3. Invoke the Lambda function.
{
    int increment = askQuestionForInt("Enter an increment integer: ");

    Aws::Lambda::Model::InvokeResult invokeResult;
    Aws::Utils::Json::JsonValue jsonPayload;
    jsonPayload.WithString("action", "increment");
    jsonPayload.WithInteger("number", increment);
    if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
        invokeResult, client)) {
        Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> values =
            jsonValue.View().GetAllObjects();
        auto iter = values.find("result");
        if (iter != values.end() && iter->second.IsIntegerType()) {
            {
                std::cout << INCREMENT_RESULT_PREFIX
                    << iter->second.AsInteger() << std::endl;
            }
        }
        else {
            std::cout << "There was an error in execution. Here is the log."
                << std::endl;
            Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                invokeResult.GetLogResult());
            std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
        }
    }
}

std::cout
    << "The Lambda function will now be updated with new code. Press return
to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

```

```
// 4. Update the Lambda function code.
{
    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;

#ifdef USE_CPP_LAMBDA_FUNCTION
        std::cerr
            << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
            << std::endl;
#endif

        deleteLambdaFunction(client);
        deleteIamRole(clientConfig);
        return false;
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();
    request.SetZipFile(
        Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                buffer.str().length()));

    request.SetPublish(true);

    Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda code was successfully updated." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionCode. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

std::cout
```

```
    << "This function uses an environment variable to control the logging
level."
    << std::endl;
    std::cout
    << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
    << std::endl;
    seconds = 0;

// 5. Update the Lambda function configuration.
do {
    ++seconds;
    std::this_thread::sleep_for(std::chrono::seconds(1));
    Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    Aws::Lambda::Model::Environment environment;
    environment.AddVariables("LOG_LEVEL", "DEBUG");
    request.SetEnvironment(environment);

    Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda configuration was successfully updated."
        << std::endl;
        break;
    }

    // RESOURCE_IN_USE: function code update not completed.
    else if (outcome.GetError().GetErrorType() !=
        Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
        if ((seconds % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Lambda function update in progress . After " <<
seconds
            << " seconds elapsed." << std::endl;
        }
    }
} else {
    std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
    << outcome.GetError().GetMessage()
    << std::endl;
}
```

```

} while (0 < seconds);

if (0 > seconds) {
    std::cerr << "Function failed to become active." << std::endl;
}
else {
    std::cout << "Updated function active after " << seconds << " seconds."
        << std::endl;
}

std::cout
    << "\nThe new code applies an arithmetic operator to two variables, x an
y."
    << std::endl;
std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-by"};
for (size_t i = 0; i < operators.size(); ++i) {
    std::cout << "    " << i + 1 << "    " << operators[i] << std::endl;
}

// 6. Invoke the updated Lambda function.
do {
    int operatorIndex = askQuestionForIntRange("Select an operator index 1 - 4
", 1,
                                             4);

    int x = askQuestionForInt("Enter an integer for the x value ");
    int y = askQuestionForInt("Enter an integer for the y value ");

    Aws::Utils::Json::JsonValue calculateJsonPayload;
    calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
    calculateJsonPayload.WithInteger("x", x);
    calculateJsonPayload.WithInteger("y", y);
    Aws::Lambda::Model::InvokeResult calculatedResult;
    if (invokeLambdaFunction(calculateJsonPayload,
                            Aws::Lambda::Model::LogType::Tail,
                            calculatedResult, client)) {
        Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> values =
            jsonValue.View().GetAllObjects();
        auto iter = values.find("result");
        if (iter != values.end() && iter->second.IsIntegerType()) {
            std::cout << ARITHMETIC_RESULT_PREFIX << x << " "
                << operators[operatorIndex - 1] << " "
                << y << " is " << iter->second.AsInteger() << std::endl;
        }
    }
}

```

```

        else if (iter != values.end() && iter->second.IsFloatingPointType()) {
            std::cout << ARITHMETIC_RESULT_PREFIX << x << " "
                << operators[operatorIndex - 1] << " "
                << y << " is " << iter->second.AsDouble() << std::endl;
        }
        else {
            std::cout << "There was an error in execution. Here is the log."
                << std::endl;
            Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                calculatedResult.GetLogResult());
            std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
        }
    }
}

    answer = askQuestion("Would you like to try another operation? (y/n) ");
} while (answer == "y");

std::cout
    << "A list of the lambda functions will be retrieved. Press return to
continue, ";
std::getline(std::cin, answer);

// 7. List the Lambda functions.

std::vector<Aws::String> functions;
Aws::String marker;

do {
    Aws::Lambda::Model::ListFunctionsRequest request;
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
        request);

    if (outcome.IsSuccess()) {
        const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
        std::cout << result.GetFunctions().size()
            << " lambda functions were retrieved." << std::endl;
    }
}

```

```

        for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
            functions.push_back(functionConfiguration.GetFunctionName());
            std::cout << functions.size() << " "
                << functionConfiguration.GetDescription() << std::endl;
            std::cout << " "
                << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                    functionConfiguration.GetRuntime()) << ": "
                << functionConfiguration.GetHandler()
                << std::endl;
        }
        marker = result.GetNextMarker();
    }
    else {
        std::cerr << "Error with Lambda::ListFunctions. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
} while (!marker.empty());

// 8. Get a Lambda function.
if (!functions.empty()) {
    std::stringstream question;
    question << "Choose a function to retrieve between 1 and " <<
functions.size()
        << " ";
    int functionIndex = askQuestionForIntRange(question.str(), 1,
static_cast<int>(functions.size()));

    Aws::String functionName = functions[functionIndex - 1];

    Aws::Lambda::Model::GetFunctionRequest request;
    request.SetFunctionName(functionName);

    Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

    if (outcome.IsSuccess()) {
        std::cout << "Function retrieve.\n" <<
outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
            << std::endl;
    }
}

```

```

        else {
            std::cerr << "Error with Lambda::GetFunction. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }

    std::cout << "The resources will be deleted. Press return to continue, ";
    std::getline(std::cin, answer);

    // 9. Delete the Lambda function.
    bool result = deleteLambdaFunction(client);

    // 10. Delete the IAM role.
    return result && deleteIamRole(clientConfig);
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue &jsonPayload,
                                     Aws::Lambda::Model::LogType logType,
                                     Aws::Lambda::Model::InvokeResult &invokeResult,
                                     const Aws::Lambda::LambdaClient &client) {

    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
    resources are
     * available. The Invoke function is called repeatedly until the resources are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
            "FunctionTest");
    }

```



```

    *payload << jsonPayload.View().WriteReadable();
    request.SetBody(payload);
    request.SetContentType("application/json");
    Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

    if (outcome.IsSuccess()) {
        invokeResult = std::move(outcome.GetResult());
        result = true;
        break;
    }

    // ACCESS_DENIED: because the role is not available yet.
    // RESOURCE_CONFLICT: because the Lambda function is being created or
updated.
    else if ((outcome.GetError().GetErrorType() ==
        Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
        (outcome.GetError().GetErrorType() ==
        Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
        if ((seconds % 5) == 0) { // Log status every 10 seconds.
            std::cout << "Waiting for the invoke api to be available, status "
<<
                ((outcome.GetError().GetErrorType() ==
                    Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                    "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
seconds
                << " seconds elapsed." << std::endl;
        }
    }
    else {
        std::cerr << "Error with Lambda::InvokeRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        break;
    }
    ++seconds;
    std::this_thread::sleep_for(std::chrono::seconds(1));
} while (seconds < 60);

return result;
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

MediaConvert examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with MediaConvert.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateJob

The following code example shows how to use CreateJob.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Create an AWS Elemental MediaConvert job.
/#!
  \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                          Access Management (IAM) role for the job.
  \param fileInput: A URI to an input file that is stored in Amazon Simple Storage
Service
                          (Amazon S3) or on an HTTP(S) server.
  \param fileOutput: A URI for an Amazon S3 output location and the output file name
base.
  \param jobSettingsFile: An optional JSON settings file.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
                                     const Aws::String &jobSettingsFile,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
  Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

  createJobRequest.SetRole(mediaConvertRole);
  Aws::Http::HeaderValueCollection hvc;
  hvc.emplace("Customer", "Amazon");
  createJobRequest.SetUserMetadata(hvc);

  if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
  {
    std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
    if (!jobSettingsStream) {
      std::cerr << "Unable to open the job template file." << std::endl;
      return false;
    }
  }
}

```

```

std::vector<char> buffer(jobSettingsStream.tellg());
jobSettingsStream.seekg(0);
jobSettingsStream.read(buffer.data(), buffer.size());
std::string jobSettingsJSON(buffer.data(), buffer.size());
size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
if (pos != std::string::npos) {
    jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER), fileInput);
}

pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
if (pos != std::string::npos) {
    jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
}
Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);

createJobRequest.SetSettings(jobSettings);
}
else { // Configure the job settings programmatically.
    Aws::MediaConvert::Model::JobSettings jobSettings;
    jobSettings.SetAdAvailOffset(0);
    Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
    jobSettings.SetTimecodeConfig(timecodeConfig);

// Configure the output group.
    Aws::MediaConvert::Model::OutputGroup outputGroup;
    outputGroup.SetName("File Group");
    Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
    outputGroupSettings.SetType(
        Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
    Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
    fileGroupSettings.SetDestination(fileOutput);
    outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
    outputGroup.SetOutputGroupSettings(outputGroupSettings);

    Aws::MediaConvert::Model::Output output;
    output.SetNameModifier("_1");

    Aws::MediaConvert::Model::VideoDescription videoDescription;
    videoDescription.SetScalingBehavior(
        Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);

```

```
videoDescription.SetTimecodeInsertion(
    Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);
videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
videoDescription.SetSharpness(50);

videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
videoDescription.SetDropFrameTimecode(
    Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
videoDescription.SetColorMetadata(
    Aws::MediaConvert::Model::ColorMetadata::INSERT);

Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
Aws::MediaConvert::Model::H264Settings h264Settings;
h264Settings.SetNumberReferenceFrames(3);
h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
h264Settings.SetSoftness(0);
h264Settings.SetGopClosedCadence(1);
h264Settings.SetGopSize(90);
h264Settings.SetSlices(1);
h264Settings.SetGopBReference(
    Aws::MediaConvert::Model::H264GopBReference::DISABLED);
h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
h264Settings.SetSpatialAdaptiveQuantization(
    Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
h264Settings.SetTemporalAdaptiveQuantization(
    Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
h264Settings.SetFlickerAdaptiveQuantization(
    Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
h264Settings.SetEntropyEncoding(
    Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
h264Settings.SetBitrate(5000000);
h264Settings.SetFramerateControl(
    Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
h264Settings.SetRateControlMode(
    Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
h264Settings.SetMinIInterval(0);
```

```
h264Settings.SetAdaptiveQuantization(
    Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);
h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
h264Settings.SetFieldEncoding(
    Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
h264Settings.SetSceneChangeDetect(
    Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
h264Settings.SetQualityTuningLevel(
    Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
h264Settings.SetFramerateConversionAlgorithm(
    Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
h264Settings.SetUnregisteredSeiTimecode(
    Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
h264Settings.SetGopSizeUnits(
    Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
h264Settings.SetFramerateNumerator(30);
h264Settings.SetFramerateDenominator(1);
h264Settings.SetParNumerator(1);
h264Settings.SetParDenominator(1);
videoCodecSettings.SetH264Settings(h264Settings);
videoDescription.SetCodecSettings(videoCodecSettings);
output.SetVideoDescription(videoDescription);

Aws::MediaConvert::Model::AudioDescription audioDescription;
audioDescription.SetLanguageCodeControl(
    Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
Aws::MediaConvert::Model::AacSettings aacSettings;
aacSettings.SetAudioDescriptionBroadcasterMix(
    Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
aacSettings.SetRateControlMode(
    Aws::MediaConvert::Model::AacRateControlMode::CBR);
aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
aacSettings.SetCodingMode(
    Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
```

```
    aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
    aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
aacSettings.SetBitrate(64000);
audioCodecSettings.SetAacSettings(aacSettings);
audioDescription.SetCodecSettings(audioCodecSettings);
Aws::Vector<Aws::MediaConvert::Model::AudioDescription> audioDescriptions;
audioDescriptions.emplace_back(audioDescription);
output.SetAudioDescriptions(audioDescriptions);

Aws::MediaConvert::Model::ContainerSettings mp4container;
mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
Aws::MediaConvert::Model::Mp4Settings mp4Settings;
mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
mp4Settings.SetMoovPlacement(
    Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
mp4container.SetMp4Settings(mp4Settings);
output.SetContainerSettings(mp4container);

outputGroup.AddOutputs(output);
jobSettings.AddOutputGroups(outputGroup);

// Configure inputs.
Aws::MediaConvert::Model::Input input;
input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
input.SetFilterStrength(0);

input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
input.SetTimecodeSource(
    Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
input.SetFileInput(fileInput);

Aws::MediaConvert::Model::AudioSelector audioSelector;
audioSelector.SetOffset(0);
audioSelector.SetDefaultSelection(
    Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
audioSelector.SetProgramSelection(1);
audioSelector.SetSelectorType(
```

```
        Aws::MediaConvert::Model::AudioSelectorType::TRACK);
    audioSelector.AddTracks(1);
    input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

    Aws::MediaConvert::Model::VideoSelector videoSelector;
    videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
    input.SetVideoSelector(videoSelector);

    jobSettings.AddInputs(input);

    createJobRequest.SetSettings(jobSettings);
}

Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(
    createJobRequest);
if (outcome.IsSuccess()) {
    std::cout << "Job successfully created with ID - "
                << outcome.GetResult().GetJob().GetId() << std::endl;
}
else {
    std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [CreateJob](#) in *AWS SDK for C++ API Reference*.

GetJob

The following code example shows how to use GetJob.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```
//! Retrieve the information for a specific completed transcoding job.
/*!
  \param jobID: A job ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
    const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
        request);
    if (outcome.IsSuccess()) {
        std::cout << outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
            << std::endl;
    }
    else {
        std::cerr << "DescribeEndpoints error - " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetJob](#) in *AWS SDK for C++ API Reference*.

ListJobs

The following code example shows how to use ListJobs.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Retrieve a list of created jobs.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::listJobs(
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;
    Aws::String nextToken; // Used to handle paginated results.
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome = client.ListJobs(
            request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
                std::cout << " " << job.Jsonize().View().WriteReadable() <<
std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
outcome.GetError().GetMessage()
                << std::endl;
            result = false;
            break;
        }
    } while (!nextToken.empty());

    return result;
}
```

- For API details, see [ListJobs](#) in *AWS SDK for C++ API Reference*.

Amazon RDS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon RDS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon RDS

The following code examples show how to get started using Amazon RDS.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)
```

```
# Set this project's name.
project("hello_rds")

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_rds.cpp)

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

Code for the `hello_rds.cpp` source file.

```
#include <aws/core/Aws.h>
```

```
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 * A "Hello Rds" starter application which initializes an Amazon Relational
 * Database Service (Amazon RDS) client and
 * describes the Amazon RDS instances.
 *
 * main function
 *
 * Usage: 'hello_rds'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
            }
        }
    }
}
```

```
        marker = outcome.GetResult().GetMarker();
    } else {
        result = 1;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        break;
    }
} while (!marker.empty());

std::cout << instanceDBIDs.size() << " RDS instances found." << std::endl;
for (auto &instanceDBID: instanceDBIDs) {
    std::cout << " Instance: " << instanceDBID << std::endl;
}
}

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

- For API details, see [DescribeDBInstances](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateDBInstance

The following code example shows how to use CreateDBInstance.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBInstanceRequest request;
request.SetDBName(DB_NAME);
request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
request.SetEngine(engineVersion.GetEngine());
request.SetEngineVersion(engineVersion.GetEngineVersion());
request.SetDBInstanceClass(dbInstanceClass);
request.SetStorageType(DB_STORAGE_TYPE);
request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
              << outcome.GetError().GetMessage()
              << std::endl;
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for C++ API Reference*.

CreateDBParameterGroup

The following code example shows how to use CreateDBParameterGroup.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBParameterGroupRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
request.SetDBParameterGroupFamily(dbParameterGroupFamily);
request.SetDescription("Example parameter group.");

Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
    client.CreateDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully created."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for C++ API Reference*.

CreateDBSnapshot

The following code example shows how to use CreateDBSnapshot.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBSnapshotRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
        client.CreateDBSnapshot(request);

    if (outcome.IsSuccess()) {
        std::cout << "Snapshot creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBSnapshot. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for C++ API Reference*.

DeleteDBInstance

The following code example shows how to use DeleteDBInstance.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBInstanceRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);
    request.SetSkipFinalSnapshot(true);
    request.SetDeleteAutomatedBackups(true);

    Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
        client.DeleteDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB instance deletion has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for C++ API Reference*.

DeleteDBParameterGroup

The following code example shows how to use DeleteDBParameterGroup.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DeleteDBParameterGroupRequest request;
request.SetDBParameterGroupName(parameterGroupName);

Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
    client.DeleteDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::DeleteDBParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
    result = false;
}
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for C++ API Reference*.

DescribeDBEngineVersions

The following code example shows how to use `DescribeDBEngineVersions`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets available DB engine versions for an engine name and
    //! an optional parameter group family.
    /*!
    \sa getDBEngineVersions()
    \param engineName: A DB engine name.
    \param parameterGroupFamily: A parameter group family name, ignored if empty.
    \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
    routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                          const Aws::String &parameterGroupFamily,
                                          Aws::Vector<Aws::RDS::Model::DBEngineVersion>
&engineVersionsResult,
                                          const Aws::RDS::RDSClient &client) {
        Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
        request.SetEngine(engineName);
        if (!parameterGroupFamily.empty()) {
            request.SetDBParameterGroupFamily(parameterGroupFamily);
        }

        engineVersionsResult.clear();
        Aws::String marker; // Used for pagination.

        do {
            if (!marker.empty()) {

```

```
        request.SetMarker(marker);
    }

    Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
        client.DescribeDBEngineVersions(request);

    if (outcome.IsSuccess()) {
        auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
        engineVersionsResult.insert(engineVersionsResult.end(),
engineVersions.begin(),
                                engineVersions.end());
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

} while (!marker.empty());

return true;
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for C++ API Reference*.

DescribeDBInstances

The following code example shows how to use DescribeDBInstances.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets a DB instance description.
    /*!
    \sa describeDBInstance()
    \param dbInstanceIdentifier: A DB instance identifier.
    \param instanceResult: The 'DBInstance' object containing the description.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                         Aws::RDS::Model::DBInstance &instanceResult,
                                         const Aws::RDS::RDSClient &client) {
        Aws::RDS::Model::DescribeDBInstancesRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);

        Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

        bool result = true;
        if (outcome.IsSuccess()) {
            instanceResult = outcome.GetResult().GetDBInstances()[0];
        }
        else if (outcome.GetError().GetErrorType() !=
                Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
            result = false;
            std::cerr << "Error with RDS::DescribeDBInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
        else {
            instanceResult = Aws::RDS::Model::DBInstance();
        }

        return result;
    }
}

```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for C++ API Reference*.

DescribeDBParameterGroups

The following code example shows how to use DescribeDBParameterGroups.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
    client.DescribeDBParameterGroups(request);

if (outcome.IsSuccess()) {
    std::cout << "DB parameter group named '" <<
        PARAMETER_GROUP_NAME << "' already exists." << std::endl;
    dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
}

else {
    std::cerr << "Error with RDS::DescribeDBParameterGroups. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for C++ API Reference*.

DescribeDBParameters

The following code example shows how to use DescribeDBParameters.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
```



```
Aws::RDS::Model::DescribeDBParametersRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
if (!marker.empty()) {
    request.SetMarker(marker);
}
if (!source.empty()) {
    request.SetSource(source);
}

Aws::RDS::Model::DescribeDBParametersOutcome outcome =
    client.DescribeDBParameters(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
        outcome.GetResult().GetParameters();
    for (const Aws::RDS::Model::Parameter &parameter: parameters) {
        if (!namePrefix.empty()) {
            if (parameter.GetParameterName().find(namePrefix) == 0) {
                parametersResult.push_back(parameter);
            }
        }
        else {
            parametersResult.push_back(parameter);
        }
    }

    marker = outcome.GetResult().GetMarker();
}
else {
    std::cerr << "Error with RDS::DescribeDBParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
} while (!marker.empty());

return true;
}
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for C++ API Reference*.

DescribeDBSnapshots

The following code example shows how to use DescribeDBSnapshots.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DescribeDBSnapshotsRequest request;
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
        client.DescribeDBSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBSnapshots()[0];
    }
    else {
        std::cerr << "Error with RDS::DescribeDBSnapshots. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
```

- For API details, see [DescribeDBSnapshots](#) in *AWS SDK for C++ API Reference*.

DescribeOrderableDBInstanceOptions

The following code example shows how to use DescribeOrderableDBInstanceOptions.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {

    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);
```

```

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
            outcome.GetResult().GetOrderableDBInstanceOptions();
        for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
        {
            const Aws::String &instanceClass = option.GetDBInstanceClass();
            if (instanceClass.find("micro") != std::string::npos) {
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                    instanceClass) ==
                    instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
        }
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

std::cout << "The available micro DB instance classes for your database engine
are:"
    << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which micro DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for C++ API Reference*.

ModifyDBParameterGroup

The following code example shows how to use `ModifyDBParameterGroup`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::ModifyDBParameterGroupRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
request.SetParameters(updateParameters);

Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
    client.ModifyDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully modified."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::ModifyDBParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for C++ API Reference*.

Scenarios

Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Routine which creates an Amazon RDS instance and demonstrates several operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
                << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();
}
```

```

std::cout << "Checking for an existing DB parameter group named '" <<
    PARAMETER_GROUP_NAME << "'." << std::endl;
Aws::String dbParameterGroupFamily("Undefined");
bool parameterGroupFound = true;
{
    // 1. Check if the DB parameter group already exists.
    Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

    Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
        client.DescribeDBParameterGroups(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB parameter group named '" <<
            PARAMETER_GROUP_NAME << "' already exists." << std::endl;
        dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
        std::cout << "DB parameter group named '" <<
            PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
        parameterGroupFound = false;
    }
    else {
        std::cerr << "Error with RDS::DescribeDBParameterGroups. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

if (!parameterGroupFound) {
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 2. Get available engine versions for the specified engine.
    if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
        engineVersions, client)) {
        return false;
    }

    std::cout << "Getting available database engine versions for " << DB_ENGINE
        << "."

```

```

        << std::endl;
    std::vector<Aws::String> families;
    for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
        Aws::String family = version.GetDBParameterGroupFamily();
        if (std::find(families.begin(), families.end(), family) ==
            families.end()) {
            families.push_back(family);
            std::cout << " " << families.size() << ": " << family << std::endl;
        }
    }
}

int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
    static_cast<int>(families.size()));
dbParameterGroupFamily = families[choice - 1];
}
if (!parameterGroupFound) {
    // 3. Create a DB parameter group.
    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
        client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
            << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "Let's set some parameter values in your parameter group."
    << std::endl;

Aws::String marker;
Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
// 4. Get the parameters in the DB parameter group.

```



```

    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                        autoIncrementParameters,
                        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataTypes() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                << " is described as: " <<
                autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                    << autoIncParameter.GetParameterValue()
                    << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                    Aws::String("Enter a new value in the range ") +
                    autoIncParameter.GetAllowedValues() + ": ",
                    splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);
            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                    << std::endl;
            }
        }
    }

    {
        // 5. Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);
    }
}

```

```

    Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
        client.ModifyDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully modified."
            << std::endl;
    }
    else {
        std::cerr << "Error with RDS::ModifyDBParameterGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

std::cout
    << "You can get a list of parameters you've set by specifying a source
of 'user'."
    << std::endl;

Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
// 6. Display the modified parameters in the group.
if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                    client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

for (const auto &userParameter: userParameters) {
    std::cout << " " << userParameter.GetParameterName() << ", " <<
        userParameter.GetDescription() << ", parameter value - "
        << userParameter.GetParameterValue() << std::endl;
}

printAsterisksLine();
std::cout << "Checking for an existing DB instance." << std::endl;

Aws::RDS::Model::DBInstance dbInstance;
// 7. Check if the DB instance already exists.
if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

```

```

if (dbInstance.DbInstancePortHasBeenSet()) {
    std::cout << "The DB instance already exists." << std::endl;
}
else {
    std::cout << "Let's create a DB instance." << std::endl;
    const Aws::String administratorName = askQuestion(
        "Enter an administrator username for the database: ");
    const Aws::String administratorPassword = askQuestion(
        "Enter a password for the administrator (at least 8 characters): ");
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 8. Get a list of available engine versions.
    if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    std::cout << "The available engines for your parameter group are:" <<
std::endl;

    int index = 1;
    for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
    {
        std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
            << std::endl;
        ++index;
    }
    int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
    const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

    Aws::String dbInstanceClass;
    // 9. Get a list of micro instance classes.
    if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
        engineVersion.GetEngineVersion(),
        dbInstanceClass,
        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }
}

```

```

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
        << "' and database '" << DB_NAME << "'.\n"
        << "The DB instance is configured to use your custom parameter
group '"
        << PARAMETER_GROUP_NAME << "',\n"
        << "selected engine version " << engineVersion.GetEngineVersion()
        << ",\n"
        << "selected DB instance class '" << dbInstanceClass << "',"
        << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
        << " storage.\nThis typically takes several minutes." <<
std::endl;

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBName(DB_NAME);
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetEngine(engineVersion.GetEngine());
    request.SetEngineVersion(engineVersion.GetEngineVersion());
    request.SetDBInstanceClass(dbInstanceClass);
    request.SetStorageType(DB_STORAGE_TYPE);
    request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
        client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
        << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
        << outcome.GetError().GetMessage()
        << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

```

```

int counter = 0;
// 11. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
            << counter
            << " seconds." << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}

printAsterisksLine();

// 12. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbInstance);

printAsterisksLine();

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB instance (y/n)? ") {
    Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
        Aws::String(Aws::Utils::UUID::RandomUUID()));
    {

```

```
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 13. Create a snapshot of the DB instance.
        Aws::RDS::Model::CreateDBSnapshotRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
            client.CreateDBSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBSnapshot. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }
    }

    std::cout << "Waiting for snapshot to become available." << std::endl;

    Aws::RDS::Model::DBSnapshot snapshot;
    counter = 0;
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 600) {
            std::cerr << "Wait for snapshot to be available timed out after "
                << counter
                << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        // 14. Wait for the snapshot to become available.
        Aws::RDS::Model::DescribeDBSnapshotsRequest request;
```

```
request.SetDBSnapshotIdentifier(snapshotID);

Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
    client.DescribeDBSnapshots(request);

if (outcome.IsSuccess()) {
    snapshot = outcome.GetResult().GetDBSnapshots()[0];
}
else {
    std::cerr << "Error with RDS::DescribeDBSnapshots. "
                << outcome.GetError().GetMessage()
                << std::endl;
    cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
    return false;
}

if ((counter % 20) == 0) {
    std::cout << "Current snapshot status is '"
                << snapshot.GetStatus()
                << "' after " << counter << " seconds." << std::endl;
}
} while (snapshot.GetStatus() != "available");

if (snapshot.GetStatus() != "available") {
    std::cout << "A snapshot has been created." << std::endl;
}
}

printAsterisksLine();

bool result = true;
if (askYesNoQuestion(
    "Do you want to delete the DB instance and parameter group (y/n)? ")) {
    result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
}

return result;
}

//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
```

```

\sa getDBParameters()
\param parameterGroupName: The name of the parameter group.
\param namePrefix: Prefix string to filter results by parameter name.
\param source: A source such as 'user', ignored if empty.
\param parametersResult: Vector of 'Parameter' objects returned by the routine.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
            client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
    }
}

```



```

        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                       const Aws::String &parameterGroupFamily,
                                       Aws::Vector<Aws::RDS::Model::DBEngineVersion>
&engineVersionsResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =

```

```

        client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
engineVersions.begin(),
                                     engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }

    } while (!marker.empty());

    return true;
}

//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
        client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
}

```

```

else if (outcome.GetError().GetErrorType() !=
        Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
    result = false;
    std::cerr << "Error with RDS::DescribeDBInstances. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
// This example does not log an error if the DB instance does not exist.
// Instead, instanceResult is set to empty.
else {
    instanceResult = Aws::RDS::Model::DBInstance();
}

return result;
}

//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);
    } while (marker.empty());
}

```

```

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
            outcome.GetResult().GetOrderableDBInstanceOptions();
        for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
        {
            const Aws::String &instanceClass = option.GetDBInstanceClass();
            if (instanceClass.find("micro") != std::string::npos) {
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                    instanceClass) ==
                    instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
        }
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

std::cout << "The available micro DB instance classes for your database engine
are:"
    << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which micro DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.

```

```

\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                    << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
                result = false;
            }
        }

        std::cout
            << "Waiting for DB instance to delete before deleting the parameter
group."
            << std::endl;
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out after " <<
counter
                    << " seconds." << std::endl;
            }
        }
    }
}

```

```

        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    // 16. Wait for the DB instance to be deleted.
    if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
        return false;
    }

    if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) == 0)
{
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.DBInstanceIdentifierHasBeenSet());
}

if (!parameterGroupName.empty()) {
    // 17. Delete the parameter group.
    Aws::RDS::Model::DeleteDBParameterGroupRequest request;
    request.SetDBParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
        client.DeleteDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBParameterGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.

- [CreateDBInstance](#)
- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

Amazon Rekognition examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Rekognition.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Rekognition

The following code example shows how to get started using Amazon Rekognition.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

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

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

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

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

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

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

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
may need to uncomment this
```



```

                                # and set the proper subdirectory to the
executables' location.

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

add_executable(${PROJECT_NAME}
    hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_rekognition.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 * A "Hello Rekognition" starter application which initializes an Amazon
Rekognition client and
 * lists the Amazon Rekognition collections in the current account and region.
 *
 * main function
 *
 * Usage: 'hello_rekognition'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
    }
}

```

```
Aws::Rekognition::Model::ListCollectionsRequest request;
Aws::Rekognition::Model::ListCollectionsOutcome outcome =
    rekognitionClient.ListCollections(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::String>& collectionsIds =
outcome.GetResult().GetCollectionIds();
    if (!collectionsIds.empty()) {
        std::cout << "collectionsIds: " << std::endl;
        for (auto &collectionId : collectionsIds) {
            std::cout << "- " << collectionId << std::endl;
        }
    } else {
        std::cout << "No collections found" << std::endl;
    }
} else {
    std::cerr << "Error with ListCollections: " << outcome.GetError()
        << std::endl;
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}
```

- For API details, see [ListCollections](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)


Actions

DetectLabels

The following code example shows how to use DetectLabels.

For more information, see [Detecting labels in an image](#).

SDK for C++

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Detect instances of real-world entities within an image by using Amazon
Rekognition
/*!
 \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
containing an image.
 \param imageKey: The Amazon S3 key of an image object.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3object;
    s3object.SetBucket(imageBucket);
    s3object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
```

```
        for (const Aws::Rekognition::Model::Label &label: labels) {
            std::cout << label.GetName() << ": " << label.GetConfidence() <<
std::endl;
        }
    }
} else {
    std::cerr << "Error while detecting labels: '"
        << outcome.GetError().GetMessage()
        << "'" << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for C++ API Reference*.

Amazon S3 examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon S3.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon S3

The following code examples show how to get started using Amazon S3.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

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

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

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

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

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

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

```
    AWSSDK_COPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

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

Code for the hello_s3.cpp source file.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
// You don't normally have to test that you are authenticated. But the S3
service permits anonymous requests, thus the s3Client will return "success" and 0
buckets even if you are unauthenticated, which can be confusing to a new user.
auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
auto creds = provider->GetAWSCredentials();
if (creds.IsEmpty()) {
    std::cerr << "Failed authentication" << std::endl;
}

Aws::S3::S3Client s3Client(clientConfig);
auto outcome = s3Client.ListBuckets();

if (!outcome.IsSuccess()) {
    std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
    result = 1;
} else {
    std::cout << "Found " << outcome.GetResult().GetBuckets().size()
              << " buckets\n";
    for (auto &bucket: outcome.GetResult().GetBuckets()) {
        std::cout << bucket.GetName() << std::endl;
    }
}
}

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

- For API details, see [ListBuckets](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CopyObject

The following code example shows how to use CopyObject.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CopyObject(const Aws::String &objectKey, const Aws::String
&fromBucket, const Aws::String &toBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
           .WithKey(objectKey)
           .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: CopyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully copied " << objectKey << " from " << fromBucket
        <<
            " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CopyObject](#) in *AWS SDK for C++ API Reference*.

CreateBucket

The following code example shows how to use CreateBucket.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CreateBucket(const Aws::String &bucketName,
                              const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    //TODO(user): Change the bucket location constraint enum to your target Region.
    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: CreateBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for C++ API Reference*.

DeleteBucket

The following code example shows how to use DeleteBucket.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName,
                              const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for C++ API Reference*.

DeleteBucketPolicy

The following code example shows how to use DeleteBucketPolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
    client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for C++ API Reference*.

DeleteBucketWebsite

The following code example shows how to use DeleteBucketWebsite.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketWebsite(const Aws::String &bucketName,
                                      const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketWebsite](#) in *AWS SDK for C++ API Reference*.

DeleteObject

The following code example shows how to use DeleteObject.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
           .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for C++ API Reference*.

DeleteObjects

The following code example shows how to use DeleteObjects.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String& objectKey : objectKeys)
    {
        deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i)
        {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1)
            {
                std::cout << ", ";
            }
        }
    }
}
```

```

    }

    std::cout << " from bucket " << fromBucket << "." << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [DeleteObjects](#) in *AWS SDK for C++ API Reference*.

GetBucketAcl

The following code example shows how to use `GetBucketAcl`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::GetBucketAcl(const Aws::String &bucketName,
                             const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3_client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Grant> grants =

```

```
        outcome.GetResult().GetGrants();

    for (auto it = grants.begin(); it != grants.end(); it++) {
        Aws::S3::Model::Grant grant = *it;
        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        std::cout << "For bucket " << bucketName << ": "
                  << std::endl << std::endl;

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                      << GetGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                      << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                      << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:         "
                      << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
                  GetPermissionString(grant.GetPermission()) <<
                  std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
```



```
/*!
 \sa GetGranteeTypeString()
 \param type Type enumeration.
 */

Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \sa GetPermissionString()
 \param permission Permission enumeration.
 */

Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
}
```

```
    }  
  
    return "Permission unknown";  
}
```

- For API details, see [GetBucketAcl](#) in *AWS SDK for C++ API Reference*.

GetBucketPolicy

The following code example shows how to use `GetBucketPolicy`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetBucketPolicy(const Aws::String &bucketName,  
                                const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::S3::S3Client s3_client(clientConfig);  
  
    Aws::S3::Model::GetBucketPolicyRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::GetBucketPolicyOutcome outcome =  
        s3_client.GetBucketPolicy(request);  
  
    if (!outcome.IsSuccess()) {  
        const Aws::S3::S3Error &err = outcome.GetError();  
        std::cerr << "Error: GetBucketPolicy: "  
                  << err.GetExceptionName() << ": " << err.GetMessage() <<  
std::endl;  
    }  
    else {  
        Aws::StringStream policy_stream;  
        Aws::String line;
```

```

outcome.GetResult().GetPolicy() >> line;
policy_stream << line;

std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
    policy_stream.str() << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for C++ API Reference*.

GetBucketWebsite

The following code example shows how to use `GetBucketWebsite`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::GetWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3_client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
            << err.GetMessage() << std::endl;
    }
}

```

```

else {
    Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

    std::cout << "Success: GetBucketWebsite: "
               << std::endl << std::endl
               << "For bucket '" << bucketName << "':"
               << std::endl
               << "Index page : "
               << websiteResult.GetIndexDocument().GetSuffix()
               << std::endl
               << "Error page: "
               << websiteResult.GetErrorDocument().GetKey()
               << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [GetBucketWebsite](#) in *AWS SDK for C++ API Reference*.

GetObject

The following code example shows how to use `GetObject`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::GetObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);
}

```

```

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetObject](#) in *AWS SDK for C++ API Reference*.

GetObjectAcl

The following code example shows how to use `GetObjectAcl`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::GetObjectAcl(const Aws::String &bucketName,
                             const Aws::String &objectKey,
                             const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =

```

```
s3_client.GetObjectAcl(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: GetObjectAcl: "
                << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
}
else {
    Aws::Vector<Aws::S3::Model::Grant> grants =
        outcome.GetResult().GetGrants();

    for (auto it = grants.begin(); it != grants.end(); it++) {
        std::cout << "For object " << objectKey << ": "
                  << std::endl << std::endl;

        Aws::S3::Model::Grant grant = *it;
        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                      << GetGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                      << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                      << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:         "
                      << grantee.GetURI() << std::endl;
        }
    }
}
```

```

        std::cout << "Permission:    " <<
                    GetPermissionString(grant.GetPermission()) <<
                    std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \fn GetGranteeTypeString()
 \param type Type enumeration.
 */

Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \fn GetPermissionString()
 \param permission Permission enumeration.
 */

Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                    "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:

```

```

        return "Can read this object's data and its metadata";
    case Aws::S3::Model::Permission::READ_ACP:
        return "Can read this object's permissions";
        // case Aws::S3::Model::Permission::WRITE // Not applicable.
    case Aws::S3::Model::Permission::WRITE_ACP:
        return "Can write this object's permissions";
    default:
        return "Permission unknown";
    }
}

```

- For API details, see [GetObjectAcl](#) in *AWS SDK for C++ API Reference*.

ListBuckets

The following code example shows how to use ListBuckets.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::ListBuckets(const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    }
    else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }
}

```



```
    }  
  }  
  
  return result;  
}
```

- For API details, see [ListBuckets](#) in *AWS SDK for C++ API Reference*.

ListObjectsV2

The following code example shows how to use ListObjectsV2.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::ListObjects(const Aws::String &bucketName,  
                             const Aws::Client::ClientConfiguration &clientConfig) {  
  Aws::S3::S3Client s3_client(clientConfig);  
  
  Aws::S3::Model::ListObjectsV2Request request;  
  request.WithBucket(bucketName);  
  
  Aws::String continuationToken; // Used for pagination.  
  Aws::Vector<Aws::S3::Model::Object> allObjects;  
  
  do {  
    if (!continuationToken.empty()) {  
      request.SetContinuationToken(continuationToken);  
    }  
  
    auto outcome = s3_client.ListObjectsV2(request);  
  
    if (!outcome.IsSuccess()) {  
      std::cerr << "Error: ListObjects: " <<  
                 outcome.GetError().GetMessage() << std::endl;  
      return false;  
    }  
  }  
}
```

```

    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetNextContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
}

return true;
}

```

- For API details, see [ListObjectsV2](#) in *AWS SDK for C++ API Reference*.

PutBucketAcl

The following code example shows how to use PutBucketAcl.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

bool AwsDoc::S3::PutBucketAcl(const Aws::String &bucketName,
                             const Aws::String &ownerID,
                             const Aws::String &granteePermission,
                             const Aws::String &granteeType,
                             const Aws::String &granteeID,
                             const Aws::Client::ClientConfiguration &clientConfig,
                             const Aws::String &granteeDisplayName,
                             const Aws::String &granteeEmailAddress,

```

```
        const Aws::String &granteeURI) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(SetGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeDisplayName.empty()) {
        grantee.SetDisplayName(granteeDisplayName);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(SetGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3_client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
```

```

        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: PutBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

/*! Routine which converts a human-readable string to a built-in type enumeration.
 */
\sa SetGranteePermission()
\param access Human readable string.
*/

Aws::S3::Model::Permission SetGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

/*! Routine which converts a human-readable string to a built-in type enumeration.
 */
\sa SetGranteeType()
\param type Human readable string.
*/

Aws::S3::Model::Type SetGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
}

```

```
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for C++ API Reference*.

PutBucketPolicy

The following code example shows how to use PutBucketPolicy.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutBucketPolicy(const Aws::String &bucketName,
                                const Aws::String &policyBody,
                                const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3_client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
}
```

```

    else {
        std::cout << "Set the following policy body for the bucket '" <<
            bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
 \sa GetPolicyString()
 \param userArn Aws user Amazon Resource Name (ARN).
     For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
 \param bucketName Name of a bucket.
 */

Aws::String GetPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
+ userArn +
        \"\" \n"
        "      \"Action\": [ \"s3:GetObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
+ bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "};
}

```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for C++ API Reference*.

PutBucketWebsite

The following code example shows how to use PutBucketWebsite.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPath, const Aws::String
&errorPage,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }
}
```

```
    }  
  
    return outcome.IsSuccess();  
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for C++ API Reference*.

PutObject

The following code example shows how to use PutObject.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutObject(const Aws::String &bucketName,  
                           const Aws::String &fileName,  
                           const Aws::Client::ClientConfiguration &clientConfig) {  
    Aws::S3::S3Client s3_client(clientConfig);  
  
    Aws::S3::Model::PutObjectRequest request;  
    request.SetBucket(bucketName);  
    //We are using the name of the file as the key for the object in the bucket.  
    //However, this is just a string and can be set according to your retrieval  
    needs.  
    request.SetKey(fileName);  
  
    std::shared_ptr<Aws::IOStream> inputData =  
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",  
                                       fileName.c_str(),  
                                       std::ios_base::in |  
                                       std::ios_base::binary);  
  
    if (!*inputData) {  
        std::cerr << "Error unable to read file " << fileName << std::endl;  
        return false;  
    }  
}
```



```
request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3_client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutObject: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Added object '" << fileName << "' to bucket '"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}
```

- For API details, see [PutObject](#) in *AWS SDK for C++ API Reference*.

PutObjectAcl

The following code example shows how to use `PutObjectAcl`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutObjectAcl(const Aws::String &bucketName,
                             const Aws::String &objectKey,
                             const Aws::String &ownerID,
                             const Aws::String &granteePermission,
                             const Aws::String &granteeType,
                             const Aws::String &granteeID,
                             const Aws::Client::ClientConfiguration &clientConfig,
                             const Aws::String &granteeDisplayName,
                             const Aws::String &granteeEmailAddress,
```

```
        const Aws::String &granteeURI) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(SetGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeDisplayName.empty()) {
        grantee.SetDisplayName(granteeDisplayName);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(SetGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3_client.PutObjectAcl(request);
```

```

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: PutObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \sa SetGranteePermission()
 \param access Human readable string.
 */

Aws::S3::Model::Permission SetGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \sa SetGranteeType()
 \param type Human readable string.
 */

Aws::S3::Model::Type SetGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
}

```

```

    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- For API details, see [PutObjectAcl](#) in *AWS SDK for C++ API Reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for Amazon S3 and upload an object.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Generate a pre-signed URL to download an object.

```

//! Routine which demonstrates creating a pre-signed URL to download an object from
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
 \param bucketName: Name of the bucket.
 \param key: Name of an object key.
 \param expirationSeconds: Expiration in seconds for pre-signed URL.
 \param clientConfig: Aws client configuration.
 \return Aws::String: A pre-signed URL.
 */
Aws::String AwsDoc::S3::GeneratePreSignedGetObjectURL(const Aws::String &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
Aws::Client::ClientConfiguration &clientConfig) {

```

```

    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
                                   expirationSeconds);
}

```

Download using libcurl.

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test GetObject with a pre-signed URL.
/*!
 \param presignedURL: A pre-signed URL to get an object from a bucket.
 \param resultString: A string to hold the result.
 \return bool: Function succeeded.
 */
bool AwsDoc::S3::GetObjectWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
    }
}

```

```

        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find( "<?xml" ) == 0)
    {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

Generate a pre-signed URL to upload an object.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::GeneratePreSignedPutObjectURL(const Aws::String &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,

```

```

                                const
    Aws::Client::ClientConfiguration &clientConfig) {
        Aws::S3::S3Client client(clientConfig);
        return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                expirationSeconds);
    }

```

Upload using libcurl.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test PutObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the PutObject request.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;

```

```
result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                          (curl_off_t)data.size());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
    return false;
}

std::stringstream outWriteString;

result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}
```



```
result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

std::string outString = outWriteString.str();
if (outString.empty()) {
    std::cout << "Successfully put object." << std::endl;
    return true;
}
else {
    std::cout << "A server error was encountered, output:\n" << outString
        << std::endl;
    return false;
}
}
```

Get started with buckets and objects

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/Utils/UUID.h>
#include <aws/core/Utils/StringUtils.h>
#include <aws/core/Utils/memory/stl/AWSAllocator.h>
#include <aws/core/Utils/memory/stl/AWSStreamFwd.h>
#include <fstream>
#include "awsdoc/s3/s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
         \sa DeleteBucket()
         \param bucketName The S3 bucket's name.
         \param client An S3 client.
        */
        static bool
        DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
         \sa DeleteObjectFromBucket()
         \param bucketName The S3 bucket's name.
```

```

        \param key The key for the object in the S3 bucket.
        \param client An S3 client.
    */
    static bool
    DeleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                          Aws::S3::S3Client &client);
    }
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \sa S3_GettingStartedScenario()
    \param uploadFilePath Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath Path for saving a downloaded S3 object.
    \param clientConfig Aws client configuration.
    */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = "doc-example-bucket-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }
    }
}

```

```
Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: CreateBucket: " <<
        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    return false;
}
else {
    std::cout << "Created the bucket, '" << bucketName <<
        "', in the region, '" << clientConfig.region << "'." <<
std::endl;
}
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            uploadFilePath,
            std::ios_base::in |
            std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath << "'."
            << std::endl;
        AwsDoc::S3::DeleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
```

```

        outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::DeleteBucket(bucketName, client);
        return false;
    }
    else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    }
    else {
        std::cout << "Downloaded the object with the key, '" << key
            << "', in the bucket, '"
            << bucketName << "'." << std::endl;

        Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::OFStream outputStream(saveFilePath,
            std::ios_base::out | std::ios_base::binary);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath << "'."
                << std::endl;
        }
        else {
            outputStream << ioStream.rdbuf();
            std::cout << "Wrote the downloaded object to the file '"
                << saveFilePath << "'." << std::endl;
        }
    }
}

```

```
    }
  }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
  Aws::S3::Model::CopyObjectRequest request;
  request.WithBucket(bucketName)
    .WithKey(copiedToKey)
    .WithCopySource(bucketName + "/" + key);

  Aws::S3::Model::CopyObjectOutcome outcome =
    client.CopyObject(request);
  if (!outcome.IsSuccess()) {
    std::cerr << "Error: CopyObject: " <<
      outcome.GetError().GetMessage() << std::endl;
  }
  else {
    std::cout << "Copied the object with the key, '" << key
      << "', to the key, '" << copiedToKey
      << "', in the bucket, '" << bucketName << "'." << std::endl;
  }
}

// 5. List objects in the bucket.
{
  Aws::S3::Model::ListObjectsV2Request request;
  request.WithBucket(bucketName);

  Aws::String continuationToken;
  Aws::Vector<Aws::S3::Model::Object> allObjects;

  do {
    if (!continuationToken.empty()) {
      request.SetContinuationToken(continuationToken);
    }
    Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
      request);

    if (!outcome.IsSuccess()) {
      std::cerr << "Error: ListObjects: " <<
        outcome.GetError().GetMessage() << std::endl;
      break;
    }
  }
}
```

```

    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " objects in the bucket, '" << bucketName
    << "':" << std::endl;

for (Aws::S3::Model::Object &object: allObjects) {
    std::cout << "    '" << object.GetKey() << "'" << std::endl;
}
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::DeleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::DeleteBucket(bucketName, client);
}

bool AwsDoc::S3::DeleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Deleted the object with the key, '" << key
            << "', from the bucket, '"
            << bucketName << "':" << std::endl;
    }
}

```

```
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Secrets Manager examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Secrets Manager.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

GetSecretValue

The following code example shows how to use `GetSecretValue`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);
```

```
    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
secretsManagerClient.GetSecretValue(
    request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
            << getSecretValueOutcome.GetResult().GetSecretString() <<
std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
            << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for C++ API Reference*.

Amazon SES examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon SES.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateReceiptFilter

The following code example shows how to use CreateReceiptFilter.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create an Amazon Simple Email Service (Amazon SES) receipt filter..
/#!/
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
(CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                     const Aws::String &cidr,
                                     Aws::SES::Model::ReceiptFilterPolicy policy,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
sesClient.CreateReceiptFilter(
    createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
}
```

```

    }
    else {
        std::cerr << "Error creating receipt filter: " <<
            createReceiptFilterOutcome.GetError().GetMessage() << std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}

```

- For API details, see [CreateReceiptFilter](#) in *AWS SDK for C++ API Reference*.

CreateReceiptRule

The following code example shows how to use `CreateReceiptRule`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
 *!
 * \param receiptRuleName: The name for the receipt rule.
 * \param s3BucketName: The name of the S3 bucket for incoming mail.
 * \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
 * \param ruleSetName: The name of the rule set where the receipt rule is added.
 * \param recipients: Aws::Vector of recipients.
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {

```

```
Aws::SES::SESClient sesClient(clientConfiguration);

Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

Aws::SES::Model::S3Action s3Action;
s3Action.SetBucketName(s3BucketName);
s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

Aws::SES::Model::ReceiptAction receiptAction;
receiptAction.SetS3Action(s3Action);

Aws::SES::Model::ReceiptRule receiptRule;
receiptRule.SetName(receiptRuleName);
receiptRule.WithRecipients(recipients);

Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
receiptActionList.emplace_back(receiptAction);
receiptRule.SetActions(receiptActionList);

createReceiptRuleRequest.SetRuleSetName(ruleSetName);
createReceiptRuleRequest.SetRule(receiptRule);

auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created receipt rule." << std::endl;
}
else {
    std::cerr << "Error creating receipt rule. " <<
outcome.GetError().GetMessage()
    << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [CreateReceiptRule](#) in *AWS SDK for C++ API Reference*.

CreateReceiptRuleSet

The following code example shows how to use CreateReceiptRuleSet.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/#!/
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
sesClient.CreateReceiptRuleSet(
    createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateReceiptRuleSet](#) in *AWS SDK for C++ API Reference*.

CreateTemplate

The following code example shows how to use CreateTemplate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                const Aws::String &htmlPart,
                                const Aws::String &subjectPart,
                                const Aws::String &textPart,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;

    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
        createTemplateRequest);

    if (outcome.IsSuccess()) {
```

```

        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
    else {
        std::cerr << "Error creating template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [CreateTemplate](#) in *AWS SDK for C++ API Reference*.

DeleteIdentity

The following code example shows how to use DeleteIdentity.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Delete the specified identity (an email address or a domain).
 *!
 *! \param identity: The identity to delete.
 *! \param clientConfiguration: AWS client configuration.
 *! \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);
}

```



```

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
        deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteIdentity](#) in *AWS SDK for C++ API Reference*.

DeleteReceiptFilter

The following code example shows how to use DeleteReceiptFilter.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
 *!
 *! \param receiptFilterName: The name for the receipt filter.
 *! \param clientConfiguration: AWS client configuration.
 *! \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
}

```

```
Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
sesClient.DeleteReceiptFilter(
    deleteReceiptFilterRequest);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted receipt filter." << std::endl;
}
else {
    std::cerr << "Error deleting receipt filter. "
              << outcome.GetError().GetMessage()
              << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DeleteReceiptFilter](#) in *AWS SDK for C++ API Reference*.

DeleteReceiptRule

The following code example shows how to use DeleteReceiptRule.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
    \param receiptRuleName: The name for the receipt rule.
    \param receiptRuleSetName: The name for the receipt rule set.
    \param clientConfiguration: AWS client configuration.
```

```
\return bool: Function succeeded.
*/
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                     const Aws::String &receiptRuleSetName,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome = sesClient.DeleteReceiptRule(
        deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {
        std::cout << "Error deleting receipt rule. " <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteReceiptRule](#) in *AWS SDK for C++ API Reference*.

DeleteReceiptRuleSet

The following code example shows how to use DeleteReceiptRuleSet.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
sesClient.DeleteReceiptRuleSet(
    deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteReceiptRuleSet](#) in *AWS SDK for C++ API Reference*.

DeleteTemplate

The following code example shows how to use DeleteTemplate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Delete an Amazon Simple Email Service (Amazon SES) template.
/#!
 \param templateName: The name for the template.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
        deleteTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteTemplate](#) in *AWS SDK for C++ API Reference*.

GetTemplate

The following code example shows how to use GetTemplate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Get a template's attributes.
/*!
 \param templateName: The name for the template.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
        getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetTemplate](#) in *AWS SDK for C++ API Reference*.

ListIdentities

The following code example shows how to use `ListIdentities`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ List the identities associated with this account.
/#!/
\param identityType: The identity type enum. "NOT_SET" is a valid option.
\param identities; A vector to receive the retrieved identities.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                Aws::Vector<Aws::String> &identities,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome = sesClient.ListIdentities(
            listIdentitiesRequest);
    }
```

```

    if (outcome.IsSuccess()) {
        const auto &retrievedIdentities = outcome.GetResult().GetIdentities();
        if (!retrievedIdentities.empty()) {
            identities.insert(identities.cend(), retrievedIdentities.cbegin(),
                             retrievedIdentities.cend());
        }
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cout << "Error listing identities. " <<
outcome.GetError().GetMessage()
                << std::endl;
        return false;
    }
} while (!nextToken.empty());

return true;
}

```

- For API details, see [ListIdentities](#) in *AWS SDK for C++ API Reference*.

ListReceiptFilters

The following code example shows how to use ListReceiptFilters.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

/*! List the receipt filters associated with this account.
 *!
 *! \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
 *! \param clientConfiguration: AWS client configuration.
 *! \return bool: Function succeeded.
 *!
 */

```



```

bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
&filters,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
sesClient.ListReceiptFilters(
    listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [ListReceiptFilters](#) in *AWS SDK for C++ API Reference*.

SendEmail

The following code example shows how to use SendEmail.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Send an email to a list of recipients.

```

```
/*!
 \param recipients; Vector of recipient email addresses.
 \param subject: Email subject.
 \param htmlBody: Email body as HTML. At least one body data is required.
 \param textBody: Email body as plain text. At least one body data is required.
 \param senderEmailAddress: Email address of sender. Ignored if empty string.
 \param ccAddresses: Vector of cc addresses. Ignored if empty.
 \param replyToAddress: Reply to email address. Ignored if empty string.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                           const Aws::String &subject,
                           const Aws::String &htmlBody,
                           const Aws::String &textBody,
                           const Aws::String &senderEmailAddress,
                           const Aws::Vector<Aws::String> &ccAddresses,
                           const Aws::String &replyToAddress,
                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::Body message_body;
    if (!htmlBody.empty()) {
        message_body.SetHtml(
            Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
    }

    if (!textBody.empty()) {
        message_body.SetText(
            Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
    }

    Aws::SES::Model::Message message;
    message.SetBody(message_body);
    message.SetSubject(
```

```
        Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

    Aws::SES::Model::SendEmailRequest sendEmailRequest;
    sendEmailRequest.SetDestination(destination);
    sendEmailRequest.SetMessage(message);
    if (!senderEmailAddress.empty()) {
        sendEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendEmail(sendEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [SendEmail](#) in *AWS SDK for C++ API Reference*.

SendTemplatedEmail

The following code example shows how to use `SendTemplatedEmail`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Send a templated email to a list of recipients.
/!*
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
&templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "":\"" << pair.second << "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
}

```

```
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for C++ API Reference*.

UpdateTemplate

The following code example shows how to use UpdateTemplate.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                const Aws::String &htmlPart,
                                const Aws::String &subjectPart,
                                const Aws::String &textPart,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
    } else {
        std::cerr << "Error updating template. " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateTemplate](#) in *AWS SDK for C++ API Reference*.

VerifyEmailIdentity

The following code example shows how to use `VerifyEmailIdentity`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Add an email address to the list of identities associated with this account and
//! initiate verification.
/*!
 \param emailAddress; The email address to add.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }

    else
    {
        std::cerr << "Error initiating email verification. " <<
outcome.GetError().GetMessage()
                << std::endl;
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for C++ API Reference*.

Amazon SNS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon SNS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SNS

The following code examples show how to get started using Amazon SNS.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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



```
# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_sns.cpp)

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

Code for the `hello_sns.cpp` source file.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 * A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 * Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 * main function
 *
 * Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                    outcome.GetResult().GetTopics();
            }
        }
    }
}
```

```

        if (!paginatedTopics.empty()) {
            allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                            paginatedTopics.cend());
        }
    }
    else {
        std::cerr << "Error listing topics " <<
outcome.GetError().GetMessage()
        << std::endl;
        return 1;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

std::cout << "Hello Amazon SNS! You have " << allTopics.size() << " topic"
        << (allTopics.size() == 1 ? "" : "s") << " in your account."
        << std::endl;

if (!allTopics.empty()) {
    std::cout << "Here are your topic ARNs." << std::endl;
    for (const Aws::SNS::Model::Topic &topic: allTopics) {
        std::cout << " * " << topic.GetTopicArn() << std::endl;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- For API details, see [ListTopics](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateTopic

The following code example shows how to use CreateTopic.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/ Create an Amazon Simple Notification Service (Amazon SNS) topic.
/#!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
  topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
```

```
        outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for C++ API Reference*.

DeleteTopic

The following code example shows how to use DeleteTopic.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
 \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
std::endl;
    }
}
```

```

    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteTopic](#) in *AWS SDK for C++ API Reference*.

GetSMSAttributes

The following code example shows how to use `GetSMSAttributes`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Retrieve the default settings for sending SMS messages from your AWS account by
using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration &clientConfiguration)
{
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");
}

```

```
const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
snsClient.GetSMSAttributes(
    request);

if (outcome.IsSuccess()) {
    const Aws::Map<Aws::String, Aws::String> attributes =
        outcome.GetResult().GetAttributes();
    if (!attributes.empty()) {
        for (auto const &att: attributes) {
            std::cout << att.first << ": " << att.second << std::endl;
        }
    }
    else {
        std::cout
            << "AwsDoc::SNS::getSMSType - an empty map of attributes was
retrieved."
            << std::endl;
    }
}
else {
    std::cerr << "Error while getting SMS Type: '"
        << outcome.GetError().GetMessage()
        << "'" << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for C++ API Reference*.

GetTopicAttributes

The following code example shows how to use `GetTopicAttributes`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
topic.
/*!
 \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
snsClient.GetTopicAttributes(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << " * " << attribute.first << " : " << attribute.second
                << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for C++ API Reference*.

ListSubscriptions

The following code example shows how to use ListSubscriptions.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
subscriptions.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
snsClient.ListSubscriptions(
    request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                << outcome.GetError().GetMessage()
                <<

```

```

        std::endl;
        result = false;
        break;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

if (result) {
    if (subscriptions.empty()) {
        std::cout << "No subscriptions found" << std::endl;
    }
    else {
        std::cout << "Subscriptions list:" << std::endl;
        for (auto const &subscription: subscriptions) {
            std::cout << "  * " << subscription.GetSubscriptionArn() <<
std::endl;
        }
    }
}
return result;
}

```

- For API details, see [ListSubscriptions](#) in *AWS SDK for C++ API Reference*.

ListTopics

The following code example shows how to use `ListTopics`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration &clientConfiguration)
{
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << " * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " << outcome.GetError().GetMessage()
<<
                std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}

```

- For API details, see [ListTopics](#) in *AWS SDK for C++ API Reference*.

Publish

The following code example shows how to use Publish.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                const Aws::String &topicARN,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Publish a message with an attribute.

```
static const Aws::String TONE_ATTRIBUTE("tone");
static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                    "sincere"};

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

Aws::SNS::SNSClient snsClient(clientConfiguration);

Aws::SNS::Model::PublishRequest request;
request.SetTopicArn(topicARN);
Aws::String message = askQuestion("Enter a message text to publish. ");
request.SetMessage(message);

if (filteringMessages && askYesNoQuestion(
    "Add an attribute to this message? (y/n) ")) {
    for (size_t i = 0; i < TONES.size(); ++i) {
        std::cout << " " << (i + 1) << ". " << TONES[i] << std::endl;
    }
    int selection = askQuestionForIntRange(
        "Enter a number for an attribute. ",
        1, static_cast<int>(TONES.size()));
    Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
    messageAttributeValue.SetDataType("String");
    messageAttributeValue.SetStringValue(TONES[selection - 1]);
    request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
}

Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

if (outcome.IsSuccess()) {
    std::cout << "Your message was successfully published." << std::endl;
}
else {
    std::cerr << "Error with TopicsAndQueues::Publish. "
        << outcome.GetError().GetMessage()
        << std::endl;
}
```

```
        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

    return false;
}
```

- For API details, see [Publish](#) in *AWS SDK for C++ API Reference*.

SetSMSAttributes

The following code example shows how to use `SetSMSAttributes`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

How to use Amazon SNS to set the `DefaultSMSType` attribute.

```
//! Set the default settings for sending SMS messages.
/*!
 \param smsType: The type of SMS message that you will send by default.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);
}
```

```

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
snsClient.SetSMSAttributes(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
            << outcome.GetError().GetMessage()
            << "'" << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [SetSMSAttributes](#) in *AWS SDK for C++ API Reference*.

Subscribe

The following code example shows how to use `Subscribe`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Subscribe an email address to a topic.

```

/*! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to an email address.
/*!
    \param topicARN: An SNS topic Amazon Resource Name (ARN).
    \param emailAddress: An email address.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,

```

```

        const Aws::String &emailAddress,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
        << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

Subscribe a mobile application to a topic.

```

/*! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                        const Aws::String &endpointARN,
                        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

```



```

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
        << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

Subscribe a Lambda function to a topic.

```

/*! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to an AWS Lambda function.
*/
\param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
\param lambdaFunctionARN: The ARN for an AWS Lambda function.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                const Aws::String &lambdaFunctionARN,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");

```

```

request.SetEndpoint(lambdaFunctionARN);

const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    std::cout << "Subscribed successfully." << std::endl;
    std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
    << "'." << std::endl;
}
else {
    std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
    << std::endl;
}

return outcome.IsSuccess();
}

```

Subscribe an SQS queue to a topic.

```

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

Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);

    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
    << "' has been subscribed to the topic '"
    << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "'."
    << std::endl;
    }
}

```

```

        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }

```

Subscribe with a filter to a topic.

```

static const Aws::String TONE_ATTRIBUTE("tone");
static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                "sincere"};

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

Aws::SNS::SNSClient snsClient(clientConfig);

Aws::SNS::Model::SubscribeRequest request;
request.SetTopicArn(topicARN);
request.SetProtocol("sqs");
request.SetEndpoint(queueARN);
if (isFifoTopic) {
    if (first) {
        std::cout << "Subscriptions to a FIFO topic can have filters."
                  << std::endl;
        std::cout
            << "If you add a filter to this subscription, then only
the filtered messages "
            << "will be received in the queue." << std::endl;
        std::cout << "For information about message filtering, "

```

```

        << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
        << std::endl;
        std::cout << "For this example, you can filter messages by a \""
        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
    }

    std::ostringstream ostream;
    ostream << "Filter messages for \"" << queueName
        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    }
    // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
            << "' has been subscribed to the topic '"
            << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "."

```

```

        << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }

    /** Routine that lets the user select attributes for a subscription filter policy.
    */
    \sa getFilterPolicyFromUser()
    \return Aws::String: The filter policy as JSON.
    */
    Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
        std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

        std::vector<Aws::String> filterSelections;
        int selection;
        do {
            for (size_t j = 0; j < TONES.size(); ++j) {
                std::cout << " " << (j + 1) << ". " << TONES[j]
                    << std::endl;
            }
            selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

            if (selection != 0) {
                const Aws::String &selectedTone(TONES[selection - 1]);
                // Add the tone to the selection if it is not already added.
                if (std::find(filterSelections.begin(),
                    filterSelections.end(),
                    selectedTone)

```

```

        == filterSelections.end()) {
            filterSelections.push_back(selectedTone);
        }
    }
} while (selection != 0);

Aws::String result;
if (!filterSelections.empty()) {
    std::ostringstream jsonPolicyStream;
    jsonPolicyStream << "{ \"\" << TONE_ATTRIBUTE << "\": [";

    for (size_t j = 0; j < filterSelections.size(); ++j) {
        jsonPolicyStream << "\"" << filterSelections[j] << "\"";
        if (j < filterSelections.size() - 1) {
            jsonPolicyStream << ",";
        }
    }
    jsonPolicyStream << "] }";

    result = jsonPolicyStream.str();
}

return result;
}

```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

Unsubscribe

The following code example shows how to use Unsubscribe.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
topic.
/*!
 \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [Unsubscribe](#) in *AWS SDK for C++ API Reference*.

Scenarios

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an SMS
 * text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 * NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is
 * in the SMS sandbox and you can only
 * use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
 * latest/dg/sns-sms-sandbox.html.
 * NOTE: If destination is in the US, you also have an additional restriction that
 * you have use a dedicated
 * origination ID (phone number). You can request an origination number using
 * Amazon Pinpoint for a fee.
 * See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
 * origination-numbers-with-amazon-sns/
 * for more information.
 *
 * <phone_number_value> input parameter uses E.164 format.
 * For example, in United States, this input value should be of the form:
 * +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
 \param message: The message to publish.
 \param phoneNumber: The phone number of the recipient in E.164 format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
```



```
Aws::SNS::Model::PublishRequest request;
request.SetMessage(message);
request.SetPhoneNumber(phoneNumber);

const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

if (outcome.IsSuccess()) {
    std::cout << "Message published successfully with message id, '"
                << outcome.GetResult().GetMessageId() << "'."
                << std::endl;
}
else {
    std::cerr << "Error while publishing message "
                << outcome.GetError().GetMessage()
                << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [Publish](#) in *AWS SDK for C++ API Reference*.

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.
- Poll the queues for messages received.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

    //! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
    /*!
    \param clientConfig Aws client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
        std::cout << "Welcome to messaging with topics and queues." << std::endl;
        printAsterisksLine();
        std::cout << "In this workflow, you will create an SNS topic and subscribe "
            << NUMBER_OF_QUEUES <<
            " SQS queues to the topic." << std::endl;
        std::cout
            << "You can select from several options for configuring the topic and
the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
        std::cout << "You can then post to the topic and see the results in the queues."
            << std::endl;

        Aws::SNS::SNSClient snsClient(clientConfiguration);

        printAsterisksLine();

        std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
            << std::endl;
        std::cout
            << "FIFO topics deliver messages in order and support deduplication and
message filtering."
            << std::endl;
        bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

        bool contentBasedDeduplication = false;
        Aws::String topicName;
        if (isFifoTopic) {
            printAsterisksLine();
            std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                << std::endl;
        }
    }
}
```

```

        std::cout
            << "Deduplication IDs are either set in the message or automatically
generated "
            << "from content using a hash function." << std::endl;
        std::cout
            << "If a message is successfully published to an SNS FIFO topic, any
message "
            << "published and determined to have the same deduplication ID, "
            << std::endl;
        std::cout
            << "within the five-minute deduplication interval, is accepted but
not delivered."
            << std::endl;
        std::cout
            << "For more information about deduplication, "
            << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
            << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
            "Use content-based deduplication instead of entering a deduplication
ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout

```

```
        << "Because you have selected a FIFO topic, '.fifo' must be
        appended to the topic name."
        << std::endl;
    }

    request.SetName(topicName);

    Aws::SNS::Model::CreateTopicOutcome outcome =
    snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARN = outcome.GetResult().GetTopicArn();
        std::cout << "Your new topic with the name '" << topicName
            << "' and the topic Amazon Resource Name (ARN) " << std::endl;
        std::cout << "'" << topicARN << "' has been created." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::CreateTopic. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

printAsterisksLine();

std::cout << "Now you will create " << NUMBER_OF_QUEUES
    << " SQS queues to subscribe to the topic." << std::endl;
Aws::Vector<Aws::String> queueNames;
bool filteringMessages = false;
bool first = true;
for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
    Aws::String queueURL;
    Aws::String queueName;
    {
        printAsterisksLine();
```

```

std::ostringstream ostream;
ostream << "Enter a name for " << (first ? "an" : "the next")
        << " SQS queue. ";
queueName = askQuestion(ostream.str());

// 2. Create an SQS queue.
Aws::SQS::Model::CreateQueueRequest request;
if (isFifoTopic) {
request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                    "true");
    queueName = queueName + FIFO_SUFFIX;

    if (first) // Only explain this once.
    {
        std::cout
            << "Because you are creating a FIFO SQS queue, '.fifo'
must "
            << "be appended to the queue name." << std::endl;
    }
}

request.SetQueueName(queueName);
queueNames.push_back(queueName);

Aws::SQS::Model::CreateQueueOutcome outcome =
    sqsClient.CreateQueue(request);

if (outcome.IsSuccess()) {
    queueURL = outcome.GetResult().GetQueueUrl();
    std::cout << "Your new SQS queue with the name '" << queueName
        << "' and the queue URL " << std::endl;
    std::cout << "'" << queueURL << "' has been created." << std::endl;
}
else {
    std::cerr << "Error with SQS::CreateQueue. "
        << outcome.GetError().GetMessage()
        << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);
}

```

```
        return false;
    }
}
queueURLS.push_back(queueURL);

if (first) // Only explain this once.
{
    std::cout
        << "The queue URL is used to retrieve the queue ARN, which is "
        << "used to create a subscription." << std::endl;
}

Aws::String queueARN;
{
    // 3. Get the SQS queue ARN attribute.
    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN
                << "' has been retrieved."
                << std::endl;
        }
    }
    else {
        std::cerr
            << "Error ARN attribute not returned by
GetQueueAttribute."
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
```

```
        subscriptionARNS,
        snsClient,
        sqsClient);

    return false;
}
}
else {
    std::cerr << "Error with SQS::GetQueueAttributes. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}

if (first) {
    std::cout
        << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
        << "messages from an SNS topic." << std::endl;
}

{
    // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    Aws::String policy = createPolicyForQueue(queueARN, topicARN);
    request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                        policy);

    Aws::SQS::Model::SetQueueAttributesOutcome outcome =
        sqsClient.SetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        std::cout << "The attributes for the queue '" << queueName
                << "' were successfully updated." << std::endl;
    }
}
```

```

    }
    else {
        std::cerr << "Error with SQS::SetQueueAttributes. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNs,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

{
    // 5. Subscribe the SQS queue to the SNS topic.
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have filters."
                      << std::endl;

            std::cout
                << "If you add a filter to this subscription, then only
the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                << std::endl;
            std::cout << "For this example, you can filter messages by a \""
                << TONE_ATTRIBUTE << "\" attribute." << std::endl;
        }

        std::ostringstream ostream;
        ostream << "Filter messages for \"" << queueName
                << "\"'s subscription to the topic \""
                << topicName << "\"? (y/n)";
    }
}

```



```

// Add filter if user answers yes.
if (askYesNoQuestion(ostringstream.str())) {
    Aws::String jsonPolicy = getFilterPolicyFromUser();
    if (!jsonPolicy.empty()) {
        filteringMessages = true;

        std::cout << "This is the filter policy for this
subscription."
                    << std::endl;
        std::cout << jsonPolicy << std::endl;

        request.AddAttributes("FilterPolicy", jsonPolicy);
    }
    else {
        std::cout
            << "Because you did not select any attributes, no
filter "
            << "will be added to this subscription." <<
std::endl;
    }
} // if (isFifoTopic)
Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
    std::cout << "The queue '" << queueName
                << "' has been subscribed to the topic '"
                << "'" << topicName << "'" << std::endl;
    std::cout << "with the subscription ARN '" << subscriptionARN << "."
                << std::endl;
    subscriptionARNS.push_back(subscriptionARN);
}
else {
    std::cerr << "Error with TopicsAndQueues::Subscribe. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,

```

```

        snsClient,
        sqsClient);

    return false;
}
}

first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received in
the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupID = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupID);
        if (!contentBasedDeduplication) {
            if (first) {
                std::cout
                    << "Because you are not using content-based
deduplication, "
                    << "you must enter a deduplication ID." << std::endl;
            }
            Aws::String deduplicationID = askQuestion(
                "Enter a deduplication ID for this message. ");
            request.SetMessageDeduplicationId(deduplicationID);
        }
    }
}
}

```

```
    }

    if (filteringMessages && askYesNoQuestion(
        "Add an attribute to this message? (y/n) ")) {
        for (size_t i = 0; i < TONES.size(); ++i) {
            std::cout << " " << (i + 1) << ". " << TONES[i] << std::endl;
        }
        int selection = askQuestionForIntRange(
            "Enter a number for an attribute. ",
            1, static_cast<int>(TONES.size()));
        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

    Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Your message was successfully published." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Publish. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }

    first = false;
} while (askYesNoQuestion("Post another message? (y/n) "));

printAsterisksLine();

std::cout << "Now the SQS queue will be polled to retrieve the messages."
    << std::endl;
askQuestion("Press any key to continue...", alwaysTrueTest);
```

```
for (size_t i = 0; i < queueURLS.size(); ++i) {
    // 7. Poll an SQS queue for its messages.
    std::vector<Aws::String> messages;
    std::vector<Aws::String> receiptHandles;
    while (true) {
        Aws::SQS::Model::ReceiveMessageRequest request;
        request.SetMaxNumberOfMessages(10);
        request.SetQueueUrl(queueURLS[i]);

        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        request.SetWaitTimeSeconds(1);
        Aws::SQS::Model::ReceiveMessageOutcome outcome =
            sqsClient.ReceiveMessage(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
            if (newMessages.empty()) {
                break;
            }
            else {
                for (const Aws::SQS::Model::Message &message: newMessages) {
                    messages.push_back(message.GetBody());
                    receiptHandles.push_back(message.GetReceiptHandle());
                }
            }
        }
        else {
            std::cerr << "Error with SQS::ReceiveMessage. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }
    }
}
```

```
printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
    << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << " Message : '" << message << "'."
        << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {
    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);

    if (outcome.IsSuccess()) {
        std::cout << "The batch deletion of messages was successful."
            << std::endl;
    }
    else {
        std::cerr << "Error with SQS::DeleteMessageBatch. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanUp(topicARN,
```

```

        queueURLS,
        subscriptionARNs,
        snsClient,
        sqsClient);

    return false;
}
}
}

return cleanUp(topicARN,
               queueURLS,
               subscriptionARNs,
               snsClient,
               sqsClient,
               true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                       const Aws::Vector<Aws::String> &queueURLS,
                                       const Aws::Vector<Aws::String>
                                       &subscriptionARNs,
                                       const Aws::SNS::SNSClient &snsClient,
                                       const Aws::SQS::SQSClient &sqsClient,
                                       bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {
        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                    << "' was successfully deleted." << std::endl;
            }
            else {

```

```

        std::cerr << "Error with SQS::DeleteQueue. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}

for (const auto &subscriptionARN: subscriptionARNS) {
    // 10. Unsubscribe an SNS subscription.
    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    Aws::SNS::Model::UnsubscribeOutcome outcome =
        snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                  << "' was successful." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
                  << "' was successfully deleted." << std::endl;
    }
    else {

```

```

        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages from
    an SNS topic.
/*!
    \sa createPolicyForQueue()
    \param queueARN: The SQS queue Amazon Resource Name (ARN).
    \param topicARN: The SNS topic ARN.
    \return Aws::String: The policy as JSON.
    */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                            const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": ")" << topicARN << R"("
                    }
                }
            }
        ]
    })";

    return policyStream.str();
}

```


- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Amazon SQS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon SQS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SQS

The following code examples show how to get started using Amazon SQS.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_sqs.cpp)

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

Code for the hello_sqs.cpp source file.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 * A "Hello SQS" starter application that initializes an Amazon Simple Queue
 * Service
 * (Amazon SQS) client and lists the SQS queues in the current account.
 *
 * main function
 *
 * Usage: 'hello_sqs'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
```

```

    Aws::String nextToken; // Next token is used to handle a paginated response.
    do {
        Aws::SQS::Model::ListQueuesRequest request;

        Aws::SQS::Model::ListQueuesOutcome outcome =
sqsClient.ListQueues(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &pageOfQueueUrls =
outcome.GetResult().GetQueueUrls();
            if (!pageOfQueueUrls.empty()) {
                allQueueUrls.insert(allQueueUrls.cend(),
pageOfQueueUrls.cbegin(),
                                pageOfQueueUrls.cend());
            }
        }
        else {
            std::cerr << "Error with SQS::ListQueues. "
                << outcome.GetError().GetMessage()
                << std::endl;
            break;
        }
        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
queue"
                << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
                << std::endl;

    if (!allQueueUrls.empty()) {
        std::cout << "Here are your queue URLs." << std::endl;
        for (const Aws::String &queueUrl: allQueueUrls) {
            std::cout << " * " << queueUrl << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- For API details, see [ListQueues](#) in *AWS SDK for C++ API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

ChangeMessageVisibility

The following code example shows how to use `ChangeMessageVisibility`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Changes the visibility timeout of a message in an Amazon Simple Queue Service
//! (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param visibilityTimeoutSeconds: Visibility timeout in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::changeMessageVisibility(
    const Aws::String &queue_url,
    const Aws::String &messageReceiptHandle,
    int visibilityTimeoutSeconds,
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```

Aws::SQS::Model::ChangeMessageVisibilityRequest request;
request.SetQueueUrl(queue_url);
request.SetReceiptHandle(messageReceiptHandle);
request.SetVisibilityTimeout(visibilityTimeoutSeconds);

auto outcome = sqsClient.ChangeMessageVisibility(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully changed visibility of message " <<
        messageReceiptHandle << " from queue " << queue_url << std::endl;
}
else {
    std::cout << "Error changing visibility of message from queue "
        << queue_url << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}

```

- For API details, see [ChangeMessageVisibility](#) in *AWS SDK for C++ API Reference*.

CreateQueue

The following code example shows how to use CreateQueue.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

//! Create an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
\param queueName: An Amazon SQS queue name.
\param clientConfiguration: AWS client configuration.

```

```
\return bool: Function succeeded.
*/
bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::CreateQueueRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::CreateQueueOutcome outcome =
sqsClient.CreateQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created queue " << queueName << " with a queue
URL "
                << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for C++ API Reference*.

DeleteMessage

The following code example shows how to use DeleteMessage.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

```

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

    /*! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
    */
    \param queueUrl: An Amazon SQS queue URL.
    \param messageReceiptHandle: A message receipt handle.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
            << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteMessage](#) in *AWS SDK for C++ API Reference*.

DeleteMessageBatch

The following code example shows how to use DeleteMessageBatch.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);

    if (outcome.IsSuccess()) {
        std::cout << "The batch deletion of messages was successful."
                  << std::endl;
    }
    else {
        std::cerr << "Error with SQS::DeleteMessageBatch. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUp(topicARN,
               queueURLS,
               subscriptionARNS,
               snsClient,
               sqsClient);
```

```
        return false;
    }
```

- For API details, see [DeleteMessageBatch](#) in *AWS SDK for C++ API Reference*.

DeleteQueue

The following code example shows how to use DeleteQueue.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
 \param queueURL: An Amazon SQS queue URL.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::SQS::Model::DeleteQueueRequest request;
    request.SetQueueUrl(queueURL);

    const Aws::SQS::Model::DeleteQueueOutcome outcome =
sqsClient.DeleteQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted queue with url " << queueURL <<
        std::endl;
    }
}
```

```

else {
    std::cerr << "Error deleting queue " << queueURL << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

```

- For API details, see [DeleteQueue](#) in *AWS SDK for C++ API Reference*.

GetQueueAttributes

The following code example shows how to use `GetQueueAttributes`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

Aws::SQS::SQSClient sqsClient(clientConfig);

    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);

```

```

        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN
                << "' has been retrieved."
                << std::endl;
        }
    }
    else {
        std::cerr << "Error with SQS::GetQueueAttributes. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

```

- For API details, see [GetQueueAttributes](#) in *AWS SDK for C++ API Reference*.

GetQueueUrl

The following code example shows how to use `GetQueueUrl`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    /*! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
    */
    \param queueName: An Amazon SQS queue name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,

```

```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
            outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetQueueUrl](#) in *AWS SDK for C++ API Reference*.

ListQueues

The following code example shows how to use ListQueues.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    /*! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS account.
    /*!

```

```
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool
AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration &clientConfiguration)
{
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
            listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}
```

- For API details, see [ListQueues](#) in *AWS SDK for C++ API Reference*.

ReceiveMessage

The following code example shows how to use `ReceiveMessage`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
 \param queueUrl: An Amazon SQS queue URL.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);

    const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
        request);
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
            outcome.GetResult().GetMessages();
        if (!messages.empty()) {
```

```

        const Aws::SQS::Model::Message &message = messages[0];
        std::cout << "Received message:" << std::endl;
        std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
        std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
        std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
    }
    else {
        std::cout << "No messages received from queue " << queueUrl <<
            std::endl;

    }
}
else {
    std::cerr << "Error receiving message from queue " << queueUrl << ": "
        << outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

```

- For API details, see [ReceiveMessage](#) in *AWS SDK for C++ API Reference*.

SendMessage

The following code example shows how to use `SendMessage`.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    //! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.

```



```

    \param messageBody: A message body.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                             const Aws::String &messageBody,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
            std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [SendMessage](#) in *AWS SDK for C++ API Reference*.

SetQueueAttributes

The following code example shows how to use SetQueueAttributes.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

    //! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
    queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param attributeName: An attribute name enum.
    \param attribute: The attribute value as a string.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                        Aws::SQS::Model::QueueAttributeName
    attributeName,
                                        const Aws::String &attribute,
                                        const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(queueURL);
        request.AddAttributes(
            attributeName,
            attribute);

        const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
        sqsClient.SetQueueAttributes(
            request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully set the attribute " <<

        Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
            attributeName)
            << " with value " << attribute << " in queue " <<
            queueURL << "." << std::endl;
        }
        else {
            std::cout << "Error setting attribute for queue " <<
            queueURL << ": " << outcome.GetError().GetMessage() <<
            std::endl;
        }
    }

```

```

    return outcome.IsSuccess();
}

```

Configure a dead-letter queue.

```

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

    //! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
    //! dead-letter queue.
    /*!
    \param srcQueueUrl: An Amazon SQS queue URL.
    \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS dead-
    letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                         const Aws::String &deadLetterQueueARN,
                                         int maxReceiveCount,
                                         const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
    maxReceiveCount);

        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(srcQueueUrl);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

        const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully set dead letter queue for queue " <<
                srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
        }
    }

```

```

    }
    else {
        std::cerr << "Error setting dead letter queue for queue " <<
            srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
            std::endl;
    }

    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
    \param queueArn: An Amazon SQS ARN for the dead-letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \return Aws::String: Policy as JSON string.
    */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}

```

Configure an Amazon SQS queue to use long polling.

```

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

    //! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param pollTimeSeconds: The receive message wait time in seconds.
    \param clientConfiguration: AWS client configuration.
    */

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                               const Aws::String &pollTimeSeconds,
                                               const
    Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
        Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
        pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
            queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
            queueURL << ": " << outcome.GetError().GetMessage() <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [SetQueueAttributes](#) in *AWS SDK for C++ API Reference*.

Scenarios

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.

- Poll the queues for messages received.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
        << NUMBER_OF_QUEUES <<
        " SQS queues to the topic." << std::endl;
    std::cout
        << "You can select from several options for configuring the topic and
the subscriptions for the "
        << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the queues."
        << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
        << std::endl;
    std::cout
        << "FIFO topics deliver messages in order and support deduplication and
message filtering."
```

```
        << std::endl;
    bool isFifoTopic = askYesNoQuestion(
        "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
            << std::endl;
        std::cout
            << "Deduplication IDs are either set in the message or automatically
generated "
            << "from content using a hash function." << std::endl;
        std::cout
            << "If a message is successfully published to an SNS FIFO topic, any
message "
            << "published and determined to have the same deduplication ID, "
            << std::endl;
        std::cout
            << "within the five-minute deduplication interval, is accepted but
not delivered."
            << std::endl;
        std::cout
            << "For more information about deduplication, "
            << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
            << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
            "Use content-based deduplication instead of entering a deduplication
ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNs;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");
```

```
// 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
Aws::SNS::Model::CreateTopicRequest request;

if (isFifoTopic) {
    request.AddAttributes("FifoTopic", "true");
    if (contentBasedDeduplication) {
        request.AddAttributes("ContentBasedDeduplication", "true");
    }
    topicName = topicName + FIFO_SUFFIX;

    std::cout
        << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
        << std::endl;
}

request.SetName(topicName);

Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

if (outcome.IsSuccess()) {
    topicARN = outcome.GetResult().GetTopicArn();
    std::cout << "Your new topic with the name '" << topicName
        << "' and the topic Amazon Resource Name (ARN) " << std::endl;
    std::cout << "'" << topicARN << "' has been created." << std::endl;
}
else {
    std::cerr << "Error with TopicsAndQueues::CreateTopic. "
        << outcome.GetError().GetMessage()
        << std::endl;

    cleanUp(topicARN,
        queueURLS,
        subscriptionARNS,
        snsClient,
        sqsClient);

    return false;
}
}

printAsterisksLine();
```



```

std::cout << "Now you will create " << NUMBER_OF_QUEUES
           << " SQS queues to subscribe to the topic." << std::endl;
Aws::Vector<Aws::String> queueNames;
bool filteringMessages = false;
bool first = true;
for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
    Aws::String queueURL;
    Aws::String queueName;
    {
        printAsterisksLine();
        std::ostringstream ostream;
        ostream << "Enter a name for " << (first ? "an" : "the next")
                << " SQS queue. ";
        queueName = askQuestion(ostream.str());

        // 2. Create an SQS queue.
        Aws::SQS::Model::CreateQueueRequest request;
        if (isFifoTopic) {
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                "true");
            queueName = queueName + FIFO_SUFFIX;

            if (first) // Only explain this once.
            {
                std::cout
                    << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                    << "be appended to the queue name." << std::endl;
            }
        }

        request.SetQueueName(queueName);
        queueNames.push_back(queueName);

        Aws::SQS::Model::CreateQueueOutcome outcome =
            sqsClient.CreateQueue(request);

        if (outcome.IsSuccess()) {
            queueURL = outcome.GetResult().GetQueueUrl();
            std::cout << "Your new SQS queue with the name '" << queueName
                    << "' and the queue URL " << std::endl;
            std::cout << "'" << queueURL << "' has been created." << std::endl;
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error with SQS::CreateQueue. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}
queueURLS.push_back(queueURL);

if (first) // Only explain this once.
{
    std::cout
        << "The queue URL is used to retrieve the queue ARN, which is "
        << "used to create a subscription." << std::endl;
}

Aws::String queueARN;
{
    // 3. Get the SQS queue ARN attribute.
    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN

```

```
        << "" has been retrieved."
        << std::endl;
    }
    else {
        std::cerr
            << "Error ARN attribute not returned by
GetQueueAttribute."
            << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}
else {
    std::cerr << "Error with SQS::GetQueueAttributes. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}

if (first) {
    std::cout
        << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
        << "messages from an SNS topic." << std::endl;
}

{
    // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
    Aws::SQS::Model::SetQueueAttributesRequest request;
```

```
request.SetQueueUrl(queueURL);
Aws::String policy = createPolicyForQueue(queueARN, topicARN);
request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                    policy);

Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(request);

if (outcome.IsSuccess()) {
    std::cout << "The attributes for the queue '" << queueName
              << "' were successfully updated." << std::endl;
}
else {
    std::cerr << "Error with SQS::SetQueueAttributes. "
              << outcome.GetError().GetMessage()
              << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}

printAsterisksLine();

{
    // 5. Subscribe the SQS queue to the SNS topic.
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have filters."
                    << std::endl;

            std::cout
                << "If you add a filter to this subscription, then only
the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
```

```

        << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
        << std::endl;
        std::cout << "For this example, you can filter messages by a \""
        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
    }

    std::ostringstream ostream;
    ostream << "Filter messages for \"" << queueName
        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    }
    // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
            << "' has been subscribed to the topic '"
            << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "."

```

```
        << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received in
the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupID = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupID);
```

```

        if (!contentBasedDeduplication) {
            if (first) {
                std::cout
                    << "Because you are not using content-based
deduplication, "
                    << "you must enter a deduplication ID." << std::endl;
            }
            Aws::String deduplicationID = askQuestion(
                "Enter a deduplication ID for this message. ");
            request.SetMessageDeduplicationId(deduplicationID);
        }
    }

    if (filteringMessages && askYesNoQuestion(
        "Add an attribute to this message? (y/n) ")) {
        for (size_t i = 0; i < TONES.size(); ++i) {
            std::cout << " " << (i + 1) << ". " << TONES[i] << std::endl;
        }
        int selection = askQuestionForIntRange(
            "Enter a number for an attribute. ",
            1, static_cast<int>(TONES.size()));
        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

    Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Your message was successfully published." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Publish. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

```

```

    }

    first = false;
} while (askYesNoQuestion("Post another message? (y/n) "));

printAsterisksLine();

std::cout << "Now the SQS queue will be polled to retrieve the messages."
    << std::endl;
askQuestion("Press any key to continue...", alwaysTrueTest);

for (size_t i = 0; i < queueURLS.size(); ++i) {
    // 7. Poll an SQS queue for its messages.
    std::vector<Aws::String> messages;
    std::vector<Aws::String> receiptHandles;
    while (true) {
        Aws::SQS::Model::ReceiveMessageRequest request;
        request.SetMaxNumberOfMessages(10);
        request.SetQueueUrl(queueURLS[i]);

        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        request.SetWaitTimeSeconds(1);
        Aws::SQS::Model::ReceiveMessageOutcome outcome =
            sqsClient.ReceiveMessage(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
            if (newMessages.empty()) {
                break;
            }
            else {
                for (const Aws::SQS::Model::Message &message: newMessages) {
                    messages.push_back(message.GetBody());
                    receiptHandles.push_back(message.GetReceiptHandle());
                }
            }
        }
        else {
            std::cerr << "Error with SQS::ReceiveMessage. "
                << outcome.GetError().GetMessage()

```



```

        << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
            << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << "  Message : '" << message << "'."
                << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {
    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =

```

```

        sqsClient.DeleteMessageBatch(request);

        if (outcome.IsSuccess()) {
            std::cout << "The batch deletion of messages was successful."
                << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteMessageBatch. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }
    }
}

return cleanUp(topicARN,
    queueURLS,
    subscriptionARNS,
    snsClient,
    sqsClient,
    true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
    const Aws::Vector<Aws::String> &queueURLS,
    const Aws::Vector<Aws::String>
    &subscriptionARNS,
    const Aws::SNS::SNSClient &snsClient,
    const Aws::SQS::SQSClient &sqsClient,
    bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.

```

```
Aws::SQS::Model::DeleteQueueRequest request;
request.SetQueueUrl(queueURL);

Aws::SQS::Model::DeleteQueueOutcome outcome =
    sqsClient.DeleteQueue(request);

if (outcome.IsSuccess()) {
    std::cout << "The queue with URL '" << queueURL
                << "' was successfully deleted." << std::endl;
}
else {
    std::cerr << "Error with SQS::DeleteQueue. "
                << outcome.GetError().GetMessage()
                << std::endl;
    result = false;
}
}

for (const auto &subscriptionARN: subscriptionARNS) {
    // 10. Unsubscribe an SNS subscription.
    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    Aws::SNS::Model::UnsubscribeOutcome outcome =
        snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                    << "' was successful." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        result = false;
    }
}
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
```

```

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
            << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

/*! Create an IAM policy that gives an SQS queue permission to receive messages from
an SNS topic.
*/
\sa createPolicyForQueue()
\param queueARN: The SQS queue Amazon Resource Name (ARN).
\param topicARN: The SNS topic ARN.
\return Aws::String: The policy as JSON.
*/
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                    const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {

```

```
        "ArnEquals": {
            "aws:SourceArn": ")" << topicARN << R "("
        }
    }
}
]
}));

return policyStream.str();
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

AWS STS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS STS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

AssumeRole

The following code example shows how to use AssumeRole.

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
    }
}
```

```
    const Aws::STS::Model::Credentials &temp_credentials =
result.GetCredentials();

    // Store temporary credentials in return argument.
    // Note: The credentials object returned by assumeRole differs
    // from the AWSCredentials object used in most situations.
    credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
    credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
    credentials.SetSessionToken(temp_credentials.GetSessionToken());
}

return outcome.IsSuccess();
}
```

- For API details, see [AssumeRole](#) in *AWS SDK for C++ API Reference*.

Amazon Transcribe examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Transcribe.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

StartStreamTranscriptionAsync

The following code example shows how to use `StartStreamTranscriptionAsync`.

SDK for C++**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
        managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
        SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
        windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
            [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                std::cerr << "ERROR: " + error.GetMessage() << std::endl;
            });
        //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
```



```

        for (auto &&r: ev.GetTranscript().GetResults()) {
            if (r.GetIsPartial()) {
                std::cout << "[partial] ";
            }
            else {
                std::cout << "[Final] ";
            }
            for (auto &&alt: r.GetAlternatives()) {
                std::cout << alt.GetTranscript() << std::endl;
            }
        }
    });

    StartStreamTranscriptionRequest request;
    request.SetMediaSampleRateHertz(SAMPLE_RATE);
    request.SetLanguageCode(LanguageCode::en_US);
    request.SetMediaEncoding(
        MediaEncoding::pcm); // wav and aiff files are PCM formats.
    request.SetEventStreamHandler(handler);

    auto OnStreamReady = [](AudioStream &stream) {
        Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
        if (!file.is_open()) {
            std::cerr << "Failed to open " << FILE_NAME << '\n';
        }
        std::array<char, BUFFER_SIZE> buf;
        int i = 0;
        while (file) {
            file.read(&buf[0], buf.size());

            if (!file)
                std::cout << "File: only " << file.gcount() << " could be
read"
                    << std::endl;

            Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
            AudioEvent event(std::move(bits));
            if (!stream) {
                std::cerr << "Failed to create a stream" << std::endl;
                break;
            }
            //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns

```

```

        //the number of characters extracted by the last read()
operation.
        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" <<
std::endl;
                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
            25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {
        // Per the spec, we have to send an empty event (an event
without a payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    stream.Close();
};

    Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
    auto OnResponseCallback = [&signaling](
        const TranscribeStreamingServiceClient * /*unused*/,
        const Model::StartStreamTranscriptionRequest & /*unused*/,
        const Model::StartStreamTranscriptionOutcome &outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

        if (!outcome.IsSuccess()) {
            std::cerr << "Transcribe streaming error "
                << outcome.GetError().GetMessage() << std::endl;
        }

        signaling.Release();
};

```

```
        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
        OnResponseCallback,
                                     nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- For API details, see [StartStreamTranscriptionAsync](#) in *AWS SDK for C++ API Reference*.

Cross-service examples using SDK for C++

The following sample applications use the AWS SDK for C++ to work across multiple AWS services.

Cross-service examples target an advanced level of experience to help you start building applications.

Examples

- [Create a photo asset management application that lets users manage photos using labels](#)
- [Create an Aurora Serverless work item tracker](#)

Create a photo asset management application that lets users manage photos using labels

SDK for C++

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [AWS Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Create an Aurora Serverless work item tracker

SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Security for AWS SDK for C++

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The [Shared Responsibility Model](#) describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS Compliance Programs](#).

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization’s requirements, and applicable laws and regulations.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Topics

- [Data Protection in AWS SDK for C++](#)
- [Identity and Access Management](#)
- [Compliance Validation for this AWS Product or Service](#)
- [Resilience for this AWS Product or Service](#)
- [Infrastructure Security for this AWS Product or Service](#)
- [Enforcing a minimum TLS version in the AWS SDK for C++](#)
- [Amazon S3 Encryption Client Migration](#)

Data Protection in AWS SDK for C++

The AWS [shared responsibility model](#) applies to data protection in AWS SDK for C++. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this

infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with SDK for C++ or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Identity and Access Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)

- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How AWS services work with IAM](#)
- [Troubleshooting AWS identity and access](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS.

Service user – If you use AWS services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS, see [Troubleshooting AWS identity and access](#) or the user guide of the AWS service you are using.

Service administrator – If you're in charge of AWS resources at your company, you probably have full access to AWS. It's your job to determine which AWS features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS, see the user guide of the AWS service you are using.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS. To view example AWS identity-based policies that you can use in IAM, see the user guide of the AWS service you are using.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [Signing AWS API requests](#) in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Multi-factor authentication](#) in the *AWS IAM Identity Center User Guide* and [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For

information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](#) in the *AWS IAM Identity Center User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
 - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).
 - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
 - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are *IAM role trust policies* and *Amazon S3 bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to

any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How AWS services work with IAM

To get a high-level view of how AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

To learn how to use a specific AWS service with IAM, see the security section of the relevant service's User Guide.

Troubleshooting AWS identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS and IAM.

Topics

- [I am not authorized to perform an action in AWS](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my AWS resources](#)

I am not authorized to perform an action in AWS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but doesn't have the fictional `awes:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awes:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the `my-example-widget` resource by using the `awes:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS supports these features, see [How AWS services work with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Compliance Validation for this AWS Product or Service

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

Note

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Resilience for this AWS Product or Service

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Infrastructure Security for this AWS Product or Service

This AWS product or service uses managed services, and therefore is protected by the AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access this AWS Product or Service through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Enforcing a minimum TLS version in the AWS SDK for C++

To increase security when communicating with AWS services, you should configure SDK for C++ to use TLS 1.2 or later. We recommend using TLS 1.3.

The AWS SDK for C++ is a cross-platform library. You can build and run your application on the platforms you want. Different platforms might depend on different underlying HTTP clients.

By default, macOS, Linux, Android and other non-Windows platforms use [libcurl](#). If the libcurl version is later than 7.34.0, TLS 1.0 is the minimum version used by the underlying HTTP clients.

For Windows, the default library is [WinHttp](#). Windows decides the actual protocol to use among the TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3 protocols available. [WinINet](#) and [IXMLHttpRequest2](#) are the other two options that are available on Windows. You can configure your application to replace the default library during CMake and at runtime. For these two HTTP clients, Windows also decides the secure protocol.

The AWS SDK for C++ also provides the flexibility to override the default HTTP clients. For example, you can enforce libcurl or use whatever HTTP clients you want by using a custom HTTP client factory. So to use TLS 1.2 as the minimum version, you must be aware of the HTTP client library you're using.

Enforce specific TLS version with libcurl on all platforms

This section assumes that the AWS SDK for C++ is using libcurl as a dependency for HTTP protocol support. To explicitly specify the TLS version, you will need a minimum libcurl version of 7.34.0. In addition, you might need to modify the source code of the AWS SDK for C++ and then rebuild it.

The following procedure shows you how to perform these tasks.

To enforce TLS 1.2 with libcurl

1. Verify that your installation of libcurl is at least version 7.34.0.
2. Download the source code for the AWS SDK for C++ from [GitHub](#).
3. Open `aws-cpp-sdk-core/source/http/curl/CurlHttpClient.cpp` and find the following lines of code.

```
#if LIBCURL_VERSION_MAJOR >= 7
#if LIBCURL_VERSION_MINOR >= 34
curl_easy_setopt(connectionHandle, CURLOPT_SSLVERSION, CURL_SSLVERSION_TLSv1);
#endif //LIBCURL_VERSION_MINOR
#endif //LIBCURL_VERSION_MAJOR
```

4. If necessary, change the last parameter in the function call as follows.

```
#if LIBCURL_VERSION_MAJOR >= 7
#if LIBCURL_VERSION_MINOR >= 34
curl_easy_setopt(connectionHandle, CURLOPT_SSLVERSION, CURL_SSLVERSION_TLSv1_2);
#endif //LIBCURL_VERSION_MINOR
#endif //LIBCURL_VERSION_MAJOR
```

5. If you performed the preceding code changes, build and install the AWS SDK for C++ according to the instructions at <https://github.com/aws/aws-sdk-cpp#building-the-sdk>.
6. For the service client in your application, enable `verifySSL` in its client configuration, if this option isn't already enabled.

To enforce TLS 1.3 with libcurl

To enforce TLS 1.3, follow the steps in the preceding section setting the `CURL_SSLVERSION_TLSv1_3` option instead of `CURL_SSLVERSION_TLSv1_2`.

Enforce specific TLS version on Windows

The following procedures show you how to enforce TLS 1.2 or TLS 1.3 with WinHttp, WinINet, or XMLHttpRequest2.

Prerequisite: Determine Windows TLS support

- Determine the TLS protocol version support available for your system as described at <https://docs.microsoft.com/en-us/windows/win32/secauthn/protocols-in-tls-ssl-schannel-ssp->.
- If you're running on Windows 7 SP1 or Windows Server 2008 R2 SP1, you need to ensure that TLS 1.2 support is enabled in the registry, as described at <https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings#tls-12>. If you're running an earlier distribution, you must upgrade your operating system.

To enforce TLS 1.2 or TLS 1.3 with WinHttp

WinHttp provides an API to explicitly set the acceptable secure protocols. However, to make this configurable at runtime, you need to modify the source code of the AWS SDK for C++ and then rebuild it.

1. Download the source code for the AWS SDK for C++ from [GitHub](https://github.com).

2. Open `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` and find the following lines of code.

```
#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)
    DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_1 |
        WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
#else
    DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_1 | WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
#endif

if (!WinHttpSetOption(GetOpenHandle(), WINHTTP_OPTION_SECURE_PROTOCOLS, &flags,
    sizeof(flags)))
{
    AWS_LOGSTREAM_FATAL(GetLogTag(), "Failed setting secure crypto protocols with
    error code: " << GetLastError());
}
```

The `WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3` option flag is defined if TLS 1.3 is present on the current build system. For more information, see [WINHTTP_OPTION_SECURE_PROTOCOLS](#) and [TLS protocol version support](#) on the Microsoft website.

3. Choose one of the following:

- **To enforce TLS 1.2:**

Under the `#else` directive, change the value of the `flags` variable, as follows.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
```

- **To enforce TLS 1.3:**

Under the `#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)` directive, change the value of the `flags` variable, as follows.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
```

4. If you performed the preceding code changes, build and install the AWS SDK for C++ according to the instructions at <https://github.com/aws/aws-sdk-cpp#building-the-sdk>.

- For the service client in your application, enable `verifySSL` in its client configuration, if this option isn't already enabled.

To enforce TLS 1.2 with WinINet and IXMLHTTPRequest2

There is no API to specify the secure protocol for the WinINet and IXMLHTTPRequest2 libraries. So the AWS SDK for C++ uses the default for the operating system. You can update the Windows registry to enforce the use of TLS 1.2, as shown in the following procedure. Be advised, however, that the result is a global change that impacts all applications that depend on Schannel.

- Open Registry Editor and go to `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`.
- If they don't already exist, create the following subkeys: `TLS 1.0`, `TLS 1.1`, and `TLS 1.2`.
- Under each of the subkeys, create a `Client` subkey and a `Server` subkey.
- Create the following keys and values.

Key name	Key type	Value
-----	-----	-----
<code>TLS 1.0\Client\DisabledByDefault</code>	DWORD	0
<code>TLS 1.1\Client\DisabledByDefault</code>	DWORD	0
<code>TLS 1.2\Client\DisabledByDefault</code>	DWORD	0
<code>TLS 1.0\Client\Enabled</code>	DWORD	0
<code>TLS 1.1\Client\Enabled</code>	DWORD	0
<code>TLS 1.2\Client\Enabled</code>	DWORD	1

Notice that `TLS 1.2\Client\Enabled` is the only key that's set to 1. Setting this key to 1 enforces TLS 1.2 as the only acceptable secure protocol.

To enforce TLS 1.3 with WinINet and IXMLHTTPRequest2

There is no API to specify the secure protocol for the WinINet and IXMLHTTPRequest2 libraries. So the AWS SDK for C++ uses the default for the operating system. You can update the Windows registry to enforce the use of TLS 1.3, as shown in the following procedure. Be advised, however, that the result is a global change that impacts all applications that depend on Schannel.

- Open Registry Editor and go to `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`.

2. If they don't already exist, create the following subkeys: TLS 1.0, TLS 1.1, TLS 1.2 and TLS 1.3.
3. Under each of the subkeys, create a Client subkey and a Server subkey.
4. Create the following keys and values.

Key name	Key type	Value
-----	-----	-----
TLS 1.0\Client\DisabledByDefault	DWORD	0
TLS 1.1\Client\DisabledByDefault	DWORD	0
TLS 1.2\Client\DisabledByDefault	DWORD	0
TLS 1.3\Client\DisabledByDefault	DWORD	0
TLS 1.0\Client\Enabled	DWORD	0
TLS 1.1\Client\Enabled	DWORD	0
TLS 1.2\Client\Enabled	DWORD	0
TLS 1.3\Client\Enabled	DWORD	1

Notice that TLS 1.3\Client\Enabled is the only key that's set to 1. Setting this key to 1 enforces TLS 1.3 as the only acceptable secure protocol.

Amazon S3 Encryption Client Migration

This topic shows how to migrate your applications from Version 1 (V1) of the Amazon Simple Storage Service (Amazon S3) encryption client to Version 2 (V2) and ensure application availability throughout the migration process.

Migration Overview

This migration happens in two phases:

1. **Update existing clients to read new formats.** First, deploy an updated version of the AWS SDK for C++ to your application. This allows existing V1 encryption clients to decrypt objects written by the new V2 clients. If your application uses multiple AWS SDKs, you must upgrade each SDK separately.
2. **Migrate encryption and decryption clients to V2.** Once all of your V1 encryption clients can read new formats, you can migrate your existing encryption and decryption clients to their respective V2 versions.

Update Existing Clients to Read New Formats

You must first update your existing clients to the latest SDK release. After completing this step, your application's V1 clients will be able to decrypt objects encrypted by V2 encryption clients without updating your application's code base.

Build and Install the Latest Version of the AWS SDK for C++

Applications Consuming the SDK from Source

If you build and install the AWS SDK for C++ from source, download or clone the SDK source from [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) on GitHub . Then repeat your normal build and install steps.

If you are upgrading AWS SDK for C++ from a version earlier than 1.8.x, see this [CHANGELOG](#) for breaking changes introduced in each major version. For more information about how to build and install the AWS SDK for C++, see [Get the AWS SDK for C++ from source code](#).

Applications Consuming the SDK from Vcpkg

If your application uses [Vcpkg](#) to track SDK updates, simply use your existing Vcpkg upgrade method to upgrade the SDK to the latest version. Keep in mind, there is a delay between when a version is released and when it is available through a package manager. The most recent version is always available through [installing from source](#).

You can run the following command to upgrade package `aws-sdk-cpp`:

```
vcpkg upgrade aws-sdk-cpp
```

And verify the version of package `aws-sdk-cpp`:

```
vcpkg list aws-sdk-cpp
```

The version should be at least 1.8.24.

For more information on using Vcpkg with the AWS SDK for C++, see [Get the AWS SDK for C++ from a package manager](#).

Build, Install, and Deploy Your Applications

If your application is statically linking against the AWS SDK for C++, code changes are not required in your application, but you must build your application again to consume the latest SDK changes. This step is not necessary for dynamic linking.

After upgrading your application's dependency version and verifying application functionality, proceed to deploying your application to your fleet. Once application deployment is complete, you can proceed with the next phase for migrating your application to use the V2 encryption and decryption clients.

Migrate Encryption and Decryption Clients to V2

The following steps show you how to successfully migrate your code from V1 to V2 of the Amazon S3 encryption client. Since code changes are required, you will need to rebuild your application regardless of whether it's statically or dynamically linking against the AWS SDK for C++.

Using New Encryption Materials

With V2 Amazon S3 encryption clients and the V2 crypto configuration, the following encryption materials have been deprecated:

- `SimpleEncryptionMaterials`
- `KMSEncryptionMaterials`

They have been replaced with the following secure encryption materials:

- `SimpleEncryptionMaterialsWithGCMAAD`
- `KMSWithContextEncryptionMaterials`

The following code changes are required to construct a V2 S3 encryption client:

- **If you are using `KMSEncryptionMaterials` when creating an S3 encryption client:**
 - When creating a V2 S3 encryption client, replace `KMSEncryptionMaterials` with `KMSWithContextEncryptionMaterials` and specify it in the V2 crypto configuration.
 - When putting an object with V2 Amazon S3 encryption clients, you must explicitly provide a string-string context map as the KMS context for encrypting the CEK. This might be an empty map.

- **If you are using SimpleEncryptionMaterials when creating an S3 encryption client:**
 - When creating a V2 Amazon S3 encryption client, replace SimpleEncryptionMaterials with SimpleEncryptionMaterialsWithGCMAAD and specify it in the V2 crypto configuration.
 - When putting an object with V2 Amazon S3 encryption clients, you must explicitly provide an empty string-string context map, otherwise the SDK will return an error.

Example: Using the KMS/KMSWithContext Key Wrap Algorithm

Pre-migration (KMS key wrap)

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
encryptionClient.PutObject(putObjectRequest);
```

Post-migration (KMSWithContext key wrap)

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
Aws::Map<Aws::String, Aws::String> kmsContextMap;
kmsContextMap.emplace("client", "aws-sdk-cpp");
kmsContextMap.emplace("version", "1.8.0");
encryptionClient.PutObject(putObjectRequest, kmsContextMap /* could be empty as well
*/);
```

Example: Using the AES/AES-GCM Key Wrap Algorithm

Pre-migration (AES key wrap)

```
auto materials = Aws::MakeShared<SimpleEncryptionMaterials>("s3Encryption",
    HashingUtils::Base64Decode(AES_MASTER_KEY_BASE64));
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
```

```
encryptionClient.PutObject(putObjectRequest);
```

Post-migration (AES-GCM key wrap)

```
auto materials = Aws::MakeShared<SimpleEncryptionMaterialsWithGCMAAD>("s3EncryptionV2",  
    HashingUtils::Base64Decode(AES_MASTER_KEY_BASE64));  
CryptoConfigurationV2 cryptoConfig(materials);  
S3EncryptionClientV2 encryptionClient(cryptoConfig);  
// Code snippet here to setup the putObjectRequest object.  
encryptionClient.PutObject(putObjectRequest, {} /* must be an empty map */);
```

Additional Examples

The following examples demonstrate how to address specific use cases related to a migration from V1 to V2.

Decrypt Objects Encrypted by Legacy Amazon S3 Encryption Clients

By default, you can't use the V2 Amazon S3 encryption client to decrypt objects that were encrypted with deprecated key wrap algorithms or deprecated content crypto schemas.

The following key wrap algorithms have been deprecated:

- KMS
- AES_KEY_WRAP

And the following content crypto schemas have been deprecated:

- CBC
- CTR

If you're using legacy Amazon S3 encryption clients in the AWS SDK for C++ to encrypt the objects, you're likely using the deprecated methods if:

- You used `SimpleEncryptionMaterials` or `KMSEncryptionMaterials`.
- You used `ENCRYPTION_ONLY` as `Crypto Mode` in your crypto configuration.

To use the V2 Amazon S3 encryption client to decrypt objects that were encrypted by deprecated key wrap algorithms or deprecated content crypto schemas, you must override the default value of `SecurityProfile` in the V2 crypto configuration from `V2` to `V2_AND_LEGACY`.

Example

Pre-migration

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

Post-migration

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
cryptoConfig.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

Decrypt Objects with Range

With legacy Amazon S3 encryption clients, you can specify a range of bytes to receive when decrypting an S3 object. In the V2 Amazon S3 encryption client, this feature is `DISABLED` by default. Therefore you have to override the default value of `RangeGetMode` from `DISABLED` to `ALL` in the V2 crypto configuration.

Example

Pre-migration

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
```

```
getObjectRequest.WithRange("bytes=38-75");
encryptionClient.GetObject(getObjectRequest);
```

Post-migration

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
cryptoConfig.SetUnAuthenticatedRangeGet(RangeGetMode::ALL);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
getObjectRequest.WithRange("bytes=38-75");
encryptionClient.GetObject(getObjectRequest);
```

Decrypt Objects with any CMK

When decrypting objects that were encrypted with `KMSWithContextEncryptionMaterials`, V2 Amazon S3 encryption clients are capable of letting KMS to find the proper CMK by providing an empty master key. This feature is `DISABLED` by default. You have to configure it explicitly by calling `SetKMSTDecryptWithAnyCMK(true)` for your KMS encryption materials.

Example

Pre-migration

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption", ""/* provide
    an empty KMS Master Key*/);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

Post-migration

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    ""/* provide an empty KMS Master Key*/);
materials.SetKMSTDecryptWithAnyCMK(true);
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

For complete code for all of these migration scenarios, see the [Amazon S3 Encryption example](#) on Github.

Document history for the AWS SDK for C++ Developer Guide

This topic lists important changes to the AWS SDK for C++ Developer Guide. For notification about updates to this documentation, you can subscribe to an [RSS feed](#).

Change	Description	Date
Custom libcrypto	Added content for custom libcrypto usage. Removed CMake maximum limitation. Updated available CMake parameters.	February 20, 2024
Table of contents	Updated table of contents to make code examples more accessible.	June 1, 2023
IAM best practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM .	March 1, 2023
Removing nuget	Removing mention of nuget as a viable package manager option because the latest version available is too old.	December 2, 2022
Updates to Getting Started	Updated vcpkg content to clearly communicate that is not supported by AWS and is an external option. Updated instructions on building the SDK for Windows with curl.	October 18, 2022
Updates to ClientConfiguration	Updated the structure of the ClientConfiguration	September 22, 2022

	to accurately reflect latest API.	
Improvements to Getting Started	Improved clarity of getting started instructions for building the SDK on Windows and Linux.	June 24, 2022
Windows curl Support	Added notes about building the SDK for Windows with curl.	June 15, 2022
Retiring EC2-Classic	Added notes about retiring EC2-Classic.	April 13, 2022
Enabling SDK Metrics	Removed information about enabling SDK metrics, which has been deprecated.	January 20, 2022
Working with AWS services	Included lists of the code examples that are available on GitHub in the Code Examples repository.	January 11, 2022
Improvements	Improvements to Getting started section, Code examples section, and general standardization.	June 9, 2021
Version update	Reflected the update to general release version of SDK to 1.9.	April 20, 2021
Getting started using the AWS SDK for C++	Updated section with new organization and details.	March 17, 2021

Amazon S3 Encryption Client Migration	Added information about how to migrate your applications from V1 to V2 of the Amazon S3 encryption client.	August 7, 2020
Security Content	Added security content.	February 6, 2020
Creating, listing, and deleting buckets	Updated the Amazon S3 <code>CreateBucket</code> example to support AWS Regions.	June 20, 2019
Build instructions	Updated the SDK build instructions.	April 16, 2019
Asynchronous methods	Added new section.	April 16, 2019
Service Client Classes	Various updates.	April 5, 2019
Managing Amazon S3 Access Permissions	Various updates.	April 3, 2019
Updates for building and for configuration variables	Updated the instructions for building the SDK. Updated the available AWS Client Configuration variables.	March 1, 2019
vcpkg C++ package manager	Updated the instructions for setting up the <code>vcpkg</code> C++ package manager.	January 19, 2019