AWS SDK for C++

Developer Guide

AWS SDK for C++: Developer Guide

Table of Contents

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.

Getting started using the AWS SDK for C++ (p. 2)

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

Getting started using 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 your AWS user credentials in your environment.

- Providing AWS credentials (p. 2)

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.

- Getting the AWS SDK for C++ from source code (p. 4)
- Getting the AWS SDK for C++ from a package manager (p. 13)

If you run into build issues regarding CMake, see Troubleshooting build issues (p. 18).

Providing AWS credentials

To connect to any of the supported services with the AWS SDK for C++, you must provide AWS credentials. 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, see Credentials Providers in the `aws-sdk-cpp` repository in GitHub.

Create an AWS account and administrator user

1. **Create an account.**

   To create an AWS account, see How do I create and activate a new AWS account?

2. **Create an administrative user.**

   Avoid using your AWS account root user (the initial account you create) to access the AWS Management Console and services. Instead, create an administrative user account, as explained in Creating your first IAM admin user and group.

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

Create AWS credentials and a profile

To use the SDK, create an AWS Identity and Access Management (IAM) user and obtain credentials for that user. Then make them available to the SDK in your development environment by saving them to the AWS shared credentials file.

To create and use credentials

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Users**, and then choose **Add user**.
3. Enter a user name. For this tutorial, we'll use *SdkUser*.
4. Under **Select AWS access type**, select **Programmatic access**, and then choose **Next: Permissions**.
5. Choose **Attach existing policies directly**.
6. In **Search**, enter **s3**, and then select **AmazonS3FullAccess**.
7. Choose **Next: Tags**, **Next: Review**, and **Create user**.
8. On the *Success* screen, choose **Download .csv**.

   The downloaded file contains the Access Key ID and the Secret Access Key for this IAM user.

   Note
   You will not have another opportunity to download or copy the Secret Access Key.
9. Treat your Secret Access Key as a password; save in a trusted location and do not share it.

   Warning
   Use appropriate security measures to keep these credentials safe and rotated.
10. Create or open the AWS shared credentials file. This file is `~/.aws/credentials` on Linux and macOS systems, and `%USERPROFILE%\.aws\credentials` on Windows.
11. Add the following text to the AWS shared credentials file, but replace the example ID value and example key value with the ones you obtained earlier. Save the file. See Location of Credentials Files for more information.

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

The preceding procedure is the simplest of several possibilities for authentication and authorization. For other options, see the following.

More information on user credentials

To explore other ways to provide credentials to SDKs, see the following:

- To create long-term AWS credentials, see Access keys in the *AWS General Reference*.
- To create short-term AWS credentials, see Temporary Security Credentials in the *IAM User Guide*.
- To learn more about supported provider chains, see the AWS SDKs and Tools Reference Guide, specifically:
  - The `.aws/credentials` and `.aws/config` files
  - Using environment variables
  - Assume role credentials
    - `role_arn` (corresponds to the `AWS_ROLE_ARN` environment variable)

- `web_identity_token_file` (corresponds to the `AWS_WEB_IDENTITY_TOKEN_FILE` environment variable)
- `role_session_name` (corresponds to the `AWS_ROLE_SESSION_NAME` environment variable)

- To learn more about the `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` environment variable, see IAM Roles for Tasks in the *Amazon Elastic Container Service Developer Guide*.

Getting 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
Build and install the SDK source 1. Use CMake to generate build files for the SDK. 2. Build the SDK. 3. Install the SDK.	First build the SDK from source and install it. • Build on Windows (p. 4) • Build on Linux/macOS (p. 7)
Build your application using the SDK 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.	Then develop your own application using the SDK. • "Hello, S3!" starter application (p. 10)

Building 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` 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 (p. 30) 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 (p. 21). 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 3.2; *maximum version 3.21*) 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 1.9 simplifies dependencies by using git submodules to wrap external dependencies.

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]
```

For more ways to modify the build output, see CMake Parameters (p. 21).

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:

```
"C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE
\CommonExtensions\Microsoft\CMake\CMake\bin\cmake.exe" "..\aws-sdk-cpp" -
DCMAKE_BUILD_TYPE=Debug
```

- **Generate build files (subset AWS services)**: To build only a particular service or services package(s) for the SDK, add the CMake BUILD_ONLY (p. 22) parameter. The following example builds only the Amazon S3 service package:

```
cmake ..\aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DBUILD_ONLY="s3"
```

- **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 (p. 24), CURL_INCLUDE_DIR (p. 23), and CURL_LIBRARY (p. 23). 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'
```

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 3.2, maximum version 3.21.

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

```
{path to MSBuild if not in PATH} ALL_BUILD.vcxproj -p:Configuration=[Debug |
Release]
```

```
"C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin
\MSBuild.exe" ALL_BUILD.vcxproj  -p:Configuration=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. This command installs the SDK in `\Program Files (x86)\aws-cpp-sdk-all\`. Command syntax:

   ```
   msbuild INSTALL.vcxproj -p:Configuration=[Debug | Release]
   ```

   ```
   msbuild INSTALL.vcxproj -p:Configuration=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 NMAKEinstalled 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++

Building 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 (p. 21).

To build the SDK from source

1. Install CMake (minimum version 3.2; *maximum version 3.21*) 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 1.9 simplifies dependencies by using git submodules to wrap external dependencies.

   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 (p. 21).

   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 (p. 22) 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 3.2, maximum version 3.21.

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" app

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 (`main.cpp`) and one `CMakeLists.txt` file. The program uses Amazon S3 to report storage bucket information.

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 (p. 4), Linux/macOS (p. 7)):

- 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: Runtime Libraries on Windows
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 uses Amazon S3 and so also requires `aws-cpp-sdk-s3`). If these DLLs are not in the executable location then runtime exceptions of 'file not found' occur. 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. Review the two `AWSSDK_CPY_DYN_LIBS` lines and CHOOSE which pattern is accurate for your build environment.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/logging/LogLevel.h>
#include <aws/s3/S3Client.h>
#include <iostream>

using namespace Aws;

int main()
{
    //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.
    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    //The AWS SDK for C++ must be initialized by calling Aws::InitAPI.
    InitAPI(options);
    {
        S3::S3Client client;

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

    //Before the application terminates, the SDK must be shut down.
    ShutdownAPI(options);
    return 0;
}
```

To create the folder and source files

1. Create a directory to hold your source files.

   Note
   This example can also be completed in Visual Studio: choose **Create New Project** and then choose **CMake Project**.
2. Within that folder, add a `main.cpp` file that includes the following code, which reports the Amazon S3 buckets you own.
3. Add a `CMakeLists.txt` file that specifies your project's name, executables, source files, and linked libraries. **Note**: There is a "TODO" comment in this file prompting you to choose the appropriate line for your build environment.

```
cmake_minimum_required(VERSION 3.3)
set(CMAKE_CXX_STANDARD 11)
project(app LANGUAGES CXX)

#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()

message(STATUS "CMAKE_PREFIX_PATH: ${CMAKE_PREFIX_PATH}")
set(BUILD_SHARED_LIBS ON CACHE STRING "Link to shared libraries by default.")

#Load required services/packages: This basic example uses S3.
find_package(AWSSDK REQUIRED COMPONENTS s3)
add_executable(${PROJECT_NAME} "main.cpp") #Add app's main starting file.

#Windows: This 'if' clause copies the SDK libraries from the installation location to
 the place where
#  this project's executable is located.  Without this you'll need to copy the install
#  /bin folder to the exe location (app.exe in this case) to prevent runtime errors.
```

```
if(MSVC AND BUILD_SHARED_LIBS)
    target_compile_definitions(${PROJECT_NAME} PUBLIC "USE_IMPORT_EXPORT")
    add_definitions(-DUSE_IMPORT_EXPORT)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.
    list(APPEND SERVICE_LIST s3)

    #For IDE's like Xcode and Visual Studio this line will be ignored because Release/
Debug
    #  is switched internally, but this is necessary for non-IDE builds.
    set(CMAKE_BUILD_TYPE Debug) #TODO: Set to your build type

    #TODO:Choose appropriate one of the following two lines, you want to copy to the
 same folder where your executables are.
    AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}/
${CMAKE_BUILD_TYPE})  #Choose this line if your executables are in /build/Debug
    #AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR})  #Choose this
 line for Visual Studio and possibly other IDEs

    message(STATUS ">>CMAKE_CURRENT_BINARY_DIR: ${CMAKE_CURRENT_BINARY_DIR}")
    message(STATUS ">>CMAKE_BUILD_TYPE: ${CMAKE_BUILD_TYPE}")
    message(STATUS ">>EXECUTABLE_OUTPUT_PATH : ${EXECUTABLE_OUTPUT_PATH}")
endif()

set_compiler_flags(${PROJECT_NAME})
set_compiler_warnings(${PROJECT_NAME})
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.

To build the application

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 (`"C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MSBuild.exe" ALL_BUILD.vcxproj`) 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.

To run the program

1. Change to the Debug directory where the result of the build was generated. In this example, `app.exe` is in folder `my_project_build\Debug`.

```
cd Debug
```

2. Run the program using the name of the executable.

```
app
```

For additional examples using the AWS SDK for C++, see Code examples with guidance for the AWS SDK for C++ (p. 47).

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

You can use the AWS SDK for C++ in from your code by using a package manager to install the SDK. The package manager automatically includes the binaries for each runtime/architecture configuration you use — you don't need to manage these dependencies yourself.

Each package manager is independent from AWS and controls its own availability of versions. There is a delay between when a version is released by AWS and when it is available through a package manager. The most recent version is always available through installing from source (p. 4).

vcpkg is a package manager option actively supported by external contributors. Nuget is no longer being updated with new versions and only provides up to version 1.6 of the AWS SDK for C++.

Getting 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 (p. 4).

You can use the vcpkg package manager to access AWS SDK for C++ functionality for your project. Note that this package manager is updated and maintained by external contributors and may not reflect the latest available version for the AWS SDK for C++. To use vcpkg with the SDK for C++, see:

- Installing vcpkg on Windows (p. 14)
- Installing vcpkg on Linux/macOS (p. 15)

Then you can create a simple application to use it:

- "Hello, S3!" starter application (p. 16)

Getting the SDK using NuGet (deprecated)

Important
You can only use NuGet to install AWS SDK for C++ versions 1.6 and earlier with Microsoft Visual Studio 2015 and 2017.

For AWS SDK for C++ versions 1.6 and earlier, and for Microsoft Visual Studio 2015 and 2017, you can use NuGet to manage dependencies for AWS SDK for C++ projects that you develop with Microsoft Visual C++. To use this procedure, first install NuGet.

To install the SDK with NuGet

1. In Visual Studio, open your project.
2. In **Solution Explorer**, right-click your project name, and then choose **Manage NuGet Packages**.
3. Select the packages to use by searching for a particular service or library name. For example, you could use a search term such as `aws s3 native`. Use the search term `AWSSDKCPP-`*`service name`* to find a library for a particular service to your project.
4. Choose **Install** to install the libraries and add them to your project.

AWS SDK for C++ Setup using vcpkg 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 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.

Getting the SDK using vcpkg

You can use the vcpkg package manager to access AWS SDK for C++ functionality for your project. To use this procedure, you must install vcpkg on your system.

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 (p. 4).

To install AWS SDK for C++ with vcpkg

1. 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`

- Also as part of those instructions, proceed to integrate with whatever build environment you are using, such as `.\vcpkg\vcpkg integrate install` if you are using Visual Studio, or following the provided alternatives to use CMake, etc.

The output displays a messages similar to the following:

```
All MSBuild C++ projects can now #include any installed libraries.
Linking will be handled automatically.
Installing new libraries will make them instantly available.

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

2. 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.
3. You'll 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 packages will be built and installed:
    aws-sdk-cpp[core,dynamodb,kinesis,s3]:x86-windows -> 1.8.126#6
```

This example output indicates the build configuration is "x86-windows" and the AWS SDK for C++ version installed is 1.8.

AWS SDK for C++ Setup using vcpkg 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.

Getting the SDK Using Vcpkg

You can use the vcpkg package manager to install AWS SDK for C++. To use this procedure, you must install vcpkg on your system.

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 (p. 4).

To install AWS SDK for C++ with vcpkg

1. 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`
   - Also as part of those instructions, proceed to integrate the build environment you are using, such as `./vcpkg/vcpkg integrate install` if you are using Visual Studio Code, or following the provided alternatives to use CMake, etc.

   The output displays a messages similar to the following:

   ```
   Applied user-wide integration for this vcpkg root.

   CMake projects should use: "-DCMAKE_TOOLCHAIN_FILE=/local/home/username/vcpkg/scripts/
   buildsystems/vcpkg.cmake"
   ```

2. Copy the complete `-DCMAKE_TOOLCHAIN_FILE` command to use for CMake later. The vcpkg GitHub Readme also describes where to use this for your toolset.

"Hello S3" app

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.

You can use any IDE or C++ tools to build your program. This tutorial shows how to use Visual Studio, but it is not required for vcpkg.

This example consists of one source file (`main.cpp`) and one `CMakeLists.txt` file. The program uses Amazon S3 to report storage bucket information.

To create and run the application

1. In Visual Studio, choose **Create New Project** and then choose **CMake Project**.
2. Remove the generated *.cpp and *.h files that are created with the solution.
3. Add a `main.cpp` file that includes the following code, which reports the Amazon S3 buckets you own.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/logging/LogLevel.h>
#include <aws/s3/S3Client.h>
#include <iostream>

using namespace Aws;

int main()
{
    //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.
    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    //The AWS SDK for C++ must be initialized by calling Aws::InitAPI.
    InitAPI(options);
    {
        S3::S3Client client;

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

    //Before the application terminates, the SDK must be shut down.
    ShutdownAPI(options);
    return 0;
}
```

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

```
# Minimal CMakeLists.txt for the AWS SDK for C++.
cmake_minimum_required(VERSION 3.3)
set(CMAKE_CXX_STANDARD 11)
project(app LANGUAGES CXX)

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

#Include in any AWS service packages your code will be using by service name
find_package(AWSSDK REQUIRED COMPONENTS s3)
add_executable(${PROJECT_NAME} "main.cpp") #add app's main sourcefile

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

5. For the build configuration (e.g. Debug vs. Release) in the toolbar, choose **Manage Configurations**. This adds a `CMakeSettings.json` to your project.
6. On the **CmakeSettings.json** tab, choose **Configurations**, and add or choose the configuration that matches the SDK executables installed during the vcpkg installation step. The output of the prior vcpkg install indicates the configuration (like *x86-windows*).

7. On the **CmakeSettings.json** tab, in the **General** section, for **CMake toolchain file** enter the CMAKE_TOOLCHAIN_FILE path obtained from the prior vcpkg install output.
8. On either `CMakeLists.txt` and `CMakeSettings.json`, choose **Save**. Saving either of these files automatically runs the CMake generation for your build configuration.
9. Confirm that CMake set up successfully by verifying that the last output is **CMake generation finished.**
10. (Optional) If you have not already set up your AWS credentials in your environment, you can use the **AWS Explorer** within Visual Studio to load a **New Account Profile** based on your downloaded credentials file. See Adding a profile to the SDK Credential Store in the *AWS Toolkit for Visual Studio User Guide*.
11. Open `main.cpp` so that it is your active tab.
12. Choose **Run**.

    Visual Studio uses CMake to use the information in `CMakeLists.txt` and `CMakeSettings.json` to build an executable program.
13. Confirm that the console output lists the total number of Amazon S3 buckets and the name of each bucket.

For additional examples using the AWS SDK for C++, see Code examples with guidance for the AWS SDK for C++ (p. 47).

Troubleshooting build issues

Topics

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 (p. 4), Linux/macOS (p. 7)). Insert the following command before your first call to `find_package()` in your `CMakeLists.txt` file. See ??? (p. 10) 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 (p. 4), Linux/macOS (p. 7)). Insert the following commands before your first call to `find_package()` in your `CMakeLists.txt` file. See ??? (p. 10) 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 ??? (p. 10) 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 ??? (p. 10) 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.

Configuring the AWS SDK for C++

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

Topics

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 your SDK build.

> **Note**
> To use the *ADD_CUSTOM_CLIENTS* or *REGENERATE_CLIENTS* variables, you must have Python 2.7, Java (JDK 1.8+), and Maven installed and in your PATH.

Topics

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,versio
```

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_SHARED_LIBS

A built-in CMake option, re-exposed here for visibility. If enabled, 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

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/libcur/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 enabled, custom memory management defaults to *off* (`0`). If dynamic linking is enabled, 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.

ENABLE_CURL_LOGGING

If enabled, the internal log for curl is piped to the SDK logger.

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 enabled, 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

FORCE_CURL

Windows only. If enabled, forces usage of the curl client instead of the default WinHTTP data transfer provider.

Values

ON | *OFF*

Default

OFF

FORCE_SHARED_CRT

If enabled, 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 (p. 24). If enabled, this option turns on *ENABLE_UNITY_BUILD* and additional binary size reduction settings.

Values

ON | *OFF*

Default

OFF

NO_ENCRYPTION

If enabled, 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 enabled, 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

This argument wipes out all generated code and generates the client directories from the `code-generation/api-definitions` folder. For example:

```
-DREGENERATE_CLIENTS=1
```

SIMPLE_INSTALL

If enabled, 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 (p. 26).

Values

WINDOWS | *LINUX* | *APPLE* | *ANDROID*

USE_OPENSSL

If enabled, the SDK builds using OpenSSL; otherwise, it uses `awslabs/aws-lc`. `AWS-LC` is a general-purpose cryptographic library maintained by the AWS Cryptography team for AWS and their customers. For more information, see CMake Parameters on GitHub.

Values

ON | *OFF*

Default

ON

Android CMake Variables and Options

Use the following variables when you are creating an Android build of the SDK (when TARGET_ARCH (p. 26) is set to *ANDROID*).

Topics

- NDK_DIR (p. 28)

ANDROID_ABI

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_NATIVE_API_LEVEL

Controls what API level the SDK builds against. If you set ANDROID_STL (p. 27) 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

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

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

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

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.

Overriding 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 Building the AWS SDK for C++ on Windows (p. 4).

Controlling 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);
```

Using the AWS SDK for C++

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++ (p. 2).

For service-specific programming examples, see AWS SDK for C++ Code Examples (p. 47).

Topics

Basic Use

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.

Initializing and Shutting Down 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.

Best practice requires all AWS SDK for C++ calls performed between `Aws::InitAPI` and `Aws::ShutdownAPI` either to be contained within a pair of curly braces or 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;
}
```

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.

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`*`::`*`Service`*`Client`. 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 (p. 2).

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 (p. 30).

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 *`tinyxml2`*). 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 (p. 35).

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

```
#ifdef 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;
}
```

Working with AWS services in 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 Getting started (p. 2) topic first.

You can find more code examples in the C++ example folder on GitHub.

Topics

Getting 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++ (p. 2). 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

Some examples include unit tests. The unit tests are in the `tests` folder, and there is one test source file for each example source file. Each test source file builds to its own executable that you can run by using the command line.

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++ (p. 2). 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 Providing AWS credentials (p. 2) 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

Note
NuGet SDK packages are deprecated. Even if you use Visual Studio, it is recommended to still get the latest source code from GitHub, and build the AWS SDK for C++ from source following the instructions.

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++ (p. 107) 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. 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. This source file is not for Windows users (unless you have taken additional steps to explicitly add cURL support).
6. Open file `CMakeLists.txt`.
7. 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 Troubleshooting build issues (p. 18).

- 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.

8. 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.
9. 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.
10. 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.

11. 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.

12. In the top menu, there is a **green arrow** and a **drop-down selector** for **Debug Target**. Choose `run_list_buckets.exe`.
13. Click the **green arrow run button** to **Select Startup Item**.
14. A Visual Studio Debug Console window will open and display the output of the code.
15. 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 Getting started on code examples (p. 40).

This section selects several AWS services and guides you through the examples using them. For other service examples (but without additional guidance beyond the code) see Additional code examples (p. 140).

Source code that shows how to work with AWS services using the AWS SDK for C++ is available in the AWS Code Examples Repository on GitHub. The following 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 (p. 48)	Collects and monitors metrics for AWS resources you are using
Amazon DynamoDB (p. 59)	A NoSQL database service
Amazon Elastic Compute Cloud (p. 68) (Amazon EC2)	Secure, resizable compute capacity
AWS Identity and Access Management (p. 86) (IAM)	Create and manage AWS users and groups, and use permissions to control access to AWS resources
Amazon Simple Storage Service (p. 107) (Amazon S3)	Data storage and retrieval (objects into buckets)
Amazon Simple Queue Service (p. 130) (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 (p. 44).

To propose a new code example to the AWS documentation team, follow the guidance on the Readme 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 Troubleshooting build issues (p. 18).

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 (p. 49)
- Publishing Custom Metric Data (p. 50)

- Working with CloudWatch Alarms (p. 51)
- Using Alarm Actions in CloudWatch (p. 54)
- Sending Events to CloudWatch (p. 56)

Getting Metrics from CloudWatch

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 (p. 51).

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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++ (p. 2) and have configured default AWS credentials using the information in Providing AWS Credentials (p. 2).

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

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/core/utils/Outcome.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

```
       Aws::Client::ClientConfiguration clientConfig;
       if (!region.empty())
           clientConfig.region = region;
       Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

       std::cout << "Creating table " << table <<
           " with a simple primary key: \"Name\"" << std::endl;

       Aws::DynamoDB::Model::CreateTableRequest req;

       Aws::DynamoDB::Model::AttributeDefinition haskKey;
       haskKey.SetAttributeName("Name");
       haskKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
       req.AddAttributeDefinitions(haskKey);

       Aws::DynamoDB::Model::KeySchemaElement keyscelt;

keyscelt.WithAttributeName("Name").WithKeyType(Aws::DynamoDB::Model::KeyType::HASH);
       req.AddKeySchema(keyscelt);

       Aws::DynamoDB::Model::ProvisionedThroughput thruput;
       thruput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
       req.SetProvisionedThroughput(thruput);
       req.SetTableName(table);

       const Aws::DynamoDB::Model::CreateTableOutcome& result =
dynamoClient.CreateTable(req);
       if (result.IsSuccess())
       {
           std::cout << "Table \"" <<
result.GetResult().GetTableDescription().GetTableName() <<
               " created!" << std::endl;
```

```
        }
        else
        {
            std::cout << "Failed to create table: " << result.GetError().GetMessage();
        }
```

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/core/utils/Outcome.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

```
        Aws::Client::ClientConfiguration clientConfig;
        if (!region.empty())
            clientConfig.region = region;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        std::cout << "Creating table " << table <<
            " with a composite primary key:\n" \
            "* Language - partition key\n" \
            "* Greeting - sort key\n";

        Aws::DynamoDB::Model::CreateTableRequest req;

        Aws::DynamoDB::Model::AttributeDefinition hashKey1, hashKey2;

hashKey1.WithAttributeName("Language").WithAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S)
        req.AddAttributeDefinitions(hashKey1);

hashKey2.WithAttributeName("Greeting").WithAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S)
        req.AddAttributeDefinitions(hashKey2);

        Aws::DynamoDB::Model::KeySchemaElement kse1, kse2;

kse1.WithAttributeName("Language").WithKeyType(Aws::DynamoDB::Model::KeyType::HASH);
        req.AddKeySchema(kse1);

kse2.WithAttributeName("Greeting").WithKeyType(Aws::DynamoDB::Model::KeyType::RANGE);
        req.AddKeySchema(kse2);

        Aws::DynamoDB::Model::ProvisionedThroughput thruput;
        thruput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
        req.SetProvisionedThroughput(thruput);

        req.SetTableName(table);

        const Aws::DynamoDB::Model::CreateTableOutcome& result =
dynamoClient.CreateTable(req);
        if (result.IsSuccess())
```

```
        {
            std::cout << "Table \"" <<
 result.GetResult().GetTableDescription().GetTableName() <<
                "\" was created!\n";
        }
        else
        {
            std::cout << "Failed to create table:" << result.GetError().GetMessage();
        }
```

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

```
        Aws::Client::ClientConfiguration clientConfig;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do
        {
            const Aws::DynamoDB::Model::ListTablesOutcome& lto =
 dynamoClient.ListTables(listTablesRequest);
            if (!lto.IsSuccess())
            {
                std::cout << "Error: " << lto.GetError().GetMessage() << std::endl;
                return 1;
            }

            for (const auto& s : lto.GetResult().GetTableNames())
                std::cout << s << std::endl;

 listTablesRequest.SetExclusiveStartTableName(lto.GetResult().GetLastEvaluatedTableName());

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

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

Code

```
       Aws::Client::ClientConfiguration clientConfig;
       if (!region.empty())
           clientConfig.region = region;
       Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

       Aws::DynamoDB::Model::DescribeTableRequest dtr;
       dtr.SetTableName(table);

       const Aws::DynamoDB::Model::DescribeTableOutcome& result =
dynamoClient.DescribeTable(dtr);

       if (result.IsSuccess())
       {
           const Aws::DynamoDB::Model::TableDescription& td =
result.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::cout << "Failed to describe table: " << result.GetError().GetMessage();
       }
```

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

```
       Aws::Client::ClientConfiguration clientConfig;
       Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

       std::cout << "Updating " << table << " with new provisioned throughput values" <<
std::endl;
       std::cout << "Read capacity : " << rc << std::endl;
       std::cout << "Write capacity: " << wc << std::endl;

       Aws::DynamoDB::Model::UpdateTableRequest utr;
       Aws::DynamoDB::Model::ProvisionedThroughput pt;
       pt.WithReadCapacityUnits(rc).WithWriteCapacityUnits(wc);
       utr.WithProvisionedThroughput(pt).WithTableName(table);

       const Aws::DynamoDB::Model::UpdateTableOutcome& result =
dynamoClient.UpdateTable(utr);
       if (!result.IsSuccess())
       {
           std::cout << result.GetError().GetMessage() << std::endl;
           return 1;
       }
       std::cout << "Done!" << std::endl;
```

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

Code

```
       Aws::Client::ClientConfiguration clientConfig;
       if (!region.empty())
           clientConfig.region = region;
       Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

       Aws::DynamoDB::Model::DeleteTableRequest dtr;
       dtr.SetTableName(table);

       const Aws::DynamoDB::Model::DeleteTableOutcome& result =
dynamoClient.DeleteTable(dtr);
       if (result.IsSuccess())
       {
           std::cout << "Your Table \"" <<
result.GetResult().GetTableDescription().GetTableName() << " was deleted!\n";
       }
```

```
        else
        {
            std::cout << "Failed to delete table: " << result.GetError().GetMessage();
        }
```

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

Code

```
        Aws::Client::ClientConfiguration clientConfig;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
        Aws::DynamoDB::Model::GetItemRequest req;

        // Set up the request.
        req.SetTableName(table);
        Aws::DynamoDB::Model::AttributeValue hashKey;
        hashKey.SetS(keyval);
        req.AddKey(key, hashKey);

        // Retrieve the item's fields and values
        const Aws::DynamoDB::Model::GetItemOutcome& result = dynamoClient.GetItem(req);
        if (result.IsSuccess())
        {
            // Reference the retrieved fields/values.
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>& item =
 result.GetResult().GetItem();
            if (item.size() > 0)
            {
                // 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 " << key << std::endl;
            }
        }
        else
        {
            std::cout << "Failed to get item: " << result.GetError().GetMessage();
        }
```

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/core/utils/Outcome.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

```
        Aws::Client::ClientConfiguration clientConfig;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

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

        Aws::DynamoDB::Model::AttributeValue av;
        av.SetS(keyVal);

        Aws::DynamoDB::Model::AttributeValue album;
        album.SetS(AlbumTitleValue);

        Aws::DynamoDB::Model::AttributeValue awards;
        awards.SetS(AwardVal);

        Aws::DynamoDB::Model::AttributeValue song;
        song.SetS(SongTitleVal);

        // Add all AttributeValue objects.
        putItemRequest.AddItem(key, av);
        putItemRequest.AddItem(albumTitle, album);
        putItemRequest.AddItem(Awards, awards);
        putItemRequest.AddItem(SongTitle, song);

        const Aws::DynamoDB::Model::PutItemOutcome result =
dynamoClient.PutItem(putItemRequest);
        if (!result.IsSuccess())
        {
            std::cout << result.GetError().GetMessage() << std::endl;
```

```
            return 1;
        }
        std::cout << "Successfully added Item!" << std::endl;
```

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

Code

```
        Aws::Client::ClientConfiguration clientConfig;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        // *** 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(keyValue);
        request.AddKey("id", attribValue);

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

        // Parse the attribute name and value. Syntax: "name=value".
        auto parsed = Aws::Utils::StringUtils::Split(attributeNameAndValue, '=');

        if (parsed.size() != 2)
        {
            std::cout << "Invalid argument syntax: " << attributeNameAndValue << USAGE;
            return 1;
        }

        // Construct attribute name argument
        // Note: Setting the ExpressionAttributeNames argument is required only
        // when the name is a reserved word, such as "default". Otherwise, the
        // name can be included in the update_expression, as in
        // "SET MyAttributeName = :valueA"
        Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
        expressionAttributeNames["#a"] = parsed[0];
        request.SetExpressionAttributeNames(expressionAttributeNames);

        // Construct attribute value argument.
        Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
        attributeUpdatedValue.SetS(parsed[1]);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
```

```
        expressionAttributeValues[":valueA"] = attributeUpdatedValue;
        request.SetExpressionAttributeValues(expressionAttributeValues);

        // Update the item.
        const Aws::DynamoDB::Model::UpdateItemOutcome& result =
dynamoClient.UpdateItem(request);
        if (!result.IsSuccess())
        {
            std::cout << result.GetError().GetMessage() << std::endl;
            return 1;
        }
        std::cout << "Item was updated" << std::endl;
```

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

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 ec2;

    Aws::EC2::Model::RunInstancesRequest run_request;
    run_request.SetImageId(ami_id);
    run_request.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    run_request.SetMinCount(1);
    run_request.SetMaxCount(1);

    auto run_outcome = ec2.RunInstances(run_request);
    if (!run_outcome.IsSuccess())
    {
        std::cout << "Failed to start ec2 instance " << instanceName <<
            " based on ami " << ami_id << ":" <<
            run_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    const auto& instances = run_outcome.GetResult().GetInstances();
    if (instances.size() == 0)
    {
        std::cout << "Failed to start ec2 instance " << instanceName <<
            " based on ami " << ami_id << ":" <<
            run_outcome.GetError().GetMessage() << std::endl;
        return;
    }
```

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 ec2;

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

    auto dry_run_outcome = ec2.StartInstances(start_request);
```

```
    assert(!dry_run_outcome.IsSuccess());
    if (dry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION)
    {
        std::cout << "Failed dry run to start instance " << instance_id << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    start_request.SetDryRun(false);
    auto start_instancesOutcome = ec2.StartInstances(start_request);

    if (!start_instancesOutcome.IsSuccess())
    {
        std::cout << "Failed to start instance " << instance_id << ": " <<
            start_instancesOutcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully started instance " << instance_id <<
            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 ec2;
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instance_id);
    request.SetDryRun(true);

    auto dry_run_outcome = ec2.StopInstances(request);
    assert(!dry_run_outcome.IsSuccess());

    if (dry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION)
    {
        std::cout << "Failed dry run to stop instance " << instance_id << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    request.SetDryRun(false);
    auto outcome = ec2.StopInstances(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to stop instance " << instance_id << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully stopped instance " << instance_id <<
```

```
            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 ec2;

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

    auto dry_run_outcome = ec2.RebootInstances(request);
    assert(!dry_run_outcome.IsSuccess());

    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;
    }

    request.SetDryRun(false);
    auto outcome = ec2.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 ec2;
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done)
        {
            auto outcome = ec2.DescribeInstances(request);
            if (outcome.IsSuccess())
            {
                if (!header)
                {
                    std::cout << std::left <<
                        std::setw(48) << "Name" <<
                        std::setw(20) << "ID" <<
                        std::setw(15) << "Ami" <<
                        std::setw(15) << "Type" <<
                        std::setw(15) << "State" <<
                        std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }

                const auto &reservations =
                    outcome.GetResult().GetReservations();

                for (const auto &reservation : reservations)
                {
                    const auto &instances = reservation.GetInstances();
                    for (const auto &instance : instances)
                    {
                        Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                instance.GetState().GetName());

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

                        Aws::String monitor_str =

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

                        const auto &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(15) << instance.GetImageId() <<
                        std::setw(15) << type_string <<
                        std::setw(15) << instanceStateString <<
                        std::setw(15) << monitor_str << std::endl;
                }
            }

            if (outcome.GetResult().GetNextToken().size() > 0)
            {
                request.SetNextToken(outcome.GetResult().GetNextToken());
            }
            else
            {
                done = true;
            }
        }
        else
        {
            std::cout << "Failed to describe ec2 instances:" <<
                outcome.GetError().GetMessage() << std::endl;
            done = true;
        }
    }
```

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 for Linux Instances.

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/MonitorInstancesResponse.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesResponse.h>
#include <iostream>
```

Code

```
    Aws::EC2::EC2Client ec2;
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instance_id);
    request.SetDryRun(true);

    auto dry_run_outcome = ec2.MonitorInstances(request);
```

```
    assert(!dry_run_outcome.IsSuccess());
    if (dry_run_outcome.GetError().GetErrorType()
        != Aws::EC2::EC2Errors::DRY_RUN_OPERATION)
    {
        std::cout << "Failed dry run to enable monitoring on instance " <<
            instance_id << ": " << dry_run_outcome.GetError().GetMessage() <<
            std::endl;
        return;
    }

    request.SetDryRun(false);
    auto monitorInstancesOutcome = ec2.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess())
    {
        std::cout << "Failed to enable monitoring on instance " <<
            instance_id << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully enabled monitoring on instance " <<
            instance_id << 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/MonitorInstancesResponse.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesResponse.h>
#include <iostream>
```

Code

```
    Aws::EC2::EC2Client ec2;
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instance_id);
    unrequest.SetDryRun(true);

    auto undry_run_outcome = ec2.UnmonitorInstances(unrequest);
    assert(!undry_run_outcome.IsSuccess());
    if (undry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION)
    {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instance_id << ": " << undry_run_outcome.GetError().GetMessage() <<
            std::endl;
        return;
    }

    unrequest.SetDryRun(false);
    auto unmonitorInstancesOutcome = ec2.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess())
```

```
{
    std::cout << "Failed to disable monitoring on instance " << instance_id
        << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully disable monitoring on instance " <<
        instance_id << 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 ec2;

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

    auto outcome = ec2.AllocateAddress(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to allocate elastic ip address:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::String allocation_id = outcome.GetResult().GetAllocationId();

    Aws::EC2::Model::AssociateAddressRequest associate_request;
    associate_request.SetInstanceId(instance_id);
    associate_request.SetAllocationId(allocation_id);

    auto associate_outcome = ec2.AssociateAddress(associate_request);
    if (!associate_outcome.IsSuccess())
    {
        std::cout << "Failed to associate elastic ip address" << allocation_id
            << " with instance " << instance_id << ":" <<
            associate_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully associated elastic ip address " << allocation_id
        << " with instance " << instance_id << 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 ec2;
```

```
        Aws::EC2::Model::DescribeAddressesRequest request;
        auto outcome = ec2.DescribeAddresses(request);
        if (outcome.IsSuccess())
        {
            std::cout << std::left << std::setw(20) << "InstanceId" <<
                std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
                std::setw(20) << "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(20) << address.GetAllocationId() << std::setw(25)
                    << address.GetNetworkInterfaceId() << std::endl;
            }
        }
        else
        {
            std::cout << "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::Client::ClientConfiguration config;
        config.region = Aws::Region::US_WEST_2;

        Aws::EC2::EC2Client ec2(config);

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

        auto outcome = ec2.ReleaseAddress(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to release elastic ip address " <<
                allocation_id << ":" << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
```

```
            std::cout << "Successfully released elastic ip address " <<
                allocation_id << 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 for Linux Instances
- 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 ec2;
```

```
        Aws::EC2::Model::DescribeRegionsRequest request;
        auto outcome = ec2.DescribeRegions(request);
        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::cout << "Failed to describe regions:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
```

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 = ec2.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::cout << "Failed to describe availability zones:" <<
            describe_outcome.GetError().GetMessage() << std::endl;
    }
```

See the complete example.

More Information

- Regions and Availability Zones in the Amazon EC2 User Guide for Linux Instances
- 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 ec2;
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(pair_name);

    auto outcome = ec2.CreateKeyPair(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create key pair:" <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully created key pair named " <<
```

```
                pair_name << 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 ec2;
        Aws::EC2::Model::DescribeKeyPairsRequest request;

        auto outcome = ec2.DescribeKeyPairs(request);
        if (outcome.IsSuccess())
        {
            std::cout << std::left <<
                std::setw(32) << "Name" <<
                std::setw(64) << "Fingerprint" << std::endl;

            const auto &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::cout << "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 ec2;
        Aws::EC2::Model::DeleteKeyPairRequest request;

        request.SetKeyName(pair_name);
        auto outcome = ec2.DeleteKeyPair(request);

        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete key pair " << pair_name <<
                ":" << outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted key pair named " << pair_name <<
                std::endl;
        }
```

See the complete example.

More Information

- Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances
- 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 ec2;
    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(group_name);
    request.SetDescription(description);
    request.SetVpcId(vpc_id);

    auto outcome = ec2.CreateSecurityGroup(request);

    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create security group:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created security group named " << group_name <<
        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 authorize_request;

    authorize_request.SetGroupName(group_name);
```

```
    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);
```

```
    auto ingress_req = ec2.AuthorizeSecurityGroupIngress(authorize_request);

    if (!ingress_req.IsSuccess())
    {
        std::cout << "Failed to set ingress policy for security group " <<
            group_name << ":" << ingress_req.GetError().GetMessage() <<
            std::endl;
        return;
    }

    std::cout << "Successfully added ingress policy to security group " <<
        group_name << 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 ec2;
        Aws::EC2::Model::DescribeSecurityGroupsRequest request;

        if (argc == 2)
        {
            request.AddGroupIds(argv[1]);
        }

        auto outcome = ec2.DescribeSecurityGroups(request);

        if (outcome.IsSuccess())
        {
            std::cout << std::left <<
                std::setw(32) << "Name" <<
                std::setw(20) << "GroupId" <<
                std::setw(20) << "VpcId" <<
                std::setw(64) << "Description" << std::endl;

            const auto &securityGroups =
                outcome.GetResult().GetSecurityGroups();

            for (const auto &securityGroup : securityGroups)
            {
```

```
                std::cout << std::left <<
                    std::setw(32) << securityGroup.GetGroupName() <<
                    std::setw(20) << securityGroup.GetGroupId() <<
                    std::setw(20) << securityGroup.GetVpcId() <<
                    std::setw(64) << securityGroup.GetDescription() <<
                    std::endl;
            }
        }
        else
        {
            std::cout << "Failed to describe security groups:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
```

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 <iomanip>
#include <iostream>
```

Code

```
        Aws::EC2::EC2Client ec2;
        Aws::EC2::Model::DeleteSecurityGroupRequest request;

        request.SetGroupId(groupId);
        auto outcome = ec2.DeleteSecurityGroup(request);

        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete security group " << groupId <<
                ":" << outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted security group " << groupId <<
                std::endl;
        }
```

See the complete example.

More Information

- Amazon EC2 Security Groups in the Amazon EC2 User Guide for Linux Instances
- Authorizing Inbound Traffic for Your Linux Instances in the Amazon EC2 User Guide for Linux Instances
- 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

IAM code examples using the AWS SDK for C++

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. You can use the following examples to program IAM 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 IAM Access Keys

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

Create an Access Key

To create an IAM access key, call the IAMClient's `CreateAccessKey` function with an CreateAccessKeyRequest object.

You must set the user name using the `CreateAccessKeyRequest`'s `WithUserName` setter function before passing it to the `CreateAccessKey` function.

Includes:

```
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/CreateAccessKeyRequest.h>
#include <aws/iam/model/CreateAccessKeyResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

List Access Keys

To list the access keys for a given user, create a ListAccessKeysRequest object that contains the user name to list keys for, and pass it to the IAMClient's `ListAccessKeys` function.

> **Note**
> If you do not supply a user name to `ListAccessKeys`, it will attempt to list access keys associated with the AWS account that signed the request.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListAccessKeysRequest.h>
#include <aws/iam/model/ListAccessKeysResult.h>
#include <iomanip>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

The results of `ListAccessKeys` are paged (with a default maximum of 100 records per call). You can call `GetIsTruncated` on the returned ListAccessKeysResult object to see if the query returned fewer results then are available. If so, then call `SetMarker` on the `ListAccessKeysRequest` and pass it back to the next invocation of `ListAccessKeys`.

See the complete example.

Retrieve an Access Key's Last Used Time

To get the time an access key was last used, call the IAMClient's `GetAccessKeyLastUsed` function with the access key's ID (which can be passed in using a GetAccessKeyLastUsedRequest object, or directly to the overload that takes the access key ID directly.

You can then use the returned GetAccessKeyLastUsedResult object to retrieve the key's last used time.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/GetAccessKeyLastUsedRequest.h>
#include <aws/iam/model/GetAccessKeyLastUsedResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

Activate or Deactivate Access Keys

You can activate or deactivate an access key by creating an UpdateAccessKeyRequest object, providing the access key ID, optionally the user name, and the desired `Status` Type, then passing the request object to the IAMClient's `UpdateAccessKey` function.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/UpdateAccessKeyRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

Delete an Access Key

To permanently delete an access key, call the IAMClient's `DeleteKey` function, providing it with a DeleteAccessKeyRequest containing the access key's ID and username.

> **Note**
> Once deleted, a key can no longer be retrieved or used. To temporarily deactivate a key so that it can be activated again later, use updateAccessKey (p. 89) function instead.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DeleteAccessKeyRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

More Information

- CreateAccessKey in the IAM API Reference
- ListAccessKeys in the IAM API Reference
- GetAccessKeyLastUsed in the IAM API Reference
- UpdateAccessKey in the IAM API Reference
- DeleteAccessKey in the IAM API Reference

Managing IAM Users

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

Create a User

Use the IAMClient `CreateUser` function, passing it a CreateUserRequest with the name of the user to create.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/CreateUserRequest.h>
#include <aws/iam/model/GetUserRequest.h>
#include <aws/iam/model/GetUserResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
    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();
```

Get Information About a User

To get information about a particular user, such as the user's creation date, path, ID or ARN, call the IAMClient `GetUser` function with a GetUserRequest containing the user name. If successful, you can get the User from the returned GetUserResult outcome.

If the user doesn't already exist, `GetUser` will fail with `Aws::IAM::IAMErrors::NO_SUCH_ENTITY`.

Includes:

```
#include <aws/iam/model/GetUserRequest.h>
#include <aws/iam/model/GetUserResult.h>
```

Code:

```
    Aws::IAM::IAMClient iam(clientConfig);
```

```
    Aws::IAM::Model::GetUserRequest get_request;
    get_request.SetUserName(userName);
```

```
    auto get_outcome = iam.GetUser(get_request);
    if (get_outcome.IsSuccess()) {
        std::cout << "IAM user " << userName << " already exists" << std::endl;
        return true;
    }
    else if (get_outcome.GetError().GetErrorType() !=
             Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
        std::cerr << "Error checking existence of IAM user " << userName << ":"
                  << get_outcome.GetError().GetMessage() << std::endl;
        return false;
    }
```

See the complete example.

List Users

List the existing IAM users for your account by calling the IAMClient `ListUsers` function, passing it a ListUsersRequest object. The list of users is returned in a ListUsersResult object that you can use to get information about the users.

The result may be paginated; to check to see if there are more results available, check the value of `GetResult().GetIsTruncated()`. If `true`, then set a marker on the request and call `ListUsers` again to get the next batch of users. This code demonstrates the technique.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListUsersRequest.h>
#include <iomanip>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

Update a User

To update an existing user, create an UpdateUserRequest and pass it to the IAMClient `UpdateUser` member function.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/UpdateUserRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

Delete a User

To delete an existing user, call the IAMClient `DeleteUser` function, passing it a DeleteUserRequest object containing the name of the user to delete.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DeleteUserRequest.h>
#include <aws/iam/model/GetUserRequest.h>
#include <aws/iam/model/GetUserResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
    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();
```

See the complete example.

Using IAM Account Aliases

If you want the URL for your sign-in page to contain your company name or other friendly identifier instead of your AWS account ID, you can create an alias for your AWS account.

Note
AWS supports exactly one account alias per account.

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

Create an Account Alias

To create an account alias, call the IAMClient's `CreateAccountAlias` function with a CreateAccountAliasRequest object that contains the alias name.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/CreateAccountAliasRequest.h>
```

```
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

List Account Aliases

To list your account's alias, if any, call the IAMClient's `ListAccountAliases` function. It takes a ListAccountAliasesRequest object.

> **Note**
> The returned ListAccountAliasesResult supports the same `GetIsTruncated` and `GetMarker` functions as other AWS SDK for C++*list* functions, but an AWS account can have only *one* account alias.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListAccountAliasesRequest.h>
#include <aws/iam/model/ListAccountAliasesResult.h>
#include <iomanip>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

see the complete example.

Delete an Account Alias

To delete your account's alias, call the IAMClient's `DeleteAccountAlias` function. When deleting an account alias, you must supply its name using a DeleteAccountAliasRequest object.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DeleteAccountAliasRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

More Information

- Your AWS Account ID and Its Alias in the IAM User Guide
- CreateAccountAlias in the IAM API Reference
- ListAccountAliases in the IAM API Reference
- DeleteAccountAlias in the IAM API Reference

Working with IAM Policies

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

Create a Policy

To create a new policy, provide the policy's name and a JSON-formatted policy document in a CreatePolicyRequest to the IAMClient's `CreatePolicy` function.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/CreatePolicyRequest.h>
#include <aws/iam/model/CreatePolicyResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

IAM policy documents are JSON strings with a well-documented syntax. Here is an example that provides access to make particular requests to DynamoDB. It takes the policy ARN as a passed-in variable.

```
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();
}
```

See the complete example.

Retrieve a Policy

To retrieve an existing policy, call the IAMClient's `GetPolicy` function, providing the policy's ARN within a GetPolicyRequest object.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/GetPolicyRequest.h>
#include <aws/iam/model/GetPolicyResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

Delete a Policy

To delete a policy, provide the policy's ARN in a DeletePolicyRequest to the IAMClient's `DeletePolicy` function.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DeletePolicyRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

Attach a Policy

You can attach a policy to an IAMrole by calling the IAMClient's `AttachRolePolicy` function, providing it with the role name and policy ARN in an AttachRolePolicyRequest.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/AttachRolePolicyRequest.h>
#include <aws/iam/model/ListAttachedRolePoliciesRequest.h>
#include <aws/iam/model/ListAttachedRolePoliciesResult.h>
#include <iostream>
#include <iomanip>
#include "iam_samples.h"
```

Code:

```
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();
}
```

See the complete example.

List Attached Policies

List attached policies on a role by calling the IAMClient's `ListAttachedRolePolicies` function. It takes a ListAttachedRolePoliciesRequest object that contains the role name to list the policies for.

Call `GetAttachedPolicies` on the returned ListAttachedRolePoliciesResult object to get the list of attached policies. Results may be truncated; if the `ListAttachedRolePoliciesResult` object's `GetIsTruncated` function returns `true`, call the `ListAttachedRolePoliciesRequest` object's `SetMarker` function and use it to call `ListAttachedRolePolicies` again to get the next batch of results.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <aws/iam/model/ListPoliciesResult.h>
#include <iomanip>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

Detach a Policy

To detach a policy from a role, call the IAMClient's `DetachRolePolicy` function, providing it with the role name and policy ARN in a DetachRolePolicyRequest.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DetachRolePolicyRequest.h>
#include <aws/iam/model/ListAttachedRolePoliciesRequest.h>
#include <aws/iam/model/ListAttachedRolePoliciesResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
    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();
```

See the complete example.

More Information

- Overview of IAM Policies in the IAM User Guide.
- IAM Policy Reference in the IAM User Guide.
- CreatePolicy in the IAM API Reference
- GetPolicy in the IAM API Reference
- DeletePolicy in the IAM API Reference

- AttachGroupPolicy, AttachRolePolicy and AttachUserPolicy in the IAM API Reference
- DetachGroupPolicy, DetachRolePolicy and DetachUserPolicy in the IAM API Reference
- ListAttachedGroupPolicies, ListAttachedRolePolicies and ListAttachedUserPolicies in the IAM API Reference

Working with IAM Server Certificates

To enable HTTPS connections to your website or application on AWS, you need an SSL/TLS *server certificate*. You can use a server certificate provided by AWS Certificate Manager or one that you obtained from an external provider.

We recommend that you use ACM to provision, manage, and deploy your server certificates. With ACM you can request a certificate, deploy it to your AWS resources, and let ACM handle certificate renewals for you. Certificates provided by ACM are free. For more information about ACM, see the AWS Certificate Manager User Guide.

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

Retrieve a Server Certificate

You can retrieve a server certificate by calling the IAMClient's `GetServerCertificate` function, passing it a GetServerCertificateRequest with the certificate's name.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/GetServerCertificateRequest.h>
#include <aws/iam/model/GetServerCertificateResult.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

List Server Certificates

To list your server certificates, call the IAMClient's `ListServerCertificates` function with a ListServerCertificatesRequest. It returns a ListServerCertificatesResult.

Call the returned `ListServerCertificateResult` object's `GetServerCertificateMetadataList` function to get a list of ServerCertificateMetadata objects that you can use to get information about each certificate.

Results may be truncated; if the `ListServerCertificateResult` object's `GetIsTruncated` function returns `true`, call the `ListServerCertificatesRequest` object's `SetMarker` function and use it to call `listServerCertificates` again to get the next batch of results.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListServerCertificatesRequest.h>
#include <iomanip>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

Update a Server Certificate

You can update a server certificate's name or path by calling the IAMClient's `UpdateServerCertificate` function. It takes a UpdateServerCertificateRequest object set with the server certificate's current name and either a new name or new path to use.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/UpdateServerCertificateRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

Delete a Server Certificate

To delete a server certificate, call the IAMClient's `DeleteServerCertificate` function with a DeleteServerCertificateRequest containing the certificate's name.

Includes:

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/DeleteServerCertificateRequest.h>
#include <iostream>
#include "iam_samples.h"
```

Code:

```
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;
}
```

See the complete example.

More Information

- Working with Server Certificates in the IAM User Guide
- GetServerCertificate in the IAM API Reference
- ListServerCertificates in the IAM API Reference
- UpdateServerCertificate in the IAM API Reference
- DeleteServerCertificate in the IAM API Reference
- AWS Certificate Manager User Guide

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 (p. 108)
  - Operations on objects (p. 110) – Uploading and downloading data objects
  - Managing Amazon S3 Access Permissions (p. 114)
  - Managing Access to Amazon S3 Buckets Using Bucket Policies (p. 122)
  - Configuring an Amazon S3 Bucket as a Website (p. 125)
- `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 (p. 128)
- `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 (p. 127)

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 (p. 108).

Prerequisites

Before you begin, we recommend you read Getting started using the AWS SDK for C++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 (p. 44)

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 (p. 44)

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::ListObjectsRequest request;
    request.WithBucket(bucketName);

    auto outcome = s3_client.ListObjects(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListObjects: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

        for (Aws::S3::Model::Object &object: objects) {
            std::cout << object.GetKey() << std::endl;
        }
    }
```

```
    return outcome.IsSuccess();
}
```

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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""           },\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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 &indexPage, const Aws::String
 &errorPage,
                                  const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    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 (p. 122).

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 AbortMultipartUpload 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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;

        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());
            g_file_size = uploadHandle->GetBytesTotalSize();

            // Create buffer to hold data received by the data stream.
            Aws::Utils::Array<unsigned char> buffer(BUFFER_SIZE);
            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",
 Aws::New<Stream::PreallocatedStreamBuf>("TestTag", buffer.GetUnderlyingData(),
 BUFFER_SIZE));
                });
            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 AbortMultipartUpload 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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=gsk8vDbmnlA5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8xzZI1P0zp
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=gsk8vDbmnlA5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8xzZI1P0zp
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

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 <aws/sqs/model/CreateQueueResult.h>
#include <iostream>
```

Code

```
        Aws::SQS::SQSClient sqs;

        Aws::SQS::Model::CreateQueueRequest cq_req;
        cq_req.SetQueueName(queue_name);

        auto cq_out = sqs.CreateQueue(cq_req);
        if (cq_out.IsSuccess())
        {
            std::cout << "Successfully created queue " << queue_name << std::endl;
        }
        else
        {
            std::cout << "Error creating queue " << queue_name << ": " <<
                cq_out.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 <aws/sqs/model/ListQueuesResult.h>
#include <iostream>
```

Code

```
        Aws::SQS::SQSClient sqs;

        Aws::SQS::Model::ListQueuesRequest lq_req;

        auto lq_out = sqs.ListQueues(lq_req);
        if (lq_out.IsSuccess())
        {
            std::cout << "Queue Urls:" << std::endl << std::endl;
            const auto &queue_urls = lq_out.GetResult().GetQueueUrls();
            for (const auto &iter : queue_urls)
            {
                std::cout << " " << iter << std::endl;
            }
        }
        else
        {
            std::cout << "Error listing queues: " <<
                lq_out.GetError().GetMessage() << 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 <aws/sqs/model/GetQueueUrlResult.h>
#include <iostream>
```

Code

```
        Aws::SQS::SQSClient sqs;

        Aws::SQS::Model::GetQueueUrlRequest gqu_req;
        gqu_req.SetQueueName(queue_name);

        auto gqu_out = sqs.GetQueueUrl(gqu_req);
        if (gqu_out.IsSuccess()) {
            std::cout << "Queue " << queue_name << " has url " <<
            gqu_out.GetResult().GetQueueUrl() << std::endl;
        } else {
            std::cout << "Error getting url for queue " << queue_name << ": " <<
                gqu_out.GetError().GetMessage() << std::endl;
        }
```

See the complete example.

Delete a Queue

Provide the URL (p. 132) 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 dq_req;
        dq_req.SetQueueUrl(queue_url);

        auto dq_out = sqs.DeleteQueue(dq_req);
        if (dq_out.IsSuccess())
        {
            std::cout << "Successfully deleted queue with url " << queue_url <<
                std::endl;
        }
        else
        {
            std::cout << "Error deleting queue " << queue_url << ": " <<
                dq_out.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 (p. 130). 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 (p. 132), 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 <aws/sqs/model/SendMessageResult.h>
#include <iostream>
```

Code

```
        Aws::SQS::SQSClient sqs;

        Aws::SQS::Model::SendMessageRequest sm_req;
        sm_req.SetQueueUrl(queue_url);
        sm_req.SetMessageBody(msg_body);

        auto sm_out = sqs.SendMessage(sm_req);
        if (sm_out.IsSuccess())
        {
            std::cout << "Successfully sent message to " << queue_url <<
                std::endl;
        }
        else
        {
            std::cout << "Error sending message to " << queue_url << ": " <<
                sm_out.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 <aws/sqs/model/ReceiveMessageResult.h>
```

Code

```
    Aws::SQS::SQSClient sqs(client_cfg);

    Aws::SQS::Model::ReceiveMessageRequest rm_req;
    rm_req.SetQueueUrl(queue_url);
    rm_req.SetMaxNumberOfMessages(1);

    auto rm_out = sqs.ReceiveMessage(rm_req);
    if (!rm_out.IsSuccess())
    {
        std::cout << "Error receiving message from queue " << queue_url << ": "
            << rm_out.GetError().GetMessage() << std::endl;
        return;
    }

    const auto& messages = rm_out.GetResult().GetMessages();
    if (messages.size() == 0)
    {
        std::cout << "No messages received from queue " << queue_url <<
            std::endl;
        return;
    }

    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;
```

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/sqs/model/DeleteMessageRequest.h>
```

Code

```
    Aws::SQS::Model::DeleteMessageRequest dm_req;
    dm_req.SetQueueUrl(queue_url);
    dm_req.SetReceiptHandle(message.GetReceiptHandle());

    auto dm_out = sqs.DeleteMessage(dm_req);
    if (dm_out.IsSuccess())
    {
        std::cout << "Successfully deleted message " << message.GetMessageId()
            << " from queue " << queue_url << std::endl;
    }
    else
    {
        std::cout << "Error deleting message " << message.GetMessageId() <<
            " from queue " << queue_url << ": " <<
            dm_out.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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 <aws/sqs/model/CreateQueueResult.h>
#include <iostream>
```

Code

```
        Aws::SQS::SQSClient sqs;

        Aws::SQS::Model::CreateQueueRequest request;
        request.SetQueueName(queue_name);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            poll_time);

        auto outcome = sqs.CreateQueue(request);
        if (outcome.IsSuccess())
        {
            std::cout << "Successfully created queue " << queue_name <<
                std::endl;
        }
        else
        {
            std::cout << "Error creating queue " << queue_name << ": " <<
                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 sqs;

        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(queue_url);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            poll_time);

        auto outcome = sqs.SetQueueAttributes(request);
        if (outcome.IsSuccess())
        {
            std::cout << "Successfully updated long polling time for queue " <<
                queue_url << " to " << poll_time << std::endl;
        }
        else
        {
            std::cout << "Error updating long polling time for queue " <<
                queue_url << ": " << 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>
#include <aws/sqs/model/ReceiveMessageResult.h>
```

Code

```
    Aws::SQS::SQSClient sqs(client_cfg);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queue_url);
    request.SetMaxNumberOfMessages(1);
    request.SetWaitTimeSeconds(wait_time);

    auto outcome = sqs.ReceiveMessage(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Error receiving message from queue " << queue_url << ": "
            << outcome.GetError().GetMessage() << std::endl;
        return;
    }

    const auto& messages = outcome.GetResult().GetMessages();
    if (messages.size() == 0)
    {
        std::cout << "No messages received from queue " << queue_url <<
            std::endl;
        return;
    }

    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;
```

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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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 <aws/sqs/model/ReceiveMessageResult.h>
#include <iostream>
```

Code

```
    Aws::SQS::Model::ChangeMessageVisibilityRequest request;
    request.SetQueueUrl(queue_url);
    request.SetReceiptHandle(message.GetReceiptHandle());
    request.SetVisibilityTimeout(visibility_timeout);
    auto outcome = sqs.ChangeMessageVisibility(request);
    if (outcome.IsSuccess())
    {
        std::cout << "Successfully changed visibility of message " <<
            message.GetMessageId() << " from queue " << queue_url << std::endl;
    }
    else
    {
        std::cout << "Error changing visibility of message " <<
            message.GetMessageId() << " 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++ (p. 2).

Download the example code and build the solution as described in Getting started on code examples (p. 40).

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 (p. 2).

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& queue_arn, int max_msg)
{
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queue_arn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(max_msg);

    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(src_queue_url);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

        auto outcome = sqs.SetQueueAttributes(request);
        if (outcome.IsSuccess())
        {
            std::cout << "Successfully set dead letter queue for queue  " <<
                src_queue_url << " to " << queue_arn << std::endl;
        }
        else
        {
            std::cout << "Error setting dead letter queue for queue " <<
                src_queue_url << ": " << 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

Additional code examples for the AWS SDK for C++

The code examples in this topic show you how to use the AWS SDK for C++ with AWS.

Actions are code excerpts that show you how to call individual service functions.

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++ (p. 141)
- Cross-service examples using SDK for C++ (p. 226)

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 that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Services

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 that show you how to call individual CloudWatch functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple CloudWatch functions.

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

Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

SDK for C++

> **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*.

Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

SDK for C++

> **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*.

Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

SDK for C++

> **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*.

Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

SDK for C++

> **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*.

Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

SDK for C++

> **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*.

List metrics

The following code example shows how to list Amazon CloudWatch metrics.

SDK for C++

> **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*.

Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

SDK for C++

> **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 that show you how to call individual CloudWatch Logs functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple CloudWatch Logs functions.

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

Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

SDK for C++

> **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*.

Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

SDK for C++

> **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*.

Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

SDK for C++

> **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*.

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 that show you how to call individual DynamoDB functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple DynamoDB functions.

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

Create a table

The following code example shows how to create a DynamoDB 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;
        if (!region.empty())
            clientConfig.region = region;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        std::cout << "Creating table " << table <<
            " with a simple primary key: \"Name\"" << std::endl;

        Aws::DynamoDB::Model::CreateTableRequest req;

        Aws::DynamoDB::Model::AttributeDefinition haskKey;
        haskKey.SetAttributeName("Name");
        haskKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
        req.AddAttributeDefinitions(haskKey);

        Aws::DynamoDB::Model::KeySchemaElement keyscelt;

keyscelt.WithAttributeName("Name").WithKeyType(Aws::DynamoDB::Model::KeyType::HASH);
        req.AddKeySchema(keyscelt);

        Aws::DynamoDB::Model::ProvisionedThroughput thruput;
        thruput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
        req.SetProvisionedThroughput(thruput);
        req.SetTableName(table);
```

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

- For API details, see CreateTable in *AWS SDK for C++ API Reference*.

Delete a table

The following code example shows how to delete a DynamoDB 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;
        if (!region.empty())
            clientConfig.region = region;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        Aws::DynamoDB::Model::DeleteTableRequest dtr;
        dtr.SetTableName(table);

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

- For API details, see DeleteTable in *AWS SDK for C++ API Reference*.

Get an item from a table

The following code example shows how to get an item from a DynamoDB 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;
```

```
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
        Aws::DynamoDB::Model::GetItemRequest req;

        // Set up the request.
        req.SetTableName(table);
        Aws::DynamoDB::Model::AttributeValue hashKey;
        hashKey.SetS(keyval);
        req.AddKey(key, hashKey);

        // Retrieve the item's fields and values
        const Aws::DynamoDB::Model::GetItemOutcome& result = dynamoClient.GetItem(req);
        if (result.IsSuccess())
        {
            // Reference the retrieved fields/values.
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>& item =
result.GetResult().GetItem();
            if (item.size() > 0)
            {
                // 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 " << key << std::endl;
            }
        }
        else
        {
            std::cout << "Failed to get item: " << result.GetError().GetMessage();
        }
```

- For API details, see GetItem in *AWS SDK for C++ API Reference*.

Get information about a table

The following code example shows how to get information about a DynamoDB 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;
        if (!region.empty())
            clientConfig.region = region;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        Aws::DynamoDB::Model::DescribeTableRequest dtr;
        dtr.SetTableName(table);

        const Aws::DynamoDB::Model::DescribeTableOutcome& result =
dynamoClient.DescribeTable(dtr);

        if (result.IsSuccess())
        {
            const Aws::DynamoDB::Model::TableDescription& td =
result.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::cout << "Failed to describe table: " <<
result.GetError().GetMessage();
        }
```

- For API details, see DescribeTable in *AWS SDK for C++ API Reference*.

List tables

The following code example shows how to list DynamoDB tables.

SDK for C++

> **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;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do
        {
            const Aws::DynamoDB::Model::ListTablesOutcome& lto =
dynamoClient.ListTables(listTablesRequest);
            if (!lto.IsSuccess())
            {
                std::cout << "Error: " << lto.GetError().GetMessage() << std::endl;
                return 1;
            }

            for (const auto& s : lto.GetResult().GetTableNames())
                std::cout << s << std::endl;

listTablesRequest.SetExclusiveStartTableName(lto.GetResult().GetLastEvaluatedTableName());
```

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

- For API details, see ListTables in *AWS SDK for C++ API Reference*.

Put an item in a table

The following code example shows how to put an item in a DynamoDB 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;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

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

        Aws::DynamoDB::Model::AttributeValue av;
        av.SetS(keyVal);

        Aws::DynamoDB::Model::AttributeValue album;
        album.SetS(AlbumTitleValue);

        Aws::DynamoDB::Model::AttributeValue awards;
        awards.SetS(AwardVal);

        Aws::DynamoDB::Model::AttributeValue song;
        song.SetS(SongTitleVal);

        // Add all AttributeValue objects.
        putItemRequest.AddItem(key, av);
        putItemRequest.AddItem(albumTitle, album);
        putItemRequest.AddItem(Awards, awards);
        putItemRequest.AddItem(SongTitle, song);

        const Aws::DynamoDB::Model::PutItemOutcome result =
dynamoClient.PutItem(putItemRequest);
        if (!result.IsSuccess())
        {
            std::cout << result.GetError().GetMessage() << std::endl;
            return 1;
        }
        std::cout << "Successfully added Item!" << std::endl;
```

- For API details, see PutItem in *AWS SDK for C++ API Reference*.

Query a table

The following code example shows how to query a DynamoDB 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;

        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
        Aws::DynamoDB::Model::QueryRequest req;

        req.SetTableName(table);

        // Set query key condition expression
        req.SetKeyConditionExpression(partitionKeyAttributeName + "= :valueToMatch");

        // Set Expression AttributeValues
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
        attributeValues.emplace(":valueToMatch", partitionKeyAttributeValue);

        req.SetExpressionAttributeValues(attributeValues);

        // Perform Query operation
        const Aws::DynamoDB::Model::QueryOutcome& result = dynamoClient.Query(req);
        if (result.IsSuccess())
        {
            // Reference the retrieved items
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();
            if(items.size() > 0)
            {
                std::cout << "Number of items retrieved from Query: " << items.size()
<< std::endl;
                //Iterate each item and print
                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: " << table << std::endl;
            }
        }
        else
        {
            std::cout << "Failed to Query items: " << result.GetError().GetMessage();
        }
```

- For API details, see Query in *AWS SDK for C++ API Reference*.

Scan a table

The following code example shows how to scan a DynamoDB 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;
```

```
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
        Aws::DynamoDB::Model::ScanRequest req;
        req.SetTableName(table);

        if (!projection.empty())
            req.SetProjectionExpression(projection);

        // Perform scan on table
        const Aws::DynamoDB::Model::ScanOutcome& result = dynamoClient.Scan(req);
        if (result.IsSuccess())
        {
            // Reference the retrieved items
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();
            if(items.size() > 0)
            {
                std::cout << "Number of items retrieved from scan: " << items.size() <<
std::endl;
                //Iterate each item and print
                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: " << table << std::endl;
            }
        }
        else
        {
            std::cout << "Failed to Scan items: " << result.GetError().GetMessage();
        }
```

- For API details, see Scan in *AWS SDK for C++ API Reference*.

Update an item in a table

The following code example shows how to update an item in a DynamoDB 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;
        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

        // *** 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(keyValue);
        request.AddKey("id", attribValue);

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

        // Parse the attribute name and value. Syntax: "name=value".
        auto parsed = Aws::Utils::StringUtils::Split(attributeNameAndValue, '=');

        if (parsed.size() != 2)
        {
            std::cout << "Invalid argument syntax: " << attributeNameAndValue << USAGE;
            return 1;
        }

        // Construct attribute name argument
        // Note: Setting the ExpressionAttributeNames argument is required only
        // when the name is a reserved word, such as "default". Otherwise, the
        // name can be included in the update_expression, as in
        // "SET MyAttributeName = :valueA"
        Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
        expressionAttributeNames["#a"] = parsed[0];
        request.SetExpressionAttributeNames(expressionAttributeNames);

        // Construct attribute value argument.
        Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
        attributeUpdatedValue.SetS(parsed[1]);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
        expressionAttributeValues[":valueA"] = attributeUpdatedValue;
        request.SetExpressionAttributeValues(expressionAttributeValues);

        // Update the item.
        const Aws::DynamoDB::Model::UpdateItemOutcome& result =
dynamoClient.UpdateItem(request);
        if (!result.IsSuccess())
        {
            std::cout << result.GetError().GetMessage() << std::endl;
            return 1;
        }
        std::cout << "Item was updated" << std::endl;
```

- For API details, see UpdateItem in *AWS SDK for C++ API Reference*.

Scenarios

Get started using 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.
- 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.

```
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<
 std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<
 std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    // 1. Create a table.
    {
        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, dynamoClient)) {
            deleteDynamoTable(MOVIE_TABLE_NAME, dynamoClient);
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    // 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> plotAttibute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttibute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttibute);

        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();
            deleteDynamoTable(MOVIE_TABLE_NAME, dynamoClient);
            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 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;
           deleteDynamoTable(MOVIE_TABLE_NAME, dynamoClient);
           return false;
       }
   }

   std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

   // 4. Put 250 movies in the table from moviedata.json.
   if (!movieTableAlreadyExisted) {
       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::JsonView> 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 =
movieJsonViewToAttributeMap(
                       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(ASTERIX_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();
       }

   } 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::cout << "Failed to Scan movies: "
                          << result.GetError().GetMessage();
            }
        } 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;
            }
        }
    }

    // 9.Delete the table. (DeleteTable)
    return deleteDynamoTable(MOVIE_TABLE_NAME, dynamoClient);
}

bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                         const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {
    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();
    }

    return result.IsSuccess();
}

bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {
    // 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;
}
```

```
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;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

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 that show you how to call individual EventBridge functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple EventBridge functions.

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

Add a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon EventBridge event.

SDK for C++

> **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*.

Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

SDK for C++

> **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*.

Send events

The following code example shows how to send Amazon EventBridge events.

SDK for C++

> **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*.

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 that show you how to call individual IAM functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple IAM functions.

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

Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

SDK for C++

> **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*.

Attach an inline policy to a role

The following code example shows how to attach an inline policy to an IAM role.

SDK for C++

> **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*.

Create a policy

The following code example shows how to create an IAM policy.

SDK for C++

> **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*.

Create a role

The following code example shows how to create an IAM role.

SDK for C++

> **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*.

Create a user

The following code example shows how to create an IAM user.

SDK for C++

> **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*.

Create an access key

The following code example shows how to create an IAM access key.

SDK for C++

> **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*.

Create an alias for an account

The following code example shows how to create an alias for an IAM account.

SDK for C++

> **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*.

Delete a policy

The following code example shows how to delete an IAM policy.

SDK for C++

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*.

Delete a server certificate

The following code example shows how to delete an IAM server certificate.

SDK for C++

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*.

Delete a user

The following code example shows how to delete an IAM user.

SDK for C++

> **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*.

Delete an access key

The following code example shows how to delete an IAM access key.

SDK for C++

> **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*.

Delete an account alias

The following code example shows how to delete an IAM account alias.

SDK for C++

> **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*.

Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

SDK for C++

> **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*.

Get a policy

The following code example shows how to get an IAM policy.

SDK for C++

> **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*.

Get a server certificate

The following code example shows how to get an IAM server certificate.

SDK for C++

> **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*.

Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

SDK for C++

> **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*.

List a user's access keys

The following code example shows how to list a user's IAM access keys.

SDK for C++

> **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*.

List account aliases

The following code example shows how to list IAM account aliases.

SDK for C++

> **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*.

List policies

The following code example shows how to list IAM policies.

SDK for C++

> **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*.

List server certificates

The following code example shows how to list IAM server certificates.

SDK for C++

> **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*.

List users

The following code example shows how to list IAM users.

SDK for C++

> **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*.

Update a server certificate

The following code example shows how to update an IAM server certificate.

SDK for C++

> **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*.

Update a user

The following code example shows how to update an IAM user.

SDK for C++

> **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*.

Update an access key

The following code example shows how to update an IAM access key.

SDK for C++

> **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*.

Scenarios

Create a user and assume a role

The following code example shows how to:

- Create a user who has 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 Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

SDK for C++

> **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);
    }
}

//! 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::STSClient 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()),
                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 20 times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome = s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > 20) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after 20 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

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 that show you how to call individual Amazon S3 functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon S3 functions.

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

Add a policy to a bucket

The following code example shows how to add a policy to an S3 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.

```
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""          },\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*.

Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

SDK for C++

> **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*.

Create a bucket

The following code example shows how to create an S3 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.

```
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*.

Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 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.

```
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*.

Delete an empty bucket

The following code example shows how to delete an empty S3 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.

```
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*.

Delete an object

The following code example shows how to delete an S3 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.

```
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*.

Delete the website configuration from a bucket

The following code example shows how to delete the website configuration from an S3 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.

```
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*.

Get an object from a bucket

The following code example shows how to read data from an object in an S3 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.

```
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*.

Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 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.

```
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 GetObjectAcl in *AWS SDK for C++ API Reference*.

Get the policy for a bucket

The following code example shows how to get the policy for an S3 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.

```
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*.

Get the website configuration for a bucket

The following code example shows how to get the website configuration for an S3 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.

```
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*.

List buckets

The following code example shows how to list S3 buckets.

SDK for C++

> **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*.

List objects in a bucket

The following code example shows how to list objects in an S3 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.

```
bool AwsDoc::S3::ListObjects(const Aws::String &bucketName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::ListObjectsRequest request;
    request.WithBucket(bucketName);

    auto outcome = s3_client.ListObjects(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListObjects: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

        for (Aws::S3::Model::Object &object: objects) {
            std::cout << object.GetKey() << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

- For API details, see ListObjects in *AWS SDK for C++ API Reference*.

Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 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.

```
bool AwsDoc::S3::PutWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    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*.

Upload an object to a bucket

The following code example shows how to upload an object to an S3 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.

```
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*.

Scenarios

Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a 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/ListObjectsRequest.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 outStream(saveFilePath);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath << "'." <<
std::endl;
            }
            else {
                outStream << 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::ListObjectsRequest request;
        request.WithBucket(bucketName);

        Aws::S3::Model::ListObjectsOutcome outcome = client.ListObjects(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
        }
        else {
            Aws::Vector<Aws::S3::Model::Object> objects =
```

```
                    outcome.GetResult().GetContents();

            std::cout << objects.size() << " objects in the bucket, '" << bucketName <<
 "':" << std::endl;

            for (Aws::S3::Model::Object &object: objects) {
                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
- ListObjects
- PutObject

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 that show you how to call individual Amazon SNS functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon SNS functions.

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

Create a topic

The following code example shows how to create an Amazon SNS topic.

SDK for C++

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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::String topic_name = argv[1];
  Aws::SNS::SNSClient sns;

  Aws::SNS::Model::CreateTopicRequest ct_req;
  ct_req.SetName(topic_name);

  auto ct_out = sns.CreateTopic(ct_req);

  if (ct_out.IsSuccess())
  {
    std::cout << "Successfully created topic " << topic_name << std::endl;
  }
  else
  {
    std::cout << "Error creating topic " << topic_name << ":" <<
      ct_out.GetError().GetMessage() << std::endl;
  }
}
```

```
Aws::ShutdownAPI(options);
```

- For API details, see CreateTopic in *AWS SDK for C++ API Reference*.

Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::SNS::SNSClient sns;
  Aws::String subscription_arn = argv[1];

  Aws::SNS::Model::UnsubscribeRequest s_req;
  s_req.SetSubscriptionArn(subscription_arn);

  auto s_out = sns.Unsubscribe(s_req);

  if (s_out.IsSuccess())
  {
    std::cout << "Unsubscribed successfully " << std::endl;
  }
  else
  {
    std::cout << "Error while unsubscribing " << s_out.GetError().GetMessage()
      << std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see Unsubscribe in *AWS SDK for C++ API Reference*.

Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::String topic_arn = argv[1];
  Aws::SNS::SNSClient sns;
```

```
  Aws::SNS::Model::DeleteTopicRequest dt_req;
  dt_req.SetTopicArn(topic_arn);

  auto dt_out = sns.DeleteTopic(dt_req);

  if (dt_out.IsSuccess())
  {
    std::cout << "Successfully deleted topic " << topic_arn << std::endl;
  }
  else
  {
    std::cout << "Error deleting topic " << topic_arn << ":" <<
      dt_out.GetError().GetMessage() << std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see DeleteTopic in *AWS SDK for C++ API Reference*.

Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::SNS::SNSClient sns;
  Aws::String topic_arn = argv[1];

  Aws::SNS::Model::GetTopicAttributesRequest gta_req;
  gta_req.SetTopicArn(topic_arn);

  auto gta_out = sns.GetTopicAttributes(gta_req);

  if (gta_out.IsSuccess())
  {
    std::cout << "Topic Attributes:" << std::endl;
    for (auto const &attribute : gta_out.GetResult().GetAttributes())
    {
      std::cout << "  * " << attribute.first << " : " << attribute.second <<
std::endl;
    }
  }
  else
  {
    std::cout << "Error while getting Topic attributes " <<
gta_out.GetError().GetMessage()
       << std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see GetTopicAttributes in *AWS SDK for C++ API Reference*.

Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::SNS::SNSClient sns;

  Aws::SNS::Model::GetSMSAttributesRequest gsmst_req;
  //Set the request to only retrieve the DefaultSMSType setting.
  //Without the following line, GetSMSAttributes would retrieve all settings.
  gsmst_req.AddAttributes("DefaultSMSType");

  auto gsmst_out = sns.GetSMSAttributes(gsmst_req);

  if (gsmst_out.IsSuccess())
  {
      for (auto const& att : gsmst_out.GetResult().GetAttributes())
      {
          std::cout <<  att.first << ":  " <<  att.second << std::endl;
      }
  }
  else
  {
    std::cout << "Error while getting SMS Type: '" <<
gsmst_out.GetError().GetMessage()
      << "'" << std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see GetSMSAttributes in *AWS SDK for C++ API Reference*.

List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::SNS::SNSClient sns;
```

```
  Aws::SNS::Model::ListSubscriptionsRequest ls_req;

  auto ls_out = sns.ListSubscriptions(ls_req);

  if (ls_out.IsSuccess())
  {
      std::cout << "Subscriptions list:" << std::endl;
      for (auto const& subscription : ls_out.GetResult().GetSubscriptions())
      {
          std::cout << "  * " << subscription.GetSubscriptionArn() << std::endl;
      }
  }
  else
  {
    std::cout << "Error listing subscriptions " << ls_out.GetError().GetMessage() <<
      std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see ListSubscriptions in *AWS SDK for C++ API Reference*.

List topics

The following code example shows how to list Amazon SNS topics.

SDK for C++

> **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::SDKOptions options;
Aws::InitAPI(options);
{
  Aws::SNS::SNSClient sns;

  Aws::SNS::Model::ListTopicsRequest lt_req;

  auto lt_out = sns.ListTopics(lt_req);

  if (lt_out.IsSuccess())
  {
    std::cout << "Topics list:" << std::endl;
    for (auto const &topic : lt_out.GetResult().GetTopics())
    {
      std::cout << "  * " << topic.GetTopicArn() << std::endl;
    }
  }
  else
  {
    std::cout << "Error listing topics " << lt_out.GetError().GetMessage() <<
      std::endl;
  }
}

Aws::ShutdownAPI(options);
```

- For API details, see ListTopics in *AWS SDK for C++ API Reference*.

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 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
 */
int main(int argc, char ** argv)
{
  if (argc != 3)
  {
    std::cout << "Usage: publish_sms <message_value> <phone_number_value> " <<
 std::endl;
    return 1;
  }

  Aws::SDKOptions options;
  Aws::InitAPI(options);
  {
    Aws::SNS::SNSClient sns;
    Aws::String message = argv[1];
    Aws::String phone_number = argv[2];

    Aws::SNS::Model::PublishRequest psms_req;
    psms_req.SetMessage(message);
    psms_req.SetPhoneNumber(phone_number);

    auto psms_out = sns.Publish(psms_req);

    if (psms_out.IsSuccess())
    {
      std::cout << "Message published successfully " <<
 psms_out.GetResult().GetMessageId()
        << std::endl;
    }
    else
    {
      std::cout << "Error while publishing message " <<
 psms_out.GetError().GetMessage()
        << std::endl;
    }
  }
```

```
  Aws::ShutdownAPI(options);
  return 0;
}
```

- For API details, see Publish in *AWS SDK for C++ API Reference*.

Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

SDK for C++

> **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::SDKOptions options;
  Aws::InitAPI(options);
  {
    Aws::SNS::SNSClient sns;
    Aws::String message = argv[1];
    Aws::String topic_arn = argv[2];

    Aws::SNS::Model::PublishRequest psms_req;
    psms_req.SetMessage(message);
    psms_req.SetTopicArn(topic_arn);

    auto psms_out = sns.Publish(psms_req);

    if (psms_out.IsSuccess())
    {
      std::cout << "Message published successfully " << std::endl;
    }
    else
    {
      std::cout << "Error while publishing message " <<
psms_out.GetError().GetMessage()
        << std::endl;
    }
  }

  Aws::ShutdownAPI(options);
```

- For API details, see Publish in *AWS SDK for C++ API Reference*.

Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending 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.

How to use Amazon SNS to set the DefaultSMSType attribute.

```
 Aws::SDKOptions options;
 Aws::InitAPI(options);
 {
   Aws::SNS::SNSClient sns;
   Aws::String sms_type =  argv[1];

   Aws::SNS::Model::SetSMSAttributesRequest ssmst_req;
   ssmst_req.AddAttributes("DefaultSMSType", sms_type);

   auto ssmst_out = sns.SetSMSAttributes(ssmst_req);

   if (ssmst_out.IsSuccess())
   {
     std::cout << "SMS Type set successfully " << std::endl;
   }
   else
   {
       std::cout << "Error while setting SMS Type: '" <<
ssmst_out.GetError().GetMessage()
           << "'" << std::endl;
   }
 }

 Aws::ShutdownAPI(options);
```

- For API details, see SetSmsAttributes in *AWS SDK for C++ API Reference*.

Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

SDK for C++

> **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 AWS Lambda endpoint to a topic - demonstrates how to initiate a
 subscription to an Amazon SNS topic with delivery
 *  to an AWS Lambda function.
 *
 *  NOTE: You must first configure AWS Lambda to run this example.
 *         See https://docs.amazonaws.cn/en_us/lambda/latest/dg/with-sns-example.html
 for more information.
 *
 * <protocol_value> set to "lambda" provides delivery of JSON-encoded message to an AWS
 Lambda function
 *        (see https://docs.aws.amazon.com/sns/latest/api/API_Subscribe.html for
 available protocols).
 * <topic_arn_value> can be obtained from run_list_topics executable and includes the
 "arn:" prefix.
 * <lambda_function_arn> is the ARN of an AWS Lambda function.
 */

int main(int argc, char ** argv)
{
  if (argc != 4)
  {
```

```
    std::cout << "Usage: subscribe_lamda <protocol_value=lambda> <topic_arn_value>"
                 " <lambda_function_arn>" << std::endl;
    return 1;
  }

  Aws::SDKOptions options;
  Aws::InitAPI(options);
  {
    Aws::SNS::SNSClient sns;
    Aws::String protocol = argv[1];
    Aws::String topic_arn = argv[2];
    Aws::String endpoint = argv[3];

    Aws::SNS::Model::SubscribeRequest s_req;
    s_req.SetTopicArn(topic_arn);
    s_req.SetProtocol(protocol);
    s_req.SetEndpoint(endpoint);

    auto s_out = sns.Subscribe(s_req);

    if (s_out.IsSuccess())
    {
      std::cout << "Subscribed successfully " << std::endl;
    }
    else
    {
      std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
        << std::endl;
    }
  }

  Aws::ShutdownAPI(options);
  return 0;
}
```

- For API details, see Subscribe in *AWS SDK for C++ API Reference*.

Subscribe a mobile application to a topic

The following code example shows how to subscribe a mobile application endpoint so it receives notifications from an Amazon SNS topic.

SDK for C++

> **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 app endpoint to a topic - demonstrates how to initiate a subscription
 to an Amazon SNS topic
 * with delivery to a mobile app.
 *
 *  NOTE: You must first create an endpoint by registering an app and device.
 *     See https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-devicetoken.html
 for more information.
 *
 * <protocol_value> set to "application" provides delivery of JSON-encoded message to
 an EndpointArn
 *         for a mobile app and device (see https://docs.aws.amazon.com/sns/latest/api/
API_Subscribe.html for available protocols).
```

```
 * <topic_arn_value> can be obtained from run_list_topics executable and includes the
 "arn:" prefix.
 * <mobile_endpoint_arn> is the EndpointArn of a mobile app and device.
 */
int main(int argc, char ** argv)
{
  if (argc != 4)
  {
    std::cout << "Usage: subscribe_app <protocol_value/application> <topic_arn_value>"
                 " <mobile_endpoint_arn>" << std::endl;
    return 1;
  }

  Aws::SDKOptions options;
  Aws::InitAPI(options);
  {
    Aws::SNS::SNSClient sns;
    Aws::String protocol = argv[1];
    Aws::String topic_arn = argv[2];
    Aws::String endpoint = argv[3];

    Aws::SNS::Model::SubscribeRequest s_req;
    s_req.SetTopicArn(topic_arn);
    s_req.SetProtocol(protocol);
    s_req.SetEndpoint(endpoint);

    auto s_out = sns.Subscribe(s_req);

    if (s_out.IsSuccess())
    {
      std::cout << "Subscribed successfully " << std::endl;
    }
    else
    {
      std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
        << std::endl;
    }
  }

  Aws::ShutdownAPI(options);
  return 0;
}
```

- For API details, see Subscribe in *AWS SDK for C++ API Reference*.

Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

SDK for C++

> **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 endpoint to a topic - demonstrates how to initiate a
 subscription to an Amazon SNS topic with delivery
 *  to an email address.
 *
 * SNS will send a subscription confirmation email to the email address provided which
 you need to confirm to
```

```
 * receive messages.
 *
 * <protocol_value> set to "email" provides delivery of message via SMTP (see https://
docs.aws.amazon.com/sns/latest/api/API_Subscribe.html for available protocols).
 * <topic_arn_value> can be obtained from run_list_topics executable and includes the
 "arn:" prefix.
 */

int main(int argc, char ** argv)
{
  if (argc != 4)
  {
    std::cout << "Usage: subscribe_email <protocol_value=email> <topic_arn_value>"
                 " <email_address>" << std::endl;
    return 1;
  }

  Aws::SDKOptions options;
  Aws::InitAPI(options);
  {
    Aws::SNS::SNSClient sns;
    Aws::String protocol = argv[1];
    Aws::String topic_arn = argv[2];
    Aws::String endpoint = argv[3];

    Aws::SNS::Model::SubscribeRequest s_req;
    s_req.SetTopicArn(topic_arn);
    s_req.SetProtocol(protocol);
    s_req.SetEndpoint(endpoint);

    auto s_out = sns.Subscribe(s_req);

    if (s_out.IsSuccess())
    {
      std::cout << "Subscribed successfully " << std::endl;
    }
    else
    {
      std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
        << std::endl;
    }
  }

  Aws::ShutdownAPI(options);
  return 0;
}
```

- For API details, see Subscribe in *AWS SDK for C++ API Reference*.

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 that show you how to call individual AWS STS functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple AWS STS functions.

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

Assume a role

The following code example shows how to assume a role with AWS STS.

SDK for C++

> **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*.

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 that show you how to call individual Secrets Manager functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Secrets Manager functions.

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

Create a secret

The following code example shows how to create a Secrets Manager secret.

SDK for C++

> **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(int argc, const char *argv[])
{
    if (argc != 3) {
        std::cout << "Usage:\n" <<
        "    <secretName> <secretValue> \n\n" <<
        "Where:\n" <<
        "    secretName - The name of the secret (for example, tutorials/
MyFirstSecret). \n" <<
        "    secretValue - The secret value. " << std::endl;
        return 0;
    }

    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    InitAPI(options);
    {
        Aws::Client::ClientConfiguration config;

        //TODO(user): Enter the Region where you want to create the secret.
        String region = "us-east-1";
        if (!region.empty())
        {
                config.region = region;
        }
        SecretsManager::SecretsManagerClient sm_client(config);

        String secretName = argv[1];
        String secretString = argv[2];
        SecretsManager::Model::CreateSecretRequest request;
        request.SetName(secretName);
        request.SetSecretString(secretString);

        auto createSecretOutcome = sm_client.CreateSecret(request);
        if(createSecretOutcome.IsSuccess()){
                std::cout << "Create secret with name: " <<
 createSecretOutcome.GetResult().GetName() << std::endl;
        }else{
                std::cout << "Failed with Error: " << createSecretOutcome.GetError() <<
 std::endl;
```

```
        }
    }

    ShutdownAPI(options);
    return 0;
}
```

- For API details, see CreateSecret in *AWS SDK for C++ API Reference*.

Get a secret value

The following code example shows how to get a Secrets Manager secret value.

SDK for C++

> **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(int argc, const char *argv[])
{
    if (argc != 2) {
        std::cout << "Usage:\n" <<
                  "    <secretName> \n\n" <<
                  "Where:\n" <<
                  "    secretName - The name of the secret (for example, tutorials/
MyFirstSecret). \n"
                  << std::endl;
        return 0;
    }

    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    InitAPI(options);
    {
        Aws::Client::ClientConfiguration config;

        //TODO(user): Enter the Region where you want to create the secret.
        String region = "us-east-1";
        if (!region.empty())
        {
                config.region = region;
        }
        SecretsManager::SecretsManagerClient sm_client(config);

        String secretId = argv[1];
        SecretsManager::Model::GetSecretValueRequest request;
        request.SetSecretId(secretId);

        auto getSecretValueOutcome = sm_client.GetSecretValue(request);
        if(getSecretValueOutcome.IsSuccess()){
                std::cout << "Secret is: " <<
 getSecretValueOutcome.GetResult().GetSecretString() << std::endl;
        }else{
                std::cout << "Failed with Error: " << getSecretValueOutcome.GetError()
 << std::endl;
        }
    }

    ShutdownAPI(options);
```

```
    return 0;
}
```

- For API details, see GetSecretValue 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 that show you how to call individual Amazon Transcribe functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon Transcribe functions.

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

Produce real-time transcriptions

The following code example shows how to produce a real-time transcription with Amazon Transcribe.

SDK for C++

> **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;
    options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        Aws::Utils::Threading::Semaphore canCloseStream(0 /*initialCount*/, 1 /
*maxCount*/);

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS managed
 permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with AWS SDK version 1.9, this example only runs if
 the SDK is built
        // with the curl library.  (9/15/2022)
        // 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 transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([&canCloseStream](const TranscriptEvent &ev)
{
                bool isFinal = false;
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                        isFinal = true;
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
                if (isFinal) {
                    canCloseStream.Release();
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(8000);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [&canCloseStream](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();
            // Wait until the final transcript or an error is received.
            // Closing the stream prematurely will trigger an error.
            canCloseStream.WaitOne();
            stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /*maxCount*/);
        auto OnResponseCallback = [&signaling, &canCloseStream](
                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;
            }

            canCloseStream.Release();
            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.

Examples

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++

The shared responsibility model applies to data protection in this AWS product or service. 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. This content includes 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 user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal 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 sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with SDK for C ++ or other AWS services using the AWS Management Console, API, AWS CLI, or AWS SDKs. Any data that you enter into SDK for C++ or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Identity and Access Management for this AWS Product or Service

AWS Identity and Access Management (IAM) is an Amazon Web Services (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 resources in AWS services. IAM is an AWS service that you can use with no additional charge.

To use this AWS product or service to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your AWS account credentials.

For details about working with IAM, see AWS Identity and Access Management.

For an overview of IAM users and why they are important for the security of your account, see AWS Security Credentials in the Amazon Web Services General Reference.

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.

Compliance Validation for this AWS Product or Service

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.

The security and compliance of AWS services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS Services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using this AWS product or service to access an AWS service is determined by the sensitivity of your data, your organization's compliance objectives, and applicable

laws and regulations. If your use of an AWS service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides – Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – A whitepaper that describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – A collection of workbooks and guides that might apply to your industry and location.
- AWS Config – A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience for this AWS Product or Service

The Amazon Web Services (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 Availability 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 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.

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. In this case, TLS 1.0, TLS 1.1, and TLS 1.2 are acceptable secure protocols. Windows decides the actual protocol to use. 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 TLS 1.2 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.

Enforce TLS 1.2 on Windows

The following procedures show you how to enforce TLS 1.2 with WinHttp, WinINet, or IXMLHTTPRequest2.

Prerequisite: Enable TLS 1.1 and 1.2 on Windows

1. Determine whether your Windows version supports TLS 1.1 and TLS 1.2 natively, as described at https://docs.microsoft.com/en-us/windows/win32/secauthn/protocols-in-tls-ssl–schannel-ssp-.

2. Determine whether you need to patch your Windows version to enable TLS 1.1 and TLS 1.2 as the default, as described at https://support.microsoft.com/en-us/help/3140245/update-to-enable-tls-1-1-and-tls-1-2-as-default-secure-protocols-in-wi.
3. Proceed to one of the next procedures, as appropriate.

To enforce TLS 1.2 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.
2. Open `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` and find the following lines of code.

```
//disable insecure tls protocols, otherwise you might as well turn ssl verification off.
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1 | WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_1 |
 WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
if (!WinHttpSetOption(GetOpenHandle(), WINHTTP_OPTION_SECURE_PROTOCOLS, &flags,
 sizeof(flags)))
{
    AWS_LOGSTREAM_FATAL(GetLogTag(), "Failed setting secure crypto protocols with error
 code: " << GetLastError());
}
```

3. If necessary, change the value of the `flags` variable, as follows.

```
//disable insecure tls protocols, otherwise you might as well turn ssl verification off.
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
if (!WinHttpSetOption(GetOpenHandle(), WINHTTP_OPTION_SECURE_PROTOCOLS, &flags,
 sizeof(flags)))
{
    AWS_LOGSTREAM_FATAL(GetLogTag(), "Failed setting secure crypto protocols with error
 code: " << GetLastError());
}
```

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.
5. 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.

1. 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`, and `TLS 1.2`.
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 1.0\Client\Enabled            DWORD      0
TLS 1.1\Client\Enabled            DWORD      0
TLS 1.2\Client\Enabled            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.

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` 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 Getting the AWS SDK for C++ from source code (p. 4).

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 (p. 4).

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 Getting the AWS SDK for C++ from a package manager (p. 13).

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 `SetKMSDecryptWithAnyCMK(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.SetKMSDecryptWithAnyCMK(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
Updates to Getting Started (p. 238)	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` (p. 238)	Updated the structure of the `ClientConfiguration` to accurately reflect latest API.	September 22, 2022
Improvements to Getting Started (p. 238)	Improved clarity of getting started instructions for building the SDK on Windows and Linux.	June 24, 2022
Windows curl Support (p. 238)	Added notes about building the SDK for Windows with curl.	June 15, 2022
Retiring EC2-Classic (p. 238)	Added notes about retiring EC2-Classic.	April 13, 2022
Enabling SDK Metrics (p. 238)	Removed information about enabling SDK metrics, which has been deprecated.	January 20, 2022
Working with AWS services (p. 40)	Included lists of the code examples that are available on GitHub in the Code Examples repository.	January 11, 2022
Improvements (p. 238)	Improvements to Getting started section, Code examples section, and general standardization.	June 9, 2021
Version update (p. 238)	Reflected the update to general release version of SDK to 1.9.	April 20, 2021
Getting started using the AWS SDK for C++ (p. 2)	Updated section with new organization and details.	March 17, 2021
Amazon S3 Encryption Client Migration (p. 233)	Added information about how to migrate your applications from V1 to V2 of the Amazon S3 encryption client.	August 7, 2020
Security Content (p. 228)	Added security content.	February 6, 2020

Creating, listing, and deleting buckets (p. 108)	Updated the Amazon S3 `CreateBucket` example to support AWS Regions.	June 20, 2019
Build instructions (p. 238)	Updated the SDK build instructions.	April 16, 2019
Asynchronous methods (p. 44)	Added new section.	April 16, 2019
Service Client Classes (p. 33)	Various updates.	April 5, 2019
Managing Amazon S3 Access Permissions (p. 114)	Various updates.	April 3, 2019
Updates for building and for configuration variables (p. 238)	Updated the instructions for building the SDK. Updated the available AWS Client Configuration variables.	March 1, 2019
vcpkg C++ package manager (p. 238)	Updated the instructions for setting up the *vcpkg* C++ package manager.	January 19, 2019