

---

# AWS SDK for C++ Developer Guide

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

## Table of Contents

AWS SDK for C++ Developer Guide .....	1
Additional Documentation and Resources .....	1
Maintenance and support for SDK major versions .....	1
Getting Started .....	2
Setting Up the AWS SDK for C++ .....	2
Prerequisites .....	2
Building the SDK from Source .....	3
Getting the SDK Using Vcpkg with Visual C++ .....	5
Getting the SDK Using NuGet with Visual C++ .....	5
Providing AWS Credentials .....	6
Using the AWS SDK for C++ .....	6
Initializing and Shutting Down the SDK .....	6
Setting SDK Options .....	7
More Information .....	8
Building Your Application with CMake .....	8
Setting Up a CMake Project .....	8
Setting CMAKE_PREFIX_PATH (Optional) .....	8
Building with CMake .....	9
Configuring the SDK .....	10
CMake Parameters .....	10
General CMake Variables and Options .....	10
Android CMake Variables and Options .....	14
AWS Client Configuration .....	16
Configuration Variables .....	16
Overriding Your HTTP Client .....	18
Controlling IOStreams Used by the HttpClient and the AWSClient .....	18
SDK Metrics .....	19
Enable SDK Metrics for the AWS SDK for C++ .....	19
Update a CloudWatch Agent .....	20
Disable SDK Metrics .....	21
Definitions for SDK Metrics .....	21
Using the SDK .....	23
Service Client Classes .....	23
Utility Modules .....	23
HTTP Stack .....	23
String Utils .....	23
Hashing Utils .....	24
JSON Parser .....	24
XML Parser .....	24
Memory Management .....	24
Allocating and Deallocating Memory .....	24
STL and AWS Strings and Vectors .....	25
Remaining Issues .....	26
Native SDK Developers and Memory Controls .....	26
Logging .....	27
Error Handling .....	27
Code Examples .....	29
Amazon CloudWatch Examples .....	29
Getting Metrics from CloudWatch .....	30
Publishing Custom Metric Data .....	31
Working with CloudWatch Alarms .....	32
Using Alarm Actions in CloudWatch .....	35
Sending Events to CloudWatch .....	37
Amazon DynamoDB Examples .....	40

---

Working with Tables in DynamoDB .....	40
Working with Items in DynamoDB .....	45
Amazon EC2 Examples .....	48
Managing Amazon EC2 Instances .....	49
Using Elastic IP Addresses in Amazon EC2 .....	55
Using Regions and Availability Zones for Amazon EC2 .....	58
Working with Amazon EC2 Key Pairs .....	60
Working with Security Groups in Amazon EC2 .....	62
AWS Identity and Access Management (IAM) Examples .....	65
Managing IAM Access Keys .....	66
Managing IAM Users .....	70
Using IAM Account Aliases .....	73
Working with IAM Policies .....	76
Working with IAM Server Certificates .....	81
Amazon S3 Examples .....	85
Creating, Listing, and Deleting Buckets .....	85
Operations on Objects .....	89
Managing Amazon S3 Access Permissions .....	94
Managing Access to Amazon S3 Buckets Using Bucket Policies .....	100
Configuring an Amazon S3 Bucket as a Website .....	105
Amazon SQS Examples .....	109
Working with Amazon SQS Message Queues .....	109
Sending, Receiving, and Deleting Amazon SQS Messages .....	112
Enabling Long Polling for Amazon SQS Message Queues .....	114
Setting Visibility Timeout in Amazon SQS .....	117
Using Dead Letter Queues in Amazon SQS .....	118
Asynchronous Methods .....	119
Asynchronous SDK Methods .....	119
Calling SDK Asynchronous Methods .....	120
Notification of the Completion of an Asynchronous Operation .....	121
Security .....	123
Data Protection .....	123
Identity and Access Management .....	124
Compliance Validation .....	124
Resilience .....	125
Infrastructure Security .....	125
Enforcing TLS 1.2 .....	125
Enforce TLS 1.2 with libcurl on all platforms .....	126
Enforce TLS 1.2 on Windows .....	126
S3 Encryption Client Migration .....	128
Migration Overview .....	128
Update Existing Clients to Read New Formats .....	128
Migrate Encryption and Decryption Clients to V2 .....	129
Additional Examples .....	130
Document History .....	133

# 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.

## Additional Documentation and Resources

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

- [AWS SDK for C++ Reference](#)
- *Video: Introducing the AWS SDK for C++ from AWS re:invent 2015*
- [AWS C++ Developer Blog](#)
- GitHub:
  - [SDK source](#)
  - [SDK issues](#)
- The [AWS Code Sample Catalog](#)
- [SDK License](#)

## 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 Shared Configuration and Credentials Reference Guide](#):

- [AWS SDKs and Tools Maintenance Policy](#)
- [AWS SDKs and Tools Version Support Matrix](#)

# Getting Started Using the AWS SDK for C++

The topics in this section will help you set up and use the AWS SDK for C++.

## Topics

- [Setting Up the AWS SDK for C++ \(p. 2\)](#)
- [Providing AWS Credentials \(p. 6\)](#)
- [Using the AWS SDK for C++ \(p. 6\)](#)
- [Building Your Application with CMake \(p. 8\)](#)

## Setting Up the AWS SDK for C++

This section presents information about how to set up the AWS SDK for C++ on your development platform.

### Prerequisites

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

- Microsoft Visual Studio 2015 or later
- *or* GNU Compiler Collection (GCC) 4.9 or later
- *or* Clang 3.3 or later
- A minimum of 4 GB of RAM

#### Note

You need 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.

### Additional Requirements for Linux Systems

To compile on Linux, you must have the header files (*-dev* packages) for `libcurl`, `libopenssl`, `libuuid`, `zlib`, and, optionally, `libpulse` for Amazon Polly support. The packages are typically found by using the 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 *Redhat/Fedora-based systems*

```
sudo dnf install libcurl-devel openssl-devel libuuid-devel pulseaudio-devel
```

#### To install the packages on *CentOS-based systems*

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

## Building the SDK from Source

If you don't use Visual Studio (or don't want to or can't use vcpkg or NuGet as described later in this topic), you can build the SDK from source using command-line tools. This method also enables you to customize your SDK build—see [CMake Parameters \(p. 10\)](#) for the available options.

### To build the SDK from source

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

- Direct download: [aws/aws-sdk-cpp/archive/master.zip](#)

- Clone with Git:

HTTPS

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

SSH

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

2. Install [cmake \(v3.2 or later\)](#) and the relevant build tools for your platform. Ensure they are available in your `PATH`.
3. Recommended approach: Store the generated build files outside of the SDK source directory. Create a new directory to store them in. Then generate the build files by running `cmake`. Specify on the `cmake` command line whether to build a *Debug* or *Release* version.

```
sudo mkdir sdk_build  
cd sdk_build  
sudo cmake <path/to/sdk/source> -DCMAKE_BUILD_TYPE=[Debug | Release]
```

Alternatively, create the build files directly in the SDK source directory.

```
cd <path/to/sdk/source>  
sudo cmake . -DCMAKE_BUILD_TYPE=[Debug | Release]
```

Building the entire SDK can take a while. To build only a particular service, use the `cmake BUILD_ONLY` parameter. The example shown below builds only the Amazon S3 service. For more ways to modify the build output, see [CMake Parameters \(p. 10\)](#).

```
sudo cmake -DCMAKE_BUILD_TYPE=[Debug | Release] -D BUILD_ONLY="s3"
```

4. Build the SDK binaries by running one of the following operating system-dependent commands. If you're building the entire SDK, the operation can take one hour or longer.

auto make (Linux/macOS)

```
sudo make
```

Visual Studio (Windows)

```
msbuild ALL_BUILD.vcxproj
```

5. Install the SDK by running one of the following operating system-dependent commands. Administrator privileges required.

auto make (Linux/macOS)

```
sudo make install
```

Visual Studio (Windows)

```
rem Run this command in a command shell running in ADMIN mode  
rem The SDK is installed in `Program Files (x86)\aws-cpp-sdk-all\  
msbuild INSTALL.vcxproj /p:Configuration=[Debug | Release | "Debug;Release"]
```

## Building for Android

To build for Android, add `-DTARGET_ARCH=ANDROID` to your `cmake` command line. The AWS SDK for C++ includes a `cmake` toolchain file that should cover what's needed, assuming you've set the appropriate environment variables (`ANDROID_NDK`).

### Android on Windows

Building for Android on Windows requires additional setup. In particular, you have to run `cmake` from a Visual Studio (2013 or later) developer command prompt. You'll also need 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 targets in serially. To make things go more quickly, we recommend installing `JOM` as an alternative to `nmake`, and then changing the `cmake` invocation as follows:

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

## Creating Release Builds

auto make

```
cmake -DCMAKE_BUILD_TYPE=Release <path/to/sdk/source>  
make  
sudo make install
```

Visual Studio

```
cmake <path-to-root-of-this-source-code> -G "Visual Studio 12 Win64"  
msbuild INSTALL.vcxproj /p:Configuration=Release
```

## Running Integration Tests

Several directory names include the suffix `*integration-tests`. After the project is built, the tests stored in these directories can be run to verify the project's correct execution.



## Getting the SDK Using Vcpkg with Visual C++

You can use vcpkg to manage dependencies for AWS SDK for C++ projects that you develop with Microsoft Visual C++. To use this procedure, you must have [vcpkg](#) installed on your system.

### To use the SDK with vcpkg

1. Open a Windows command prompt and navigate to the vcpkg directory.
2. Integrate vcpkg into Visual Studio. You can [integrate](#) per project or per user. The command line shown below integrates vcpkg for the current user.

```
vcpkg integrate install
```

3. Install the AWS SDK for C++ package. The package compiles the entire SDK and its dependencies. It can take a while.

```
vcpkg install aws-sdk-cpp[*]:x86-windows --recurse
```

To reduce build time, build only the SDK packages needed. Specify the package names in square brackets. Also include the SDK *core* package.

```
vcpkg install aws-sdk-cpp[core,s3,ec2]:x86-windows
```

A package name can be derived from the AWS SDK for C++ repo 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
```

4. Open your project in Visual Studio.
5. #include the AWS SDK for C++ header files you want in your source code.

When you build your project, the correct binaries are automatically included for each runtime/architecture configuration you use.

## Getting the SDK Using NuGet with Visual C++

You can use NuGet to manage dependencies for AWS SDK for C++ projects that you develop with Microsoft Visual C++. To use this procedure, you must have [NuGet](#) installed on your system.

### Note

The following information applies only to AWS SDK for C++ versions 1.6 and earlier and to Microsoft Visual Studio 2015 and 2017. For later versions of the AWS SDK for C++ and Visual Studio, choose from the other setup options earlier in this topic.

### To use the SDK with NuGet

1. Open your project in Visual Studio.
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`. Or, because AWS SDK for C++ libraries are named consistently, use `AWSSDKCPP-service name` to add a library for a particular service to your project.
4. Choose **Install** to install the libraries and add them to your project.

When you build your project, the correct binaries are automatically included for each runtime/ architecture configuration you use — you won't need to manage these dependencies yourself.

## 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.

For related information, see the following:

- To create a new AWS account, see [How do I create and activate a new AWS account](#) on the AWS Premium Support website.
- To create long-term AWS credentials, see [Programmatic access](#) in the AWS General Reference.
- To create short-term AWS credentials, see [Temporary Security Credentials](#) in the AWS Identity and Access Management User Guide.
- To learn more about supported provider chains, see the [AWS Tools and SDKs Shared Configuration and Credentials Reference Guide](#), specifically:
  - [The `.aws/credentials` and `.aws/config` files](#)
  - [Using environment variables](#)
  - [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 ECS Developer Guide.

## Using the AWS SDK for C++

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);
```

- Install a custom memory manager

```
MyMemoryManager memoryManager;
Aws::SDKOptions options;
options.memoryManagementOptions.memoryManager = &memoryManager;
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 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.

## More Information

Examples of AWS SDK for C++ application code are described in the section [AWS SDK for C++ Code Examples \(p. 29\)](#). Each example includes a link to the full source code on GitHub which can be used as a starting point for your own applications.

## Building Your Application with CMake

**CMake** is a build tool that can manage your application's dependencies and create native makefiles suitable for the platform you're building on. It's an easy way to create and build projects using the AWS SDK for C++.

### Setting Up a CMake Project

**To set up a CMake project for use with the AWS SDK for C++**

1. Create a directory to hold your source files.

```
mkdir my_example_project
```

2. Open the directory and add a `CMakeLists.txt` file that specifies your project's name, executables, source files, and linked libraries. The following is a minimal example.

```
# Minimal CMakeLists.txt for the AWS SDK for C++.
cmake_minimum_required(VERSION 3.8)

# Use shared libraries, which is the default for the AWS C++ SDK build.
option(BUILD_SHARED_LIBS "Build shared libraries" ON)

# "my-example" is just an example value.
project(my-example)

# Locate the AWS SDK for C++ package.
# "s3" and "cloudtrail" are just example values.
find_package(AWSSDK REQUIRED COMPONENTS s3 cloudtrail)

# The executable name and its sourcefiles.
add_executable(my-example my-example.cpp)

# Build using the C++ standard version 11.
target_compile_features(my-example PUBLIC cxx_std_11)

# The libraries used by your executable.
target_link_libraries(my-example ${AWSSDK_LINK_LIBRARIES})
```

#### Note

You can set many options in your `CMakeLists.txt` build configuration file. For an introduction to the file's features, see the [CMake tutorial](#) on the CMake website.

### Setting CMAKE\_PREFIX\_PATH (Optional)

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

When you install the SDK into a non-default location, CMake needs to know where to find the file `AWSSDKConfig.cmake` so that it can properly resolve the AWS SDK libraries that your application uses. You can find this file in the build directory that you used to [build the SDK \(p. 2\)](#).

By adding the path to a list in `CMAKE_PREFIX_PATH`, you won't need to type this path every time you rebuild your application.

You can set it on Linux, macOS, or Unix like this.

```
export CMAKE_PREFIX_PATH=$CMAKE_PREFIX_PATH:/path/to/sdk_install_dir
```

On Windows, use `set` instead.

```
set CMAKE_PREFIX_PATH=%CMAKE_PREFIX_PATH%;C:\path\to\sdk_install_dir
```

## Building with CMake

Create a directory into which `cmake` will build your application.

```
mkdir my_project_build
```

Open the directory and run `cmake` using the path to your project's source directory.

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

If you didn't set `CMAKE_PREFIX_PATH`, you must add the path to the SDK's build directory using `-DAWSSDK_DIR`.

```
cmake -DAWSSDK_DIR=/path/to/sdk_install_dir ../my_example_project
```

After `cmake` generates your build directory, you can use `make` (or `nmake` on Windows) to build your application.

```
make
```

# Configuring the AWS SDK for C++

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

## Topics

- [CMake Parameters](#) (p. 10)
- [AWS Client Configuration](#) (p. 16)
- [Overriding Your HTTP Client](#) (p. 18)
- [Controlling IOStreams Used by the HttpClient and the AWSClient](#) (p. 18)
- [SDK Metrics](#) (p. 19)

## 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](#) (p. 11)
- [BUILD\\_ONLY](#) (p. 11)
- [BUILD\\_SHARED\\_LIBS](#) (p. 11)
- [CPP\\_STANDARD](#) (p. 11)
- [CUSTOM\\_MEMORY\\_MANAGEMENT](#) (p. 12)
- [ENABLE\\_RTTI](#) (p. 12)
- [ENABLE\\_TESTING](#) (p. 12)
- [ENABLE\\_UNITY\\_BUILD](#) (p. 12)
- [FORCE\\_SHARED\\_CRT](#) (p. 12)
- [G](#) (p. 13)
- [MINIMIZE\\_SIZE](#) (p. 13)
- [NO\\_ENCRYPTION](#) (p. 13)
- [NO\\_HTTP\\_CLIENT](#) (p. 13)
- [REGENERATE\\_CLIENTS](#) (p. 14)

- [SIMPLE\\_INSTALL](#) (p. 14)
- [TARGET\\_ARCH](#) (p. 14)

## 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;version=2015-12-21"
```

## 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`

## 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\_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\_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. 12). 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. Turn this *ON* to inject your own 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. 14).

Values

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

# Android CMake Variables and Options

Use the following variables when you are creating an Android build of the SDK (when *TARGET\_ARCH* (p. 14) is set to *ANDROID*).

**Topics**

- [ANDROID\\_ABI](#) (p. 14)
- [ANDROID\\_NATIVE\\_API\\_LEVEL](#) (p. 15)
- [ANDROID\\_STL](#) (p. 15)
- [ANDROID\\_TOOLCHAIN\\_NAME](#) (p. 15)
- [DISABLE\\_ANDROID\\_STANDALONE\\_BUILD](#) (p. 15)
- [NDK\\_DIR](#) (p. 16)

## 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 \(p. 15\)](#) 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.

# AWS Client Configuration

Use the client configuration to control various behaviors of the AWS SDK for C++.

ClientConfiguration declaration:

```
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;
    bool 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 whether the HTTP stack follows 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`.

## Overriding Your HTTP Client

The default HTTP client for Windows is `WinHTTP`. The default HTTP client for all other platforms is `curl`. If needed, you can create a custom `HttpClientFactory` to pass to any service client's constructor.

## 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 `AmazonS3GetObject` 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);
```

## SDK Metrics

AWS SDK Metrics for Enterprise Support (SDK Metrics) enables Enterprise customers to collect metrics from AWS SDKs on their hosts and clients shared with AWS Enterprise Support. SDK Metrics provides information that helps speed up detection and diagnosis of issues occurring in connections to AWS services for AWS Enterprise Support customers.

As telemetry is collected on each host, it is relayed via UDP to 127.0.0.1 (AKA localhost), where the CloudWatch Agent aggregates the data and sends it to the SDK Metrics service. Therefore, to receive metrics, the CloudWatch Agent is required to be added to your instance.

The following steps to set up SDK Metrics pertain to an Amazon EC2 instance running Amazon Linux for a client application that is using the AWS SDK for C++. SDK Metrics is also available for your production environments if you enable it while configuring the AWS SDK for C++.

To utilize SDK Metrics, run the latest version of the CloudWatch agent. Learn how to [Configure the CloudWatch Agent for SDK Metrics](#) in the Amazon CloudWatch User Guide.

To set up SDK Metrics with the AWS SDK for C++, follow these instructions:

1. Install the latest version of the AWS SDK for C++.
2. Host your project on an Amazon EC2 instance or in your local environment.
3. Create an application with an AWS SDK for C++ client to use an AWS service.
4. Install and configure a CloudWatch agent on an EC2 instance or in your local environment.
5. Authorize SDK Metrics to Collect and Send Metrics
6. [Enable SDK Metrics for the AWS SDK for C++ \(p. 19\)](#)

For more information, see the following:

- [Update a CloudWatch Agent \(p. 20\)](#)
- [Disable SDK Metrics \(p. 21\)](#)

## Enable SDK Metrics for the AWS SDK for C++

By default, SDK Metrics uses port 31000 and is disabled.

```
//default values
[
    'enabled' => false,
    'port' => 31000,
]
```

Enabling SDK Metrics is independent of configuring your credentials to use an AWS service.

You can enable SDK Metrics by setting environment variables or by using the AWS Shared config file.

## Option 1: Set Environment Variables

If `AWS_CSM_ENABLED` is not set, the SDK checks the profile specified in the environment variable under `AWS_PROFILE` to determine if SDK Metrics is enabled. By default this is set to `false`.

To turn on SDK Metrics, add the following to your environmental variables.

```
export AWS_CSM_ENABLED=true
```

[Other configuration settings \(p. 20\)](#) are available.

Note: Enabling SDK Metrics does not configure your credentials to use an AWS service.

## Option 2: AWS Shared Config File

If no CSM configuration is found in the environment variables, the SDK looks for your default AWS profile field. If `AWS_DEFAULT_PROFILE` is set to something other than default, update that profile. To enable SDK Metrics, add `csm_enabled` to the shared config file located at `~/.aws/config`.

```
[default]
csm_enabled = true

[profile aws_csm]
csm_enabled = true
```

[Other configuration settings \(p. 20\)](#) are available.

Note: Enabling SDK Metrics is independent from configuring your credentials to use an AWS service. You can use a different profile to authenticate.

## Update a CloudWatch Agent

To make changes to the port, you need to set the values and then restart any AWS jobs that are currently active.

### Option 1: Set Environment Variables

Most services use the default port. But if your service requires a unique port ID, add `AWS_CSM_PORT=[port_number]`, to the host's environment variables.

```
export AWS_CSM_ENABLED=true
export AWS_CSM_PORT=1234
```

### Option 2: AWS Shared Config File

Most services use the default port. But if your service requires a unique port ID, add `csm_port = [port_number]` to `~/.aws/config`.

```
[default]
csm_enabled = false
csm_port = 1234

[profile aws_csm]
csm_enabled = false
csm_port = 1234
```



## Restart SDK Metrics

To restart a job, run the following commands.

```
amazon-cloudwatch-agent-ctl -a stop;  
amazon-cloudwatch-agent-ctl -a start;
```

## Disable SDK Metrics

To turn off SDK Metrics, set `esm_enabled` to `false` in your environment variables, or in your AWS Shared config file located at `~/.aws/config`. Then restart your CloudWatch agent so that the changes can take effect.

### Environment Variables

```
export AWS_CSM_ENABLED=false
```

### AWS Shared Config File

Remove `esm_enabled` from the profiles in your AWS Shared config file located at `~/.aws/config`.

#### Note

Environment variables override the AWS Shared config file. If SDK Metrics is enabled in the environment variables, the SDK Metrics remains enabled.

```
[default]  
esm_enabled = false  
  
[profile aws_csm]  
esm_enabled = false
```

To disable SDK Metrics, use the following command to stop CloudWatch Agent.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a stop &&  
echo "Done"
```

If you are using other CloudWatch features, restart CloudWatch Agent with the following command.

```
amazon-cloudwatch-agent-ctl -a start;
```

## Restart SDK Metrics

To restart a SDK Metrics job, run the following commands.

```
amazon-cloudwatch-agent-ctl -a stop;  
amazon-cloudwatch-agent-ctl -a start;
```

## Definitions for SDK Metrics

You can use the following descriptions of SDK Metrics to interpret your results. In general, these metrics are available for review with your Technical Account Manager during regular business reviews. AWS Support resources and your Technical Account Manager should have access to SDK Metrics data to help you resolve cases, but if you discover data that is confusing or unexpected, but doesn't seem to be

negatively impacting your applications' performance, it is best to review that data during scheduled business reviews.

### Metrics

Metric	Definition	How to use it
CallCount	Total number of successful or failed API calls from your code to AWS services	Use it as a baseline to correlate with other metrics like errors or throttling.
ClientErrorCount	Number of API calls that fail with client errors (4xx HTTP response codes). Examples: Throttling, Access denied, S3 bucket does not exist, and Invalid parameter value.	Except in certain cases related to throttling (ex. when throttling occurs due to a limit that needs to be increased) this metric can indicate something in your application that needs to be fixed.
ConnectionErrorCount	Number of API calls that fail because of errors connecting to the service. These can be caused by network issues between the customer application and AWS services including load balancers, DNS failures, transit providers. In some cases, AWS issues may result in this error.	Use this metric to determine whether issues are specific to your application or are caused by your infrastructure and/or network. High ConnectionErrorCount could also indicate short timeout values for API calls.
EndToEndLatency	Total time for your application to make a call using the AWS SDK, inclusive of retries. In other words, regardless of whether it is successful after several attempts, or as soon as a call fails due to an unretrievable error.	Determine how AWS API calls contribute to your application's overall latency. Higher than expected latency may be caused by issues with network, firewall, or other configuration settings, or by latency that occurs as a result of SDK retries.
ServerErrorCount	Number of API calls that fail due to server errors (5xx HTTP response codes) from AWS Services. These are typically caused by AWS services.	Determine cause of SDK retries or latency. This metric will not always indicate that AWS services are at fault, as some AWS teams classify latency as an HTTP 503 response.
ThrottleCount	Number of API calls that fail due to throttling by AWS services.	Use this metric to assess if your application has reached throttle limits, as well as to determine the cause of retries and application latency. Consider distributing calls over a window instead of batching your calls.

# 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 \(p. 29\)](#).

## Topics

- [Service Client Classes \(p. 23\)](#)
- [Utility Modules \(p. 23\)](#)
- [Memory Management \(p. 24\)](#)
- [Logging \(p. 27\)](#)
- [Error Handling \(p. 27\)](#)

## Service Client Classes

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

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

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

When instantiating a client class, AWS credentials must be supplied. For more information about credentials, see [Providing AWS Credentials \(p. 6\)](#).

## 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 \(p. 16\)](#).

Headers	<a href="#">/aws/core/http/</a>
API Documentation	<a href="#">Aws::Http</a>

### String Utils

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

Header	<a href="#">aws/core/utis/StringUtils.h</a>
API Documentation	<a href="#">Aws::Utils::StringUtils</a>

## Hashing Utils

Hashing functions such as SHA256, MD5, Base64, and SHA256\_HMAC.

Header	<a href="#">/aws/core/utis/HashingUtils.h</a>
API Documentation	<a href="#">Aws::Utils::HashingUtils</a>

## JSON Parser

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

Header	<a href="#">/aws/core/utis/json/JsonSerializer.h</a>
API Documentation	<a href="#">Aws::Utils::Json::JsonValue</a>

## XML Parser

A lightweight XML parser (a thin wrapper around *tinyxml2*). The [RAII pattern](#) has been added to the interface.

Header	<a href="#">/aws/core/utis/xml/XMLSerializer.h</a>
API Documentation	<a href="#">Aws::Utils::Xml</a>

# 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 \(p. 25\)](#).

## Allocating and Deallocating Memory

### To allocate or deallocate memory

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

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

### Note

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

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

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

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

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

## STL and AWS Strings and Vectors

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

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

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

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

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

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

```
#ifndef AWS_CUSTOM_MEMORY_MANAGEMENT
template< typename T >
```

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

#else

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

#endif
```

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

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

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

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

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

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

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

## Remaining Issues

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

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

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

## Native SDK Developers and Memory Controls

Follow these rules in the SDK code:

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

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

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

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

## Logging

The AWS SDK for C++ includes logging support that you can configure. When initializing the logging system, you can control the filter level and the logging target (file with a name that has a configurable prefix or a stream). The log file generated by the prefix option rolls over once per hour to allow for archiving or deleting log files.

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

If you don't call `InitializeAWSLogging` in your program, the SDK will not do any logging. If you do use logging, don't forget to shut it down 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;
}
```



# AWS SDK for C++ Code Examples

This section provides examples, guidance, and tips you can use to work with specific AWS services using the AWS SDK for C++.

Find the source code for these examples and others in the AWS documentation [code examples repository on GitHub](#).

To propose a new code example for the AWS documentation team to consider producing, create a new request. The team is looking to produce code examples that cover broader scenarios and use cases, versus simple code snippets that cover only individual API calls. For instructions, see the “Proposing new code examples” section in the [Readme on GitHub](#).

## Topics

- [Amazon CloudWatch Examples Using the AWS SDK for C++ \(p. 29\)](#)
- [Amazon DynamoDB Examples Using the AWS SDK for C++ \(p. 40\)](#)
- [Amazon EC2 Examples Using the AWS SDK for C++ \(p. 48\)](#)
- [IAM Code Examples Using the AWS SDK for C++ \(p. 65\)](#)
- [Amazon S3 Code Examples Using the AWS SDK for C++ \(p. 85\)](#)
- [Amazon SQS Code Examples Using the AWS SDK for C++ \(p. 109\)](#)
- [Asynchronous Methods \(p. 119\)](#)

## 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 Amazon Web Services (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 each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

## Topics

- [Getting Metrics from CloudWatch \(p. 30\)](#)
- [Publishing Custom Metric Data \(p. 31\)](#)
- [Working with CloudWatch Alarms \(p. 32\)](#)
- [Using Alarm Actions in CloudWatch \(p. 35\)](#)

- [Sending Events to CloudWatch \(p. 37\)](#)

## Getting Metrics from CloudWatch

### 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 \(p. 6\)](#).

## 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/").

### 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 \(p. 6\)](#).

## 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

### 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 \(p. 6\)](#).

## 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. 32).

### 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. 6).

## 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 \(p. 6\)](#).

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

```
#include <iostream>
```

### Code

```
Aws::CloudWatchEvents::CloudWatchEventsClient 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/CloudWatchEventsClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>
```

### Code

```
Aws::CloudWatchEvents::CloudWatchEventsClient 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/CloudWatchEventsClient.h>  
#include <aws/events/model/PutTargetsRequest.h>  
#include <aws/events/model/PutTargetsResult.h>  
#include <aws/core/Utils/Outcome.h>  
#include <iostream>
```

### Code

```
Aws::CloudWatchEvents::CloudWatchEventsClient 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 [DynamoDB](#) using the AWS SDK for C++.

## Note

Only the code that is necessary to demonstrate each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

## Topics

- [Working with Tables in DynamoDB \(p. 40\)](#)
- [Working with Items in DynamoDB \(p. 45\)](#)

## 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 account and 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 ltr;
ltr.SetLimit(50);
do
{
    const Aws::DynamoDB::Model::ListTablesOutcome& lto = dynamoClient.ListTables(ltr);
    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;
    ltr.SetExclusiveStartTableName(lto.GetResult().GetLastEvaluatedTableName());
} while (!ltr.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 << "Table \"" << result.GetResult().GetTableDescription().GetTableName() <<
        " 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(name);
req.AddKey("Name", hashKey);
if (!projection.empty())
    req.SetProjectionExpression(projection);

// 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 << i.first << ": " << i.second.GetS() << std::endl;
    }
    else
    {
        std::cout << "No item found with the key " << name << 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 pir;
pir.SetTableName(table);

Aws::DynamoDB::Model::AttributeValue av;
av.SetS(name);
pir.AddItem("Name", av);

for (int x = 3; x < argc; x++)
{
    const Aws::String arg(argv[x]);
    const Aws::Vector<Aws::String>& flds = Aws::Utils::StringUtils::Split(arg, '=');
    if (flds.size() == 2)
    {
        Aws::DynamoDB::Model::AttributeValue val;
        val.SetS(flds[1]);
        pir.AddItem(flds[0], val);
    }
    else
    {
        std::cout << "Invalid argument: " << arg << std::endl << USAGE;
        return 1;
    }
}

const Aws::DynamoDB::Model::PutItemOutcome result = dynamoClient.PutItem(pir);
if (!result.IsSuccess())
{
    std::cout << result.GetError().GetMessage() << std::endl;
    return 1;
}
std::cout << "Done!" << 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("Name", 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, '=');
// parsed[0] == attribute name, parsed[1] == attribute value
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 each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

## Topics

- [Managing Amazon EC2 Instances \(p. 49\)](#)
- [Using Elastic IP Addresses in Amazon EC2 \(p. 55\)](#)
- [Using Regions and Availability Zones for Amazon EC2 \(p. 58\)](#)
- [Working with Amazon EC2 Key Pairs \(p. 60\)](#)
- [Working with Security Groups in Amazon EC2 \(p. 62\)](#)

# Managing Amazon EC2 Instances

## 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 \(p. 6\)](#).

## 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_instances_outcome = ec2.StartInstances(start_request);

if (!start_instances_outcome.IsSuccess())
{
    std::cout << "Failed to start instance " << instance_id << ": " <<
              << start_instances_outcome.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 account and 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

### 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. 6).

## 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).

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 *EC2-Classic* or 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

### 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. 6).

## Describe Regions

To list the regions available to your 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

### 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 \(p. 6\)](#).

## 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

### 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 \(p. 6\)](#).

## 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 each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

### Topics

- [Managing IAM Access Keys \(p. 66\)](#)
- [Managing IAM Users \(p. 70\)](#)
- [Using IAM Account Aliases \(p. 73\)](#)
- [Working with IAM Policies \(p. 76\)](#)
- [Working with IAM Server Certificates \(p. 81\)](#)

## Managing IAM Access Keys

### 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 \(p. 6\)](#).

## 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/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/CreateAccessKeyRequest.h>
#include <aws/iam/model/CreateAccessKeyResult.h>
#include <iostream>
```

### Code:

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::CreateAccessKeyRequest request;
request.SetUserName(user_name);

auto outcome = iam.CreateAccessKey(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error creating access key for IAM user " << user_name
              << " " << outcome.GetError().GetMessage() << std::endl;
}
else
{
    const auto &accessKey = outcome.GetResult().GetAccessKey();
    std::cout << "Successfully created access key for IAM user " <<
              user_name << std::endl << "  aws_access_key_id = " <<
              accessKey.GetAccessKeyId() << std::endl <<
              "  aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
              std::endl;
}
```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;
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::cout << "Failed to list access keys for user " << userName
            << ": " << outcome.GetError().GetMessage() << std::endl;
        break;
    }

    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();
    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) << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated())
    {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else
    {
        done = 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>
```

### Code:

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

request.SetAccessKeyId(key_id);

auto outcome = iam.GetAccessKeyLastUsed(request);

if (!outcome.IsSuccess())
{
    std::cout << "Error querying last used time for access key " <<
        key_id << ":" << outcome.GetError().GetMessage() << std::endl;
}
else
{
    auto lastUsedTimeString =
        outcome.GetResult()
            .GetAccessKeyLastUsed()
            .GetLastUsedDate()
            .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
    std::cout << "Access key " << key_id << " last used at time " <<
        lastUsedTimeString << std::endl;
}
```

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>
```



**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::UpdateAccessKeyRequest request;
request.SetUserName(user_name);
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 " << user_name << std::endl;
}
else
{
    std::cout << "Error updated status of access key " << accessKeyId <<
        " for user " << user_name << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

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 \(p. 68\)](#) function instead.

**Includes:**

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

**Code:**

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::DeleteAccessKeyRequest request;
request.SetUserName(user_name);
request.SetAccessKeyId(key_id);

auto outcome = iam.DeleteAccessKey(request);

if (!outcome.IsSuccess())
{
    std::cout << "Error deleting access key " << key_id << " from user "
        << user_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully deleted access key " << key_id
        << " for IAM user " << user_name << std::endl;
}
```

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

### 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 \(p. 6\)](#).

### 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/CreateUserResult.h>
```

#### Code:

```
Aws::IAM::IAMClient iam;
```

```
Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(user_name);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess())
{
    std::cout << "Error creating IAM user " << user_name << " " <<
        create_outcome.GetError().GetMessage() << std::endl;
    return;
}
std::cout << "Successfully created IAM user " << user_name << std::endl;
```

### 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;
```

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

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

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 <aws/iam/model/ListUsersResult.h>
#include <iomanip>
#include <iostream>
```

**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::ListUsersRequest request;

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = iam.ListUsers(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list iam users:" <<
            outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
        break;
    }

    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) << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated())
    {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else
    {
        done = 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>
```

### Code:

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::UpdateUserRequest request;
request.SetUserName(old_name);
request.SetNewUserName(new_name);

auto outcome = iam.UpdateUser(request);
if (outcome.IsSuccess())
{
    std::cout << "IAM user " << old_name <<
        " successfully updated with new user name " << new_name <<
        std::endl;
}
else
{
    std::cout << "Error updating user name for IAM user " << old_name <<
```

```
    ":" << outcome.GetError().GetMessage() << std::endl;  
}
```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;
```

```
Aws::IAM::Model::DeleteUserRequest request;  
request.SetUserName(user_name);  
auto outcome = iam.DeleteUser(request);  
if (!outcome.IsSuccess())  
{  
    std::cout << "Error deleting IAM user " << user_name << ": " <<  
        outcome.GetError().GetMessage() << std::endl;  
    return;  
}  
std::cout << "Successfully deleted IAM user " << user_name << std::endl;
```

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.

### 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 \(p. 6\)](#).

## 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>
```

**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::CreateAccountAliasRequest request;
request.SetAccountAlias(alias_name);

auto outcome = iam.CreateAccountAlias(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error creating account alias " << alias_name << ": "
                << outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created account alias " << alias_name <<
                std::endl;
}
```

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>
```

**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::ListAccountAliasesRequest request;

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = iam.ListAccountAliases(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list account aliases: " <<
                    outcome.GetError().GetMessage() << std::endl;
        break;
    }

    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;
}
}
```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::DeleteAccountAliasRequest request;
request.SetAccountAlias(alias_name);

const auto outcome = iam.DeleteAccountAlias(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error deleting account alias " << alias_name << ": "
              << outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted account alias " << alias_name <<
              << std::endl;
}
```

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

### 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 \(p. 6\)](#).

## 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>
```

### Code:

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::CreatePolicyRequest request;
request.SetPolicyName(policy_name);
request.SetPolicyDocument(BuildSamplePolicyDocument(rsrc_arn));

auto outcome = iam.CreatePolicy(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error creating policy " << policy_name << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created policy " << policy_name <<
        std::endl;
}
```

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.

```
static const char* const POLICY_TEMPLATE =
"{
  \"Version\": \"2012-10-17\",
  \"Statement\": [
    {
      \"Effect\": \"Allow\",
      \"Action\": \"logs:CreateLogGroup\",
      \"Resource\": \"%s\"
    },
    {
      \"Effect\": \"Allow\",
```



```

"        \"Action\": [\"
"            \"dynamodb:DeleteItem\", \"
"            \"dynamodb:GetItem\", \"
"            \"dynamodb:PutItem\", \"
"            \"dynamodb:Scan\", \"
"            \"dynamodb:UpdateItem\"
"        ], \"
"        \"Resource\": \"%s\"
"    }\"
" ]\"
"}\";

Aws::String BuildSamplePolicyDocument(const Aws::String& rsrc_arn)
{
    char policyBuffer[512];
#ifdef WIN32
    sprintf_s(policyBuffer, POLICY_TEMPLATE, rsrc_arn.c_str(), rsrc_arn.c_str());
#else
    sprintf(policyBuffer, POLICY_TEMPLATE, rsrc_arn.c_str(), rsrc_arn.c_str());
#endif // WIN32
    return Aws::String(policyBuffer);
}

```

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>

```

### Code:

```

Aws::IAM::IAMClient iam;
Aws::IAM::Model::GetPolicyRequest request;
request.SetPolicyArn(policy_arn);

auto outcome = iam.GetPolicy(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error getting policy " << policy_arn << ": " <<
        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;
}

```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::DeletePolicyRequest request;
request.SetPolicyArn(policy_arn);

auto outcome = iam.DeletePolicy(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error deleting policy with arn " << policy_arn << ": "
              << outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted policy with arn " << policy_arn
              << std::endl;
}
```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;

Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
list_request.SetRoleName(role_name);

bool done = false;
while (!done)
{
    auto list_outcome = iam.ListAttachedRolePolicies(list_request);
```

```

if (!list_outcome.IsSuccess())
{
    std::cout << "Failed to list attached policies of role " <<
        role_name << ": " << list_outcome.GetError().GetMessage() <<
        std::endl;
    return;
}

const auto& policies = list_outcome.GetResult().GetAttachedPolicies();
if (std::any_of(policies.cbegin(), policies.cend(),
    [=](const Aws::IAM::Model::AttachedPolicy& policy)
    {
        return policy.GetPolicyArn() == policy_arn;
    })))
{
    std::cout << "Policy " << policy_arn <<
        " is already attached to role " << role_name << std::endl;
    return;
}

done = !list_outcome.GetResult().GetIsTruncated();
list_request.SetMarker(list_outcome.GetResult().GetMarker());
}

Aws::IAM::Model::AttachRolePolicyRequest request;
request.SetRoleName(role_name);
request.SetPolicyArn(policy_arn);

auto outcome = iam.AttachRolePolicy(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to attach policy " << policy_arn << " to role " <<
        role_name << ": " << outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully attached policy " << policy_arn << " to role " <<
    role_name << std::endl;

```

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>

```

### Code:

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::ListPoliciesRequest request;

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

    if (!header)
    {
        std::cout << std::left << std::setw(55) << "Name" <<
            std::setw(30) << "ID" << std::setw(80) << "Arn" <<
            std::setw(64) << "Description" << std::setw(12) <<
            "CreateDate" << std::endl;
        header = true;
    }

    const auto &policies = outcome.GetResult().GetPolicies();
    for (const auto &policy : policies)
    {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
            std::setw(64) << policy.GetDescription() << std::setw(12) <<
            policy.GetCreateDate().ToGmtString(DATE_FORMAT) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated())
    {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else
    {
        done = 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>
```

### Code:

```
Aws::IAM::IAMClient iam;
```

```
Aws::IAM::Model::DetachRolePolicyRequest detach_request;  
detach_request.SetRoleName(role_name);  
detach_request.SetPolicyArn(policy_arn);  
  
auto detach_outcome = iam.DetachRolePolicy(detach_request);  
if (!detach_outcome.IsSuccess())  
{  
    std::cout << "Failed to detach policy " << policy_arn << " from role "  
                << role_name << ": " << detach_outcome.GetError().GetMessage() <<  
                std::endl;  
    return;  
}  
  
std::cout << "Successfully detached policy " << policy_arn << " from role "  
            << role_name << std::endl;
```

See the [complete example](#).

## More Information

- [Overview of IAM Policies](#) in the IAM User Guide.
- [AWS 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](#).

### 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 \(p. 6\)](#).

## 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>
```

**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::GetServerCertificateRequest request;
request.SetServerCertificateName(cert_name);

auto outcome = iam.GetServerCertificate(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error getting server certificate " << cert_name <<
        " : " << outcome.GetError().GetMessage() << 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;
}
```

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 <aws/iam/model/ListServerCertificatesResult.h>
#include <iomanip>
#include <iostream>
```

**Code:**

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::ListServerCertificatesRequest request;

bool done = false;
bool header = false;
```

```

while (!done)
{
    auto outcome = iam.ListServerCertificates(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list server certificates: " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    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) <<
            std::setw(14) <<
            certificate.GetExpiration().ToGmtString(DATE_FORMAT) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated())
    {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else
    {
        done = 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>

```

### Code:

```

Aws::IAM::IAMClient iam;

```

```
Aws::IAM::Model::UpdateServerCertificateRequest request;
request.SetServerCertificateName(old_name);
request.SetNewServerCertificateName(new_name);

auto outcome = iam.UpdateServerCertificate(request);
if (outcome.IsSuccess())
{
    std::cout << "Server certificate " << old_name
                << " successfully renamed as " << new_name
                << std::endl;
}
else
{
    std::cout << "Error changing name of server certificate " <<
                old_name << " to " << new_name << ":" <<
                outcome.GetError().GetMessage() << std::endl;
}
```

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>
```

### Code:

```
Aws::IAM::IAMClient iam;
Aws::IAM::Model::DeleteServerCertificateRequest request;
request.SetServerCertificateName(cert_name);

const auto outcome = iam.DeleteServerCertificate(request);
if (!outcome.IsSuccess())
{
    std::cout << "Error deleting server certificate " << cert_name <<
                ": " << outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted server certificate " << cert_name
                << std::endl;
}
```

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 Simple Storage Service (Amazon S3) is storage for the internet. You can use the following examples to program [Amazon S3](#) using the AWS SDK for C++. The S3 example is catered specifically to only work with CURL on Linux/Mac.

### Note

Only the code that is necessary to demonstrate each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

### Topics

- [Creating, Listing, and Deleting Buckets \(p. 85\)](#)
- [Operations on Objects \(p. 89\)](#)
- [Managing Amazon S3 Access Permissions \(p. 94\)](#)
- [Managing Access to Amazon S3 Buckets Using Bucket Policies \(p. 100\)](#)
- [Configuring an Amazon S3 Bucket as a Website \(p. 105\)](#)

## Creating, Listing, and Deleting Buckets

Every *object* or file in Amazon Simple Storage Service must reside within a *bucket* which represents a folder of objects. Each bucket is known by a name that must be globally unique within the AWS ecosystem. For detailed information about buckets and their configuration, see [Working with Amazon S3 Buckets](#) in the Amazon Simple Storage Service Developer Guide.

### Note

#### Best Practice

We recommend that you enable the [AbortIncompleteMultipartUpload](#) lifecycle rule on your Amazon S3 buckets.

This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data.

For more information, see [Lifecycle Configuration for a Bucket with Versioning](#) in the Amazon Simple Storage Service Console User Guide.

### 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 \(p. 6\)](#).

## Create a Bucket

Use the S3Client object `CreateBucket` method, passing it a `CreateBucketRequest` with the bucket's name. By default, buckets are created in the *us-east-1* (N. Virginia) region. The code example demonstrates how to create a bucket in any specified region.

### Includes

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

```
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/core/Utils/UUID.h>
#include <aws/core/Utils/StringUtils.h>
#include <awsdoc/s3/s3_examples.h>
```

## Code

```
bool AwsDoc::S3::CreateBucket(const Aws::String& bucketName,
    const Aws::S3::Model::BucketLocationConstraint& region)
{
    Aws::S3::S3Client s3_client;

    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    // You only need to set the AWS Region for the bucket if it is
    // other than US East (N. Virginia) us-east-1.
    if (region != Aws::S3::Model::BucketLocationConstraint::us_east_1)
    {
        Aws::S3::Model::CreateBucketConfiguration bucket_config;
        bucket_config.SetLocationConstraint(region);

        request.SetCreateBucketConfiguration(bucket_config);
    }

    Aws::S3::Model::CreateBucketOutcome outcome =
        s3_client.CreateBucket(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        std::cout << "Error: CreateBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }

    return true;
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::S3::Model::BucketLocationConstraint region =
            Aws::S3::Model::BucketLocationConstraint::us_east_1;

        // Create a unique bucket name to increase the chance of success
        // when trying to create the bucket.
        // Format: "my-bucket-" + lowercase UUID.
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String bucket_name = "my-bucket-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());

        // Create the bucket.
        if (AwsDoc::S3::CreateBucket(bucket_name, region))
        {
            std::cout << "Created bucket " << bucket_name <<
                " in the specified AWS Region." << std::endl;
        }
        else
        {

```

```
        return 1;
    }
}
ShutdownAPI(options);

return 0;
}
```

See the [complete example](#).

## List Buckets

Use the `S3Client` object `ListBucket` method. If successful, the method returns a `ListBucketOutcome` object, which contains a `ListBucketResult` object.

Use the `ListBucketResult` object `GetBuckets` method to get a list of `Bucket` objects that contain information about each Amazon S3 bucket in your account.

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/Bucket.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::ListBuckets()
{
    Aws::S3::S3Client s3_client;
    Aws::S3::Model::ListBucketsOutcome outcome = s3_client.ListBuckets();

    if (outcome.IsSuccess())
    {
        std::cout << "Bucket names:" << std::endl << std::endl;

        Aws::Vector<Aws::S3::Model::Bucket> buckets =
            outcome.GetResult().GetBuckets();

        for (Aws::S3::Model::Bucket& bucket : buckets)
        {
            std::cout << bucket.GetName() << std::endl;
        }

        return true;
    }
    else
    {
        std::cout << "Error: ListBuckets: "
            << outcome.GetError().GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        if (!AwsDoc::S3::ListBuckets())
```

```
        {
            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

See the [complete example](#).

## Delete a Bucket

Use the S3Client object DeleteBucket method, passing it a DeleteBucketRequest object that is set with the name of the bucket to delete. *The bucket must be empty or an error will result.*

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::DeleteBucket(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        s3_client.DeleteBucket(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        std::cout << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }

    return true;
}

int main()
{
    Aws::String bucket_name = "my-bucket";
    Aws::String region = "us-east-1";

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        if (AwsDoc::S3::DeleteBucket(bucket_name, region))
        {

```

```
        std::cout << "Deleted bucket '" << bucket_name <<
            "'." << std::endl;
    }
    else
    {
        return 1;
    }
}
ShutdownAPI(options);

return 0;
}
```

See the [complete example](#).

## Operations on Objects

An Amazon S3 object represents a *file*, which is a collection of data. Every object must reside within a [bucket](#) (p. 85).

### 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. 6).

## Upload an Object

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.

### Includes

```
#include <iostream>
#include <fstream>
#include <sys/stat.h>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::PutObject(const Aws::String& bucketName,
    const Aws::String& objectName,
    const Aws::String& region)
{
    // Verify that the file exists.
    struct stat buffer;

    if (stat(objectName.c_str(), &buffer) == -1)
    {
        std::cout << "Error: PutObject: File '" <<
            objectName << "' does not exist." << std::endl;

        return false;
    }

    Aws::Client::ClientConfiguration config;

    if (!region.empty())
    {
```

```

        config.region = region;
    }

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

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectName);

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

    request.SetBody(input_data);

    Aws::S3::Model::PutObjectOutcome outcome =
        s3_client.PutObject(request);

    if (outcome.IsSuccess()) {
        std::cout << "Added object '" << objectName << "' to bucket '"
            << bucketName << "'.";
        return true;
    }
    else
    {
        std::cout << "Error: PutObject: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        const Aws::String bucket_name = "my-bucket";
        const Aws::String object_name = "my-file.txt";
        const Aws::String region = "us-east-1";

        if (!AwsDoc::S3::PutObject(bucket_name, object_name, region)) {

            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}

```

See the [complete example](#).

## 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.

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/ListObjectsRequest.h>
#include <aws/s3/model/Object.h>
#include <awsdoc/s3/s3_examples.h>
```

## Code

```
bool AwsDoc::S3::ListObjects(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;

    if (!region.empty())
    {
        config.region = region;
    }

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

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

    auto outcome = s3_client.ListObjects(request);

    if (outcome.IsSuccess())
    {
        std::cout << "Objects in bucket '" << bucketName << "':"
            << std::endl << std::endl;

        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

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

        return true;
    }
    else
    {
        std::cout << "Error: ListObjects: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::String bucket_name = "my-bucket";
        Aws::String region = "us-east-1";

        if (!AwsDoc::S3::ListObjects(bucket_name, region))
        {
            return 1;
        }
    }
}
```

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

See the [complete example](#).

## 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 you can use 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.

### Includes

```
#include <iostream>  
#include <aws/core/Aws.h>  
#include <aws/s3/S3Client.h>  
#include <aws/s3/model/GetObjectRequest.h>  
#include <fstream>  
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::GetObject(const Aws::String& objectKey,  
    const Aws::String& fromBucket, const Aws::String& region)  
{  
    Aws::S3::S3Client s3_client;  
    Aws::S3::Model::GetObjectRequest object_request;  
    object_request.SetBucket(fromBucket);  
    object_request.SetKey(objectKey);  
  
    Aws::S3::Model::GetObjectOutcome get_object_outcome =  
        s3_client.GetObject(object_request);  
  
    if (get_object_outcome.IsSuccess())  
    {  
        auto& retrieved_file = get_object_outcome.GetResultWithOwnership().  
            GetBody();  
  
        // Print a beginning portion of the text file.  
        std::cout << "Beginning of file contents:\n";  
        char file_data[255] = { 0 };  
        retrieved_file.getline(file_data, 254);  
        std::cout << file_data << std::endl;  
  
        return true;  
    }  
    else  
    {  
        auto err = get_object_outcome.GetError();  
        std::cout << "Error: GetObject: " <<  
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;  
  
        return false;  
    }  
}  
  
int main()
```



```
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        const Aws::String bucket_name = "my-bucket";
        const Aws::String object_name = "my-file.txt";

        if (!AwsDoc::S3::GetObject(object_name, bucket_name))
        {
            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

See the [complete example](#).

## 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.*

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::DeleteObject(const Aws::String& objectKey,
    const Aws::String& fromBucket, const Aws::String& region)
{
    Aws::S3::S3Client s3_client;
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        s3_client.DeleteObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        std::cout << "Error: DeleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }
    else
    {
        return true;
    }
}

int main()
{
```

```
Aws::String object_key = "my-key";
Aws::String from_bucket = "my-bucket";

Aws::SDKOptions options;
Aws::InitAPI(options);
{
    if (AwsDoc::S3::DeleteObject(object_key, from_bucket))
    {
        std::cout << "Deleted object " << object_key <<
            " from " << from_bucket << "." << std::endl;
    }
}
ShutdownAPI(options);

return 0;
}
```

See the [complete example](#).

## 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.

### 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.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/model/Permission.h>
#include <aws/s3/model/Type.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/Owner.h>
#include <aws/s3/model/Grantee.h>
#include <aws/s3/model/Grant.h>
#include <aws/s3/model/AccessControlPolicy.h>
#include <aws/s3/model/GetObjectAclRequest.h>
#include <aws/s3/model/PutObjectAclRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

```
bool AwsDoc::S3::GetObjectAcl(const Aws::String& bucketName,
    const Aws::String& objectKey,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3_client.GetObjectAcl(request);
}
```

```
if (outcome.IsSuccess())
{
    Aws::S3::Model::Owner owner = outcome.GetResult().GetOwner();
    Aws::Vector<Aws::S3::Model::Grant> grants =
        outcome.GetResult().GetGrants();

    std::cout << "Object ACL information for object '" << objectKey <<
        "' in bucket '" << bucketName << "':" << std::endl << std::endl;

    std::cout << "Owner:" << std::endl << std::endl;
    std::cout << "Display name: " << owner.GetDisplayName() << std::endl;
    std::cout << "ID:          " << owner.GetID() << std::endl <<
        std::endl;

    std::cout << "Grantees:" << std::endl << std::endl;

    for (auto it = std::begin(grants); it != end(grants); ++it) {
        auto grantee = it->GetGrantee();
        std::cout << "Display name: " << grantee.GetDisplayName() <<
            std::endl;
        std::cout << "Email address: " << grantee.GetEmailAddress() <<
            std::endl;
        std::cout << "ID:          " << grantee.GetID() << std::endl;
        std::cout << "Type:          " << GetGranteeType(
            grantee.GetType()) <<
            std::endl;
        std::cout << "URI:          " << grantee.GetURI() << std::endl <<
            std::endl;
    }

    return true;
}
else
{
    auto error = outcome.GetError();
    std::cout << "Error: GetObjectAcl: " << error.GetExceptionName()
        << " - " << error.GetMessage() << std::endl;

    return false;
}
}
```

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. For further details, see [the example source file](#).

```
bool AwsDoc::S3::PutObjectAcl(const Aws::String& bucketName,
    const Aws::String& objectKey,
    const Aws::String& region,
    const Aws::String& ownerID,
    const Aws::String& granteePermission,
    const Aws::String& granteeType,
    const Aws::String& granteeID,
    const Aws::String& granteeDisplayName,
    const Aws::String& granteeEmailAddress,
    const Aws::String& granteeURI)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

```
Aws::S3::Model::Owner owner;
owner.SetID(ownerID);

Aws::S3::Model::Grantee grantee;
grantee.SetType(SetGranteeType(granteeType));

if (granteeEmailAddress != "")
{
    grantee.SetEmailAddress(granteeEmailAddress);
}

if (granteeID != "")
{
    grantee.SetID(granteeID);
}

if (granteeDisplayName != "")
{
    grantee.SetDisplayName(granteeDisplayName);
}

if (granteeURI != "")
{
    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())
{
    return true;
}
else
{
    auto error = outcome.GetError();
    std::cout << "Error: PutObjectAcl: " << error.GetExceptionName()
        << " - " << error.GetMessage() << std::endl;

    return false;
}
}
```

## 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. For details, see [the example source file](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/model/Permission.h>
#include <aws/s3/model/Type.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/Owner.h>
#include <aws/s3/model/Grantee.h>
#include <aws/s3/model/Grant.h>
#include <aws/s3/model/AccessControlPolicy.h>
#include <aws/s3/model/PutBucketAclRequest.h>
#include <aws/s3/model/GetBucketAclRequest.h>
#include <awsdoc/s3/s3_examples.h>

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;
}

Aws::String GetGranteeType(const Aws::S3::Model::Type& type)
{
    if (type == Aws::S3::Model::Type::AmazonCustomerByEmail)
        return "Amazon customer by email";
    if (type == Aws::S3::Model::Type::CanonicalUser)
        return "Canonical user";
    if (type == Aws::S3::Model::Type::Group)
        return "Group";
    return "Not set";
}

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;
}

bool AwsDoc::S3::PutBucketAcl(const Aws::String& bucketName,
    const Aws::String& ownerID,
    const Aws::String& granteePermission,
    const Aws::String& granteeType,
    const Aws::String& granteeID,
    const Aws::String& region,
    const Aws::String& granteeDisplayName,
```

```
const Aws::String& granteeEmailAddress,  
const Aws::String& granteeURI)  
{  
    Aws::Client::ClientConfiguration config;  
    config.region = region;  
  
    Aws::S3::S3Client s3_client(config);  
  
    Aws::S3::Model::Owner owner;  
    owner.SetID(ownerID);  
  
    Aws::S3::Model::Grantee grantee;  
    grantee.SetType(SetGranteeType(granteeType));  
  
    if (granteeEmailAddress != "")  
    {  
        grantee.SetEmailAddress(granteeEmailAddress);  
    }  
  
    if (granteeID != "")  
    {  
        grantee.SetID(granteeID);  
    }  
  
    if (granteeDisplayName != "")  
    {  
        grantee.SetDisplayName(granteeDisplayName);  
    }  
  
    if (granteeURI != "")  
    {  
        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())  
    {  
        return true;  
    }  
    else  
    {  
        auto error = outcome.GetError();  
        std::cout << "Error: PutBucketAcl: " << error.GetExceptionName()  
            << " - " << error.GetMessage() << std::endl;  
  
        return false;  
    }  
}
```

```

bool AwsDoc::S3::GetBucketAcl(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3_client.GetBucketAcl(request);

    if (outcome.IsSuccess())
    {
        Aws::S3::Model::Owner owner = outcome.GetResult().GetOwner();
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        std::cout << "Bucket ACL information for bucket '" << bucketName <<
            "'" << std::endl << std::endl;

        std::cout << "Owner:" << std::endl << std::endl;
        std::cout << "Display name: " << owner.GetDisplayName() << std::endl;
        std::cout << "ID: " << owner.GetID() << std::endl <<
            std::endl;

        std::cout << "Grantees:" << std::endl << std::endl;

        for (auto it = std::begin(grants); it != end(grants); ++it) {
            auto grantee = it->GetGrantee();
            std::cout << "Display name: " << grantee.GetDisplayName() <<
                std::endl;
            std::cout << "Email address: " << grantee.GetEmailAddress() <<
                std::endl;
            std::cout << "ID: " << grantee.GetID() << std::endl;
            std::cout << "Type: " << GetGranteeType(
                grantee.GetType()) <<
                std::endl;
            std::cout << "URI: " << grantee.GetURI() << std::endl <<
                std::endl;
        }

        return true;
    }
    else
    {
        auto error = outcome.GetError();
        std::cout << "Error: GetBucketAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        const Aws::String bucket_name = "my-bucket";

        // Set the ACL's owner information.
        const Aws::String owner_id =
            "b380d412791d395dbcdc1fb1728b32a7cd07edae6467220ac4b7c0769EXAMPLE";
    }
}

```

```
// Set the ACL's grantee information.
const Aws::String grantee_permission = "READ";

// If the grantee is by canonical user, then either the user's ID or
// display name must be specified:
const Aws::String grantee_type = "Canonical user";
const Aws::String grantee_id =
    "51ffd418eb142601651cc9d54984604a32b51a23153b4898fd2224772EXAMPLE";
// const Aws::String grantee_display_name = "janedoe";

// If the grantee is by Amazon customer by email, then the email
// address must be specified:
// const Aws::String grantee_type = "Amazon customer by email";
// const Aws::String grantee_email_address = "janedoe@example.com";

// If the grantee is by group, then the predefined group URI must
// be specified:
// const Aws::String grantee_type = "Group";
// const Aws::String grantee_uri =
//     "http://acs.amazonaws.com/groups/global/AuthenticatedUsers";

// Set the bucket's ACL.
if (!AwsDoc::S3::PutBucketAcl(bucket_name,
    "us-east-1",
    owner_id,
    grantee_permission,
    grantee_type,
    grantee_id))
    // grantee_display_name,
    // grantee_email_address,
    // grantee_uri));
{
    return 1;
}

// Get the bucket's ACL information that was just set.
if (!AwsDoc::S3::GetBucketAcl(bucket_name, "us-east-1"))
{
    return 1;
}
}
Aws::ShutdownAPI(options);

return 0;
}
```

## 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.

### 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 \(p. 6\)](#).

## 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`.



## Includes

```
#include <iostream>
#include <cstdio>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PutBucketPolicyRequest.h>
#include <aws/sts/STSCClient.h>
#include <aws/sts/model/GetCallerIdentityRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

## Code

```
"{\n"
"  \"Version\": \"2012-10-17\", \n"
"  \"Statement\": [\n"
"    {\n"
"      \"Sid\": \"1\", \n"
"      \"Effect\": \"Allow\", \n"
"      \"Principal\": {\n"
"        \"AWS\": \"arn:aws:iam::\" + accountID + \":root\"\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::String& region)
{
    Aws::Client::ClientConfiguration config;

    if (!region.empty())
    {
        config.region = region;
    }

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

    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::cout << "Set the following policy body for the bucket '" <<
            bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;

        return true;
    }
    else {
        std::cout << "Error: PutBucketPolicy: "
            << outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::String bucket_name = "my-bucket";
        Aws::String region = "us-east-1";

        // Get the caller's AWS account ID to be used in the bucket policy.
        Aws::STS::STSClient sts_client;
        Aws::STS::Model::GetCallerIdentityRequest request;
        Aws::STS::Model::GetCallerIdentityOutcome outcome =
            sts_client.GetCallerIdentity(request);

        if (!outcome.IsSuccess())
        {
            std::cout << "Error: GetBucketPolicy setup: Get identity information: "
                << outcome.GetError().GetMessage() << std::endl;

            return 1;
        }

        // Extract the caller's AWS account ID from the call to AWS STS.
        Aws::String account_id = outcome.GetResult().GetAccount();

        // Use the account ID and bucket name to form the bucket policy to be added.
        Aws::String policy_string = GetPolicyString(account_id, bucket_name);

        if (!AwsDoc::S3::PutBucketPolicy(bucket_name, policy_string, region))
        {
            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

### 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](#).

## 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`.

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/GetBucketPolicyRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::GetBucketPolicy(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3_client.GetBucketPolicy(request);

    if (outcome.IsSuccess())
    {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Policy:" << std::endl << std::endl <<
            policy_stream.str() << std::endl;

        return true;
    }
    else
    {
        auto err = outcome.GetError();
        std::cout << "Error: GetBucketPolicy: "
            << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::String bucket_name = "my-bucket";
    Aws::String region = "us-east-1";

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        if (!AwsDoc::S3::GetBucketPolicy(bucket_name, region))
        {
            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

See the [complete example](#).

## Delete a Bucket Policy

To delete a bucket policy, call the S3Client's `DeleteBucketPolicy` function, providing it with the bucket name in a [DeleteBucketPolicyRequest](#).

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/DeleteBucketPolicyRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

## Code

```
bool AwsDoc::S3::DeleteBucketPolicy(const Aws::String& bucketName, const Aws::String&
    region)
{
    Aws::S3::S3Client s3_client;

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
        s3_client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        std::cout << "Error: DeleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }

    return true;
}

int main()
{
    Aws::String bucket_name = "my-bucket";

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        if (AwsDoc::S3::DeleteBucketPolicy(bucket_name))
        {
            std::cout << "Deleted bucket policy from '" << bucket_name <<
                "'." << std::endl;
        }
        else
        {
            return 1;
        }
    }
    ShutdownAPI(options);

    return 0;
}
```

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](#).

## 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 Developer Guide
- [Bucket Policy Examples](#) in the Amazon Simple Storage Service Developer 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.

### 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. 6).

## 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.

### Includes

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/IndexDocument.h>
#include <aws/s3/model/ErrorDocument.h>
#include <aws/s3/model/WebsiteConfiguration.h>
#include <aws/s3/model/PutBucketWebsiteRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

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

    if (!region.empty())
    {
        config.region = region;
    }

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

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

    Aws::S3::Model::ErrorDocument error_doc;
    error_doc.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration website_config;
    website_config.SetIndexDocument(index_doc);
    website_config.SetErrorDocument(error_doc);
```

```
Aws::S3::Model::PutBucketWebsiteRequest request;
request.SetBucket(bucketName);
request.SetWebsiteConfiguration(website_config);

Aws::S3::Model::PutBucketWebsiteOutcome outcome =
    s3_client.PutBucketWebsite(request);

if (outcome.IsSuccess())
{
    std::cout << "Success: Set website configuration for bucket '"
        << bucketName << "'." << std::endl;

    return true;
}
else
{
    std::cout << "Error: PutBucketWebsite: "
        << outcome.GetError().GetMessage() << std::endl;

    return false;
}

return 1;
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        const Aws::String bucket_name = "my-bucket";
        const Aws::String index_page = "index.html";
        const Aws::String error_page = "404.html";
        const Aws::String region = "us-east-1";

        if (!AwsDoc::S3::PutWebsiteConfig(bucket_name, index_page, error_page, region))
        {
            return 1;
        }

    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

### 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 \(p. 100\)](#).

See the [complete example](#).

## 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.

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/GetBucketWebsiteRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

## Code

```
bool AwsDoc::S3::GetWebsiteConfig(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;

    if (!region.empty())
    {
        config.region = region;
    }

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

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3_client.GetBucketWebsite(request);

    if (outcome.IsSuccess())
    {
        Aws::S3::Model::GetBucketWebsiteResult result = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
            << std::endl << std::endl
            << "For bucket '" << bucketName << "':"
            << std::endl
            << "Index page : "
            << result.GetIndexDocument().GetSuffix()
            << std::endl
            << "Error page: "
            << result.GetErrorDocument().GetKey()
            << std::endl;

        return true;
    }
    else
    {
        auto err = outcome.GetError();

        std::cout << "Error: GetBucketWebsite: "
            << err.GetMessage() << std::endl;

        return false;
    }
}

int main()
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::String bucket_name = "my-bucket";
        Aws::String region = "us-east-1";

        if (!AwsDoc::S3::GetWebsiteConfig(bucket_name, region))
        {
```

```
        return 1;
    }
}
Aws::ShutdownAPI(options);

return 0;
}
```

See the [complete example](#).

## 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.

### Includes

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/DeleteBucketWebsiteRequest.h>
#include <awsdoc/s3/s3_examples.h>
```

### Code

```
bool AwsDoc::S3::DeleteBucketWebsite(const Aws::String& bucketName,
    const Aws::String& region)
{
    Aws::Client::ClientConfiguration config;
    config.region = region;

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

    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        s3_client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        std::cout << "Error: DeleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

        return false;
    }

    return true;
}

int main()
{
    Aws::String bucket_name = "my-bucket";
    Aws::String region = "us-east-1";

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        if (AwsDoc::S3::DeleteBucketWebsite(bucket_name, region))
        {
            std::cout << "Removed website configuration from '" <<
```



```
        bucket_name << "." << std::endl;
    }
}
ShutdownAPI(options);

return 0;
}
```

See the [complete example](#).

## 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

# 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 each technique is supplied here, but [complete example code is available on GitHub](#), where you can download a single source file or you can clone the repository locally to get all examples, build and run them.

### Topics

- [Working with Amazon SQS Message Queues](#) (p. 109)
- [Sending, Receiving, and Deleting Amazon SQS Messages](#) (p. 112)
- [Enabling Long Polling for Amazon SQS Message Queues](#) (p. 114)
- [Setting Visibility Timeout in Amazon SQS](#) (p. 117)
- [Using Dead Letter Queues in Amazon SQS](#) (p. 118)

## 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.

### 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. 6).

## 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 \(p. 111\)](#) 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 \(p. 109\)](#). These C++ examples show you how to use the AWS SDK for C++ to send, receive, and delete Amazon SQS messages from SQS queues.

### 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 \(p. 6\)](#).

## 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 \(p. 111\)](#), 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*.

### 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 \(p. 6\)](#).

## 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.

### 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 \(p. 6\)](#).

## 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 region as the source queue.

### 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 \(p. 6\)](#).

## 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

# 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 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& objectName,
    const Aws::String& region)
{
    // Verify that the file exists.
    struct stat buffer;

    if (stat(objectName.c_str(), &buffer) == -1)
    {
        std::cout << "Error: PutObjectAsync: File '" <<
            objectName << "' does not exist." << std::endl;

        return false;
    }

    // Create and configure the asynchronous put object request.
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectName);

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

    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(objectName);
}
```

```
// 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 upload_mutex;

// A condition_variable is a synchronization primitive that can be used to
// block a thread, or multiple threads at the same time, until another
// thread both modifies a shared variable (the condition) and
// notifies the condition_variable.
std::condition_variable 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::cout << "Error: PutObjectAsyncFinished: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    // Unblock the thread that is waiting for this function to complete.
    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);
    {
        const Aws::String bucket_name = "my-bucket";
        const Aws::String object_name = "my-file.txt";
        const Aws::String region = "us-east-1";

        // 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(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;

        if (!region.empty())
        {
            config.region = region;
        }

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

        if (AwsDoc::S3::PutObjectAsync(s3_client, bucket_name, object_name, region)) {

            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.
            upload_variable.wait(lock);

            std::cout << std::endl << "main: File upload attempt completed."
                << std::endl;
        }
        else
        {
            return 1;
        }
    }
    Aws::ShutdownAPI(options);

    return 0;
}
```

# 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++ \(p. 123\)](#)
- [Identity and Access Management for this AWS Product or Service \(p. 124\)](#)
- [Compliance Validation for this AWS Product or Service \(p. 124\)](#)
- [Resilience for this AWS Product or Service \(p. 125\)](#)
- [Infrastructure Security for this AWS Product or Service \(p. 125\)](#)
- [Enforcing TLS 1.2 in AWS SDK for C++ \(p. 125\)](#)
- [Amazon S3 Encryption Client Migration \(p. 128\)](#)

## 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 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 TLS 1.2 in 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
<code>TLS 1.0\Client\DisabledByDefault</code>	DWORD	0

Key name	Key type	Value
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 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. For example, you could 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.

### Applications Consuming the SDK from Source

If you build and install AWS SDK for C++ artifacts from source, pull the latest code from the following GitHub repository: <https://github.com/aws/aws-sdk-cpp>. 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 the README in our GitHub repository (<https://github.com/aws/aws-sdk-cpp>) or the AWS documentation (<https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup.html>).

## 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 `SetKMSToDecryptWithAnyCMK(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.SetKMSToDecryptWithAnyCMK(true);  
CryptoConfigurationV2 cryptoConfig(materials);  
S3EncryptionClientV2 encryptionClient(cryptoConfig);  
// Code snippet here to setup the getObjectRequest object.  
encryptionClient.GetObject(getObjectRequest);
```

For complete code examples for all of the migration scenarios mentioned in these docs, see [https://github.com/awsdocs/aws-doc-sdk-examples/tree/master/cpp/example\\_code/s3encryption](https://github.com/awsdocs/aws-doc-sdk-examples/tree/master/cpp/example_code/s3encryption).



# Document History for the AWS SDK for C++ Developer Guide

This topic lists recent changes to the AWS SDK for C++ Developer Guide.

- **Latest documentation update:** Dec 18, 2020

## **August 07, 2020**

Added a topic on Amazon S3 encryption client migration.

## **June 20, 2019**

Updated the Amazon S3 `CreateBucket` example to support regions.

## **April 26, 2019**

Added new section [Asynchronous Methods \(p. 119\)](#).

## **April 16, 2019**

Updated the SDK build instructions.

## **April 05, 2019**

Updated the [Service Client Classes \(p. 23\)](#) section.

## **April 03, 2019**

Updated the [Managing Amazon S3 Access Permissions \(p. 94\)](#) section.

## **March 01, 2019**

Updated the instructions for building the SDK. Updated the available AWS Client Configuration variables.

## **January 19, 2019**

Updated the instructions for setting up the `vcpkg` C++ package manager.

## **January 11, 2019**

Added new section [SDK Metrics \(p. 19\)](#).

## **January 07, 2019**

Fixed the inclusion of code samples.