
AWS SDK for Java

Developer Guide

Table of Contents

| | |
|---|----|
| | v |
| Developer guide - AWS SDK for Java 2.x | 1 |
| Get started with the SDK | 1 |
| Developing mobile applications | 1 |
| Maintenance and support for SDK major versions | 1 |
| Additional resources | 1 |
| Features not yet in the version 2 of the SDK | 2 |
| Contributing to the SDK | 2 |
| Quick start | 3 |
| Step 1: Set up for this tutorial | 3 |
| Create an account | 3 |
| Create an IAM user | 3 |
| Install Java and Apache Maven | 4 |
| Configure credentials | 4 |
| Step 2: Create the project | 4 |
| Step 3: Write the code | 6 |
| Step 4: Build and run the application | 7 |
| Success! | 8 |
| Cleanup | 8 |
| Next steps | 8 |
| Setting up | 9 |
| Overview | 9 |
| Create an account | 9 |
| Create an IAM user and programmatic access key | 9 |
| Set default credentials and Region | 10 |
| Setting the default credentials | 10 |
| Setting the default Region | 11 |
| Install Java and a build tool | 11 |
| Next steps | 12 |
| Setting up an Apache Maven project | 13 |
| Prerequisites | 13 |
| Create a Maven project | 14 |
| Configure the Java compiler for Maven | 14 |
| Declare the SDK as a dependency | 15 |
| Set dependencies for SDK modules | 15 |
| Build your project | 16 |
| Setting up a Gradle project | 17 |
| Setting up a GraalVM Native Image project | 18 |
| Prerequisites | 18 |
| Create a project using the archetype | 18 |
| Build a native image | 18 |
| Additional setup information | 19 |
| Set up credentials profiles | 19 |
| Migrating to version 2 | 20 |
| What's new | 20 |
| What's different between 1.x and 2.x | 21 |
| Using the SDK for Java 1.x and 2.x side-by-side | 29 |
| Using the SDK | 31 |
| Creating service clients | 31 |
| Using the default client | 31 |
| Making requests | 31 |
| Handling responses | 32 |
| Closing the client | 32 |
| Handling exceptions | 32 |

| | |
|---|----|
| Using waiters | 33 |
| Configuring service clients | 33 |
| HTTP clients | 34 |
| Retries | 34 |
| Timeouts | 34 |
| Execution interceptors | 34 |
| Additional information | 34 |
| Using credentials | 34 |
| Use the default credential provider chain | 35 |
| Use a specific credentials provider or provider chain | 36 |
| Use credentials profiles | 36 |
| Supply credentials explicitly | 37 |
| Configuring IAM roles for Amazon EC2 | 37 |
| AWS region selection | 38 |
| Choosing a region | 39 |
| Choosing a specific endpoint | 39 |
| Automatically determine the Region from the environment | 39 |
| Checking for service availability in a Region | 40 |
| Optimizing cold start performance for AWS Lambda | 40 |
| Example client configuration | 41 |
| HTTP configuration | 41 |
| Configuring the AWS CRT-based HTTP client | 42 |
| Configuring the Netty-based HTTP client | 44 |
| Exception handling | 45 |
| Why unchecked exceptions? | 45 |
| SdkServiceException (and subclasses) | 46 |
| SdkClientException | 46 |
| Logging AWS SDK for Java calls | 46 |
| Add the Log4J JAR | 46 |
| Log4j configuration file | 46 |
| Setting the classpath | 47 |
| Service-specific errors and warnings | 47 |
| Request/response summary logging | 48 |
| Verbose wire logging | 48 |
| Setting the JVM TTL for DNS name lookups | 49 |
| How to set the JVM TTL | 49 |
| SDK features | 50 |
| Asynchronous programming | 50 |
| Non-streaming operations | 50 |
| Streaming operations | 52 |
| Advanced operations | 55 |
| DynamoDB Enhanced Client | 56 |
| HTTP/2 | 56 |
| SDK Metrics | 56 |
| Prerequisites | 56 |
| How to enable metrics collection | 57 |
| What information is collected? | 58 |
| How can I use this information? | 58 |
| Service client metrics | 58 |
| Pagination | 61 |
| Synchronous pagination | 61 |
| Asynchronous pagination | 63 |
| S3 Transfer Manager (Preview) | 67 |
| Prerequisites | 67 |
| Imports | 67 |
| Using the Transfer Manager (Preview) | 68 |
| Upload a file to S3 | 68 |

| | |
|--|-----|
| Download a file from S3 | 68 |
| Writers | 69 |
| Prerequisites | 69 |
| Using waiters | 69 |
| Configuring waiters | 70 |
| Code examples | 71 |
| Working with AWS services | 72 |
| Guided code examples | 72 |
| AWS database services | 73 |
| Amazon Simple Storage Service (S3) | 75 |
| Amazon DynamoDB | 88 |
| Amazon EC2 | 105 |
| AWS Identity and Access Management (IAM) | 119 |
| Amazon Athena | 136 |
| Amazon CloudWatch | 136 |
| AWS CloudTrail | 146 |
| Amazon Cognito | 146 |
| Amazon Comprehend | 151 |
| Amazon EventBridge | 151 |
| Amazon Kinesis Data Firehose | 151 |
| Amazon Forecast | 151 |
| AWS Glue | 151 |
| Amazon Kinesis | 151 |
| AWS Key Management Service | 158 |
| AWS Lambda | 158 |
| AWS Elemental MediaConvert | 160 |
| AWS Elemental MediaStore | 161 |
| AWS Migration Hub | 161 |
| Amazon Personalize | 161 |
| Amazon Pinpoint | 161 |
| Amazon Polly | 168 |
| Amazon Rekognition | 169 |
| Amazon SageMaker | 169 |
| AWS Secrets Manager | 169 |
| Amazon Simple Email Service | 169 |
| Amazon Simple Notification Service | 169 |
| Amazon Simple Queue Service (SQS) | 173 |
| AWS Systems Manager | 178 |
| Amazon Simple Workflow Service | 178 |
| Amazon Textract | 178 |
| Amazon Transcribe | 178 |
| Amazon Translate | 182 |
| Amazon WorkDocs | 182 |
| Code examples | 183 |
| Single-service actions and scenarios | 183 |
| Cross-service examples | 442 |
| Security | 447 |
| Data protection | 447 |
| Enforcing a minimum TLS version | 448 |
| How to check the TLS version | 448 |
| Enforcing a minimum TLS version | 448 |
| Identity and access management | 448 |
| Compliance validation | 449 |
| Resilience | 449 |
| Infrastructure security | 450 |
| Document history | 451 |

You can now use the [Amazon S3 Transfer Manager \(Developer Preview\)](#) in the AWS SDK for Java 2.x for accelerated file transfers. Give it a try and [let us know what you think!](#)

Developer guide - AWS SDK for Java 2.x

The AWS SDK for Java provides a Java API for AWS services. Using the SDK, you can easily build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

The AWS SDK for Java 2.x is a major rewrite of the version 1.x code base. It's built on top of Java 8+ and adds several frequently requested features. These include support for non-blocking I/O and the ability to plug in a different HTTP implementation at run time. For more information see the [AWS blog](#).

We regularly add support for new services to the AWS SDK for Java. For a list of changes and features in a particular version, view the [change log](#).

Get started with the SDK

If you're ready to get hands-on with the SDK, follow the [Quick Start \(p. 3\)](#) tutorial.

To set up your development environment, see [Setting up \(p. 9\)](#).

If you're currently using version 1.x of the SDK for Java, see [Migrating to version 2 \(p. 20\)](#) for specific guidance.

For information on making requests to Amazon S3, DynamoDB, Amazon EC2 and other AWS services, see [Using the SDK for Java \(p. 31\)](#) and [Working with AWS services \(p. 72\)](#).

Developing mobile applications

If you're a mobile app developer, Amazon Web Services provides the [AWS Amplify](#) framework. See the [Amplify Documentation](#) for more information.

Maintenance and support for SDK major versions

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

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

Additional resources

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

- [AWS SDK for Java 2.x Reference](#)
- [Java developer blog](#)

- [Java developer forums](#)
- **GitHub:**
 - [Documentation source](#)
 - [SDK source](#)
- [The AWS Code Sample Catalog](#)
- [@awsforjava](#) (Twitter)

Features not yet in the version 2 of the SDK

See the following Github issues for details about additional features not yet in 2.x. Comments and feedback are also welcome.

- High-level libraries
 - [Amazon S3 Transfer manager](#)
 - [Amazon S3 Encryption Client](#)
 - [DynamoDB document APIs](#)
 - [DynamoDB Encryption Client](#)
 - [Amazon SQS Client-side Buffering](#)
- [Progress Listeners](#)

Contributing to the SDK

Developers can also contribute feedback through the following channels:

- **Submit issues on GitHub:**
 - [Submit documentation issues](#)
 - [Submit SDK issues](#)
- Join an informal chat about SDK on the AWS SDK for Java 2.x [gitter channel](#)
- Submit feedback anonymously to aws-java-sdk-v2-feedback@amazon.com. This email is monitored by the AWS SDK for Java team.
- Submit pull requests in the documentation or SDK source GitHub repositories to contribute to the SDK development.

Get started with the AWS SDK for Java 2.x

The AWS SDK for Java 2.x provides Java APIs for Amazon Web Services (AWS). Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

This tutorial shows you how you can use [Apache Maven](#) to define dependencies for the AWS SDK for Java and then write code that connects to Amazon S3 to upload a file.

Follow these steps to complete this tutorial:

- [Step 1: Set up for this tutorial \(p. 3\)](#)
- [Step 2: Create the project \(p. 4\)](#)
- [Step 3: Write the code \(p. 6\)](#)
- [Step 4: Build and run the application \(p. 7\)](#)

Step 1: Set up for this tutorial

Before you begin this tutorial, you need an active AWS account, an AWS Identity and Access Management (IAM) user with a programmatic access key and permissions to Amazon S3, and a Java development environment configured to use that access key as credentials for AWS.

Follow these steps to set up for this tutorial:

- [Create an AWS account \(p. 3\)](#)
- [Create an IAM user \(p. 3\)](#)
- [Install Java and Apache Maven \(p. 4\)](#)
- [Configure credentials \(p. 4\)](#)

Create an account

If you do not have an AWS account, visit [the Amazon Web Services signup page](#) and follow the on-screen prompts to create and activate a new account. For detailed instructions, see [How do I create and activate a new AWS account?](#)

After you activate your new AWS account, follow the instructions in [Creating your first IAM admin user and group](#) in the IAM User Guide. Use this account instead of the root account when accessing the AWS Console. For more information, see [IAM User Guide](#).

Create an IAM user

To complete this tutorial, you need to use credentials for an IAM user that has read and write access to Amazon S3. To make requests to Amazon Web Services using the AWS SDK for Java, create an access key to use as credentials.

1. Sign in to [the IAM console](#)

2. In the navigation pane on the left, choose **Users**. Then choose **Add user**.
3. Enter *TestSDK* as the **User name** and select the **Programmatic access** checkbox. Choose **Next: Permissions**.
4. Under **Set permissions**, select **Attach existing policies directly**.
5. In the list of policies, select the checkbox for the **AmazonS3FullAccess** policy. Choose **Next: Tags**.
6. Choose **Next: Review**. Then choose **Create user**.
7. On the *Success* screen, choose **Download .csv**.

The downloaded file contains the Access Key ID and the Secret Access Key for this tutorial. Treat your Secret Access Key as a password; save in a trusted location and do not share it.

Note

You will **not** have another opportunity to download or copy the Secret Access Key.

Install Java and Apache Maven

Your development environment needs to have Java 8 or later and Apache Maven installed.

- For Java, use [Oracle Java SE Development Kit](#), [Amazon Corretto](#), [Red Hat OpenJDK](#), or [AdoptOpenJDK](#).
- For Maven, go to <https://maven.apache.org/>.

Configure credentials

Configure your development environment with your Access Key ID and the Secret Access Key. The AWS SDK for Java uses this access key as credentials when your application makes requests to Amazon Web Services.

1. In a text editor, create a new file with the following code:

```
[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
```

2. In the text file you just created, replace *YOUR_AWS_ACCESS_KEY_ID* with your unique AWS access key ID, and replace *YOUR_AWS_SECRET_ACCESS_KEY* with your unique AWS secret access key.
3. Save the file without a file extension. Refer to the following table for the correct location and file name based on your operating system.

| Operating system | File name |
|--------------------|---|
| Windows | C:\Users\<<yourUserName>\.aws \credentials |
| Linux, macOS, Unix | ~/.aws/credentials |

Step 2: Create the project

To create the project for this tutorial, you first create a Maven project. Next, you configure your project with a dependency on AWS SDK for Java and for any AWS service you use, for example Amazon S3. Then you configure the Maven compiler to use Java 1.8.

1. Open a terminal or command prompt window and navigate to a directory of your choice, for example, your Desktop or Home folder.
2. Use the following command to create a new directory called `myapp` with a project configuration file (`pom.xml`) and a basic Java class.

```
mvn -B archetype:generate \  
-DarchetypeGroupId=org.apache.maven.archetypes \  
-DgroupId=com.example.myapp \  
-DartifactId=myapp
```

To configure your project with dependencies for the AWS SDK for Java and Amazon S3, and to use Java 1.8

- In the folder `myapp` that you created in the previous procedure, open the `pom.xml` file. Replace its contents with the following code, and then save your changes.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/  
XMLSchema-instance"  
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-  
v4_0_0.xsd">  
  <modelVersion>4.0.0</modelVersion>  
  <properties>  
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
  </properties>  
  <groupId>com.example.myapp</groupId>  
  <artifactId>myapp</artifactId>  
  <packaging>jar</packaging>  
  <version>1.0-SNAPSHOT</version>  
  <name>myapp</name>  
  <dependencyManagement>  
    <dependencies>  
      <dependency>  
        <groupId>software.amazon.awssdk</groupId>  
        <artifactId>bom</artifactId>  
        <version>2.16.60</version>  
        <type>pom</type>  
        <scope>import</scope>  
      </dependency>  
    </dependencies>  
  </dependencyManagement>  
  <dependencies>  
    <dependency>  
      <groupId>junit</groupId>  
      <artifactId>junit</artifactId>  
      <version>3.8.1</version>  
      <scope>test</scope>  
    </dependency>  
    <dependency>  
      <groupId>software.amazon.awssdk</groupId>  
      <artifactId>s3</artifactId>  
    </dependency>  
  </dependencies>  
  <build>  
    <plugins>  
      <plugin>  
        <groupId>org.apache.maven.plugins</groupId>  
        <artifactId>maven-compiler-plugin</artifactId>  
        <version>3.8.1</version>  
        <configuration>  
          <source>8</source>  
          <target>8</target>
```

```
</configuration>
</plugin>
</plugins>
</build>
</project>
```

The `dependencyManagement` section contains a dependency to the AWS SDK for Java and the `dependencies` section has a dependency for Amazon S3. The Apache Maven Compiler Plugin is configured in the `build` section to use Java 1.8.

Step 3: Write the code

After the project has been created and configured, edit the project's default class `App` to use the example code below.

The example class below creates a service client for Amazon S3 and then uses it to upload a text file. To create a service client for Amazon S3, instantiate an `S3Client` object using the static factory method `builder`. To upload a file to Amazon S3, first build a `PutObjectRequest` object, supplying a bucket name and a key name. Then, call the `S3Client`'s `putObject` method, with a `RequestBody` that contains the object content and the `PutObjectRequest` object.

1. In your project folder `myapp`, navigate to the directory `src/main/java/com/example/myapp`. Open the `App.java` file.
2. Replace its contents with the following code and save the file.

```
package com.example.myapp;

import java.io.IOException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.S3Client;

public class App {

    public static void main(String[] args) throws IOException {

        Region region = Region.US_WEST_2;
        S3Client s3 = S3Client.builder().region(region).build();

        String bucket = "bucket" + System.currentTimeMillis();
        String key = "key";

        tutorialSetup(s3, bucket, region);

        System.out.println("Uploading object...");

        s3.putObject(PutObjectRequest.builder().bucket(bucket).key(key)
            .build(),
            RequestBody.fromString("Testing with the {sdk-java}"));

        System.out.println("Upload complete");
    }
}
```

```
        System.out.printf("%n");

        cleanUp(s3, bucket, key);

        System.out.println("Closing the connection to {S3}");
        s3.close();
        System.out.println("Connection closed");
        System.out.println("Exiting...");
    }

    public static void tutorialSetup(S3Client s3Client, String bucketName, Region region) {
        try {
            s3Client.createBucket(CreateBucketRequest
                .builder()
                .bucket(bucketName)
                .createBucketConfiguration(
                    CreateBucketConfiguration.builder()
                        .locationConstraint(region.id())
                        .build())
                .build());
            System.out.println("Creating bucket: " + bucketName);
            s3Client.waitFor().waitUntilBucketExists(HeadBucketRequest.builder()
                .bucket(bucketName)
                .build());
            System.out.println(bucketName + " is ready.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
        System.out.println("Cleaning up...");
        try {
            System.out.println("Deleting object: " + keyName);
            DeleteObjectRequest deleteObjectRequest =
                DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
            s3Client.deleteObject(deleteObjectRequest);
            System.out.println(keyName + " has been deleted.");
            System.out.println("Deleting bucket: " + bucketName);
            DeleteBucketRequest deleteBucketRequest =
                DeleteBucketRequest.builder().bucket(bucketName).build();
            s3Client.deleteBucket(deleteBucketRequest);
            System.out.println(bucketName + " has been deleted.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Cleanup complete");
        System.out.printf("%n");
    }
}
```

Step 4: Build and run the application

After the project is created and contains the example class, build and run the application. To view the uploaded file in the Amazon S3 console, edit the code to remove the cleanup steps and then rebuild the project.

1. Open a terminal or command prompt window and navigate to your project directory `myapp`.

2. Use the following command to build your project:

```
mvn package
```

3. Open a terminal or command prompt window and navigate to your project directory `myapp`.
4. Use the following command to run the application.

```
mvn exec:java -Dexec.mainClass="com.example.myapp.App"
```

When you run the application, it uploads a new a text file to a new bucket in Amazon S3. Afterward, it will also delete the file and bucket.

1. In `App.java`, comment out the line `cleanup(s3, bucket, key)`; and save the file.
2. Rebuild the project by running `mvn package`.
3. Upload the file by running `mvn exec:java -Dexec.mainClass="com.example.myapp.App"` again.
4. Sign in to [the S3 console](#) to view the new file in the newly-created bucket.

After you view the file, clean up test resources by deleting the object and then deleting the bucket.

Success!

If your Maven project built and ran without error, then congratulations! You have successfully built your first Java application using the AWS SDK for Java.

Cleanup

To clean up the resources you created during this tutorial:

- In [the S3 console](#), delete any objects and any buckets created when you ran the application.
- In [the IAM console](#), delete the `TestSDK` user.

If you delete this user, also remove the contents of the `credentials` file you created during setup.

- Delete the project folder (`myapp`).

Next steps

Now that you have the basics down, you can learn about:

- [Working with Amazon S3 \(p. 75\)](#)
- [Working with other Amazon Web Services \(p. 72\)](#), such as [DynamoDB \(p. 88\)](#), [Amazon EC2 \(p. 105\)](#), and [IAM \(p. 119\)](#)
- [Using the SDK \(p. 31\)](#)
- [Security for the AWS SDK for Java \(p. 447\)](#)

Setting up the AWS SDK for Java 2.x

The AWS SDK for Java 2.x provides Java APIs for Amazon Web Services (AWS). Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

This section provides information about how to set up your development environment and projects to use the latest version (2.x) of the AWS SDK for Java.

Overview

To make requests to AWS using the AWS SDK for Java, you need the following:

- An active AWS account
- An AWS Identity and Access Management (IAM) user with:
 - A programmatic access key
 - Permissions to the AWS resources you'll access using your application
- A development environment with:
 - Your access key configured as credentials for AWS
 - Java 8 or later
 - A build automation tool

Create an account

If you do not have an AWS account, visit [the Amazon Web Services signup page](#) and follow the on-screen prompts to create and activate a new account.

For more detailed instructions, see [How do I create and activate a new AWS account?](#).

After you activate your new AWS account, follow the instructions in [Creating your first IAM admin user and group](#) in the [IAM User Guide](#). Use this account instead of the root account when accessing the AWS Management Console. For more information, see [Security best practices in IAM](#) in the [IAM User Guide](#).

Create an IAM user and programmatic access key

To use the AWS SDK for Java to access AWS services, you need an AWS account and AWS credentials. To increase the security of your AWS account, for access credentials, we recommend that you use an IAM user instead of your AWS account credentials.

Note

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.

For instructions on creating an access key for an existing IAM user, see [Programmatic access](#) in the [IAM User Guide](#).

1. Go to the [IAM console](#) (you may need to sign in to AWS first).
2. Click **Users** in the sidebar to view your IAM users.
3. If you don't have any IAM users set up, click **Create New Users** to create one.
4. Select the IAM user in the list that you'll use to access AWS.

5. Open the **Security Credentials** tab, and click **Create Access Key**. NOTE: You can have a maximum of two active access keys for any given IAM user. If your IAM user has two access keys already, then you'll need to delete one of them before creating a new key.
6. On the resulting dialog box, click the **Download Credentials** button to download the credential file to your computer, or click **Show User Security Credentials** to view the IAM user's access key ID and secret access key (which you can copy and paste).

Important

There is no way to obtain the secret access key once you close the dialog box. You can, however, delete its associated access key ID and create a new one.

Set default credentials and Region

To make requests to AWS using the AWS SDK for Java, you must use cryptographically-signed credentials issued by AWS. With AWS SDKs and Tools like the AWS SDK for Java, you use a programmatic access key, consisting of an Access Key ID and a Secret Access Key, as credentials. You should set your credentials as the default credentials for accessing AWS with your application.

If you already have an IAM account created, see [Create an IAM user and programmatic access key \(p. 9\)](#) for instructions on creating a programmatic access key.

You should also set a default AWS Region for accessing AWS with your application. Some operations require a Region to be set. For the best network performance, you can select a Region that is geographically near to you or your customers.

The most common way to set the default credentials and AWS Region is to use the shared `config` and `credentials` files. You can also set the default credentials and Region using environment variables, using Java system properties or, for your applications running on Amazon EC2, using [ContainerCredentialsProvider](#) or [InstanceProfileCredentialsProvider](#).

Setting the default credentials

Select one of these options to set the default credentials:

- Set credentials in the AWS credentials profile file on your local system, located at:
 - `~/.aws/credentials` on Linux, macOS, or Unix
 - `C:\Users\USERNAME\.aws\credentials` on Windows

This file should contain lines in the following format:

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Substitute your own AWS credentials values for the values `your_access_key_id` and `your_secret_access_key`.

- Set the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables.

To set these variables on Linux, macOS, or Unix, use **export** :

```
export AWS_ACCESS_KEY_ID=your_access_key_id
```

```
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

To set these variables on Windows, use **set** :

```
set AWS_ACCESS_KEY_ID=your_access_key_id  
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

- For an Amazon EC2 instance, specify an IAM role and then give your Amazon EC2 instance access to that role. See [IAM Roles for Amazon EC2](#) in the Amazon EC2 User Guide for Linux Instances for a detailed discussion about how this works.
- Set the `aws.accessKeyId` and `aws.secretAccessKey` Java system properties.

```
java app.jar -Daws.accessKeyId=\  
"your_access_key_id" \  
-Daws.secretAccessKey=\  
"your_secret_access_key"
```

Setting the default Region

Select one of these options to set the default Region:

- Set the AWS Region in the AWS config file on your local system, located at:
 - `~/.aws/config` on Linux, macOS, or Unix
 - `C:\Users\USERNAME\.aws\config` on Windows

This file should contain lines in the following format:

```
[default]  
region = your_aws_region
```

Substitute your desired AWS Region (for example, "us-east-1") for *your_aws_region*.

- Set the `AWS_REGION` environment variable.

On Linux, macOS, or Unix, use **export** :

```
export AWS_REGION=your_aws_region
```

On Windows, use **set** :

```
set AWS_REGION=your_aws_region
```

Where *your_aws_region* is the desired AWS Region name.

For additional information about setting credentials and Region, see [The .aws/credentials and .aws/config files](#), [AWS Region](#), and [Using environment variables](#) in the [AWS SDKs and Tools Reference Guide](#).

Install Java and a build tool

Your development environment needs the following:

- Java 8 or later. The AWS SDK for Java works with the [Oracle Java SE Development Kit](#) and with distributions of Open Java Development Kit (OpenJDK) such as [Amazon Corretto](#), [Red Hat OpenJDK](#), and [AdoptOpenJDK](#).
- A build tool or IDE that supports Maven Central such as Apache Maven, Gradle, or IntelliJ.
 - For information about how to install and use Maven, see <http://maven.apache.org/>.
 - For information about how to install and use Gradle, see <https://gradle.org/>.
 - For information about how to install and use IntelliJ IDEA, see <https://www.jetbrains.com/idea/>.

Next steps

Once you have your AWS account and development environment set up, create a Java project using your preferred build tool. Check [the Maven bill of materials \(BOM\) for the AWS SDK for Java 2.x from Maven Central](#), for the latest version. Use that version number in the `<dependencyManagement>` section below. Then add dependencies for the services you'll use in your application.

Example Maven pom.xml file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <groupId>com.example.myapp</groupId>
  <artifactId>myapp</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>myapp</name>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.17.261</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>iam</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>kinesis</artifactId>
    </dependency>
  </dependencies>
</project>
```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
</dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>8</source>
        <target>8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Example build.gradle file:

```
group 'com.example.myapp'
version '1.0'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
  mavenCentral()
}

dependencies {
  implementation platform('software.amazon.awssdk:bom:2.15.0')
  implementation 'software.amazon.awssdk:dynamodb'
  implementation 'software.amazon.awssdk:iam'
  implementation 'software.amazon.awssdk:kinesis'
  implementation 'software.amazon.awssdk:s3'
  testImplementation group: 'junit', name: 'junit', version: '4.11'
}
```

For more information, see [Setting up an Apache Maven project \(p. 13\)](#) or [Setting up a Gradle project \(p. 17\)](#).

Setting up an Apache Maven project

You can use [Apache Maven](#) to set up and build AWS SDK for Java projects, or to build the SDK itself.

Prerequisites

To use the AWS SDK for Java with Maven, you need the following:

- **Java 8.0 or later.** You can download the latest Java SE Development Kit software from <http://www.oracle.com/technetwork/java/javase/downloads/>. The AWS SDK for Java also works with [OpenJDK](#) and Amazon Corretto, a distribution of the Open Java Development Kit (OpenJDK). Download the latest OpenJDK version from <https://openjdk.java.net/install/index.html>. Download the latest Amazon Corretto 8 or Amazon Corretto 11 version from [the Corretto page](#).

- *Apache Maven*. If you need to install Maven, go to <http://maven.apache.org/> to download and install it.

Create a Maven project

To create a Maven project from the command line, open a terminal or command prompt window, enter or paste the following command, and then press Enter or Return.

```
mvn -B archetype:generate \  
-DarchetypeGroupId=software.amazon.awssdk \  
-DarchetypeArtifactId=archetype-lambda -Dservice=s3 -Dregion=US_WEST_2 \  
-DgroupId=com.example.myapp \  
-DartifactId=myapp
```

Note

Replace *com.example.myapp* with the full package namespace of your application. Also replace *myapp* with your project name. This becomes the name of the directory for your project.

This command creates a Maven project using the AWS Lambda project archetype. This project archetype is preconfigured to compile with Java SE 8 and includes a dependency to the AWS SDK for Java.

For more information about creating and configuring Maven projects, see the [Maven Getting Started Guide](#).

Configure the Java compiler for Maven

If you created your project using the AWS Lambda project archetype as described earlier, this is already done for you.

To verify that this configuration is present, start by opening the `pom.xml` file from the project folder you created (for example, `myapp`) when you executed the previous command. Look on lines 11 and 12 to see the Java compiler version setting for this Maven project, and the required inclusion of the Maven compiler plugin on lines 71-75.

```
<project>  
  <properties>  
    <maven.compiler.source>1.8</maven.compiler.source>  
    <maven.compiler.target>1.8</maven.compiler.target>  
  </properties>  
  <build>  
    <plugins>  
      <plugin>  
        <groupId>org.apache.maven.plugins</groupId>  
        <artifactId>maven-compiler-plugin</artifactId>  
        <version>${maven.compiler.plugin.version}</version>  
      </plugin>  
    </plugins>  
  </build>  
</project>
```

If you create your project with a different archetype or by using another method, you must ensure that the Maven compiler plugin is part of the build and that its source and target properties are both set to **1.8** in the `pom.xml` file.

See the previous snippet for one way to configure these required settings.

Alternatively, you can configure the compiler configuration inline with the plugin declaration, as follows.

```
<project>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Declare the SDK as a dependency

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

If you created your project using the project archetype as described earlier, the SDK is already configured as a dependency in your project. We recommend that you update this configuration to reference the latest version of the AWS SDK for Java. To do so, open the `pom.xml` file and change the `aws.java.sdk.version` property (on line 16) to the latest version. The following is an example.

```
<project>
  <properties>
    <aws.java.sdk.version>2.16.1</aws.java.sdk.version>
  </properties>
</project>
```

Find the latest version of the AWS SDK for Java in the [AWS SDK for Java API Reference version 2.x](#).

If you created your Maven project in a different way, configure the latest version of the SDK for your project by ensuring that the `pom.xml` file contains the following.

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.X.X</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

Note

Replace `2.X.X` in the `pom.xml` file with a valid version of the AWS SDK for Java version 2.

Set dependencies for SDK modules

Now that you have configured the SDK, you can add dependencies for one or more of the AWS SDK for Java modules to use in your project.

Although you can specify the version number for each component, you don't need to because you already declared the SDK version in the `dependencyManagement` section. To load a custom version of a given module, specify a version number for its dependency.

If you created your project using the project archetype as described earlier, your project is already configured with multiple dependencies. These include dependences for Lambda and Amazon DynamoDB, as follows.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-lambda-java-core</artifactId>
      <version>1.2.0</version>
    </dependency>
  </dependencies>
</project>
```

Add the modules to your project for the AWS service and features you need for your project. The modules (dependencies) that are managed by the AWS SDK for Java BOM are listed on the Maven central repository (<https://mvnrepository.com/artifact/software.amazon.awssdk/bom/latest>).

Note

You can look at the `pom.xml` file from a code example to determine which dependencies you need for your project. For example, if you're interested in the dependencies for the Amazon S3 service, see [this example](#) from the [AWS Code Examples Repository](#) on GitHub. (Look for the `pom.xml` file under `/java2/example_code/s3`.)

Build the entire SDK into your project

To optimize your application, we strongly recommend that you pull in only the components you need instead of the entire SDK. However, to build the entire AWS SDK for Java into your project, declare it in your `pom.xml` file, as follows.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-sdk-java</artifactId>
      <version>2.X.X</version>
    </dependency>
  </dependencies>
</project>
```

Build your project

After you configure the `pom.xml` file, you can use Maven to build your project.

To build your Maven project from the command line, open a terminal or command prompt window, navigate to your project directory (for example, `myapp`), enter or paste the following command, then press Enter or Return.

```
mvn package
```

This creates a single `.jar` file (JAR) in the `target` directory (for example, `myapp/target`). This JAR contains all of the SDK modules you specified as dependencies in your `pom.xml` file.

Setting up a Gradle project

You can use [Gradle](#) to set up and build AWS SDK for Java projects.

To manage SDK dependencies for your Gradle project, import the Maven bill of materials (BOM) for the AWS SDK for Java into the `build.gradle` file.

Note

In the following examples, replace `2.15.0` in the `build.gradle` file with the latest version of the AWS SDK for Java v2. Find the latest version in the [AWS SDK for Java API Reference version 2.x](#).

1. Add the BOM to the `dependencies` section of the file.

```
...
dependencies {
    implementation platform('software.amazon.awssdk:bom:2.15.0')

    // Declare individual SDK dependencies without version
    ...
}
```

2. Specify the SDK modules to use in the `dependencies` section. For example, the following includes a dependency for Amazon Kinesis.

```
...
dependencies {
    ...
    implementation 'software.amazon.awssdk:kinesis'
    ...
}
```

Gradle automatically resolves the correct version of your SDK dependencies by using the information from the BOM.

The following is an example of a complete `build.gradle` file that includes a dependency for Kinesis.

```
group 'aws.test'
version '1.0'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.15.0')
    implementation 'software.amazon.awssdk:kinesis'
    testImplementation group: 'junit', name: 'junit', version: '4.11'
}
```

Note

In the previous example, replace the dependency for Kinesis with the dependencies of the AWS services you will use in your project. The modules (dependencies) that are managed by the AWS SDK for Java BOM are listed on Maven central repository (<https://mvnrepository.com/artifact/software.amazon.awssdk/bom/latest>).

For more information about specifying SDK dependencies by using the BOM, see [Setting up an Apache Maven project \(p. 13\)](#).

Setting up a GraalVM Native Image project for the AWS SDK for Java

With versions 2.16.1 and later, the AWS SDK for Java provides out-of-the-box support for GraalVM Native Image applications. Use the `archetype-app-quickstart` Maven archetype to set up a project with built-in native image support.

Prerequisites

- Complete the steps in [Setting up the AWS SDK for Java 2.x \(p. 9\)](#).
- Install [GraalVM Native Image](#).

Create a project using the archetype

To create a Maven project with built-in native image support, in a terminal or command prompt window, use the following command.

Note

Replace `com.example.mynativeimageapp` with the full package namespace of your application. Also replace `mynativeimageapp` with your project name. This becomes the name of the directory for your project.

```
mvn archetype:generate \  
  -DarchetypeGroupId=software.amazon.awssdk \  
  -DarchetypeArtifactId=archetype-app-quickstart \  
  -DarchetypeVersion=2.16.1 \  
  -DnativeImage=true \  
  -DhttpClient=apache-client \  
  -Dservice=s3 \  
  -DgroupId=com.example.mynativeimageapp \  
  -DartifactId=mynativeimageapp \  
  -DinteractiveMode=false
```

This command creates a Maven project configured with dependencies for the AWS SDK for Java, Amazon S3, and the `ApacheHttpClient` HTTP client. It also includes a dependency for the [GraalVM Native Image Maven plugin](#), so that you can build native images using Maven.

To include dependencies for a different Amazon Web Services, set the value of the `-Dservice` parameter to the artifact ID of that service. For example, `dynamodb`, `iam`, `pinpoint`, etc. For a complete list of artifact IDs, see the list of managed dependencies for [software.amazon.awssdk on Maven Central](#).

To use an asynchronous HTTP client, set the `-DhttpClient` parameter to `netty-nio-client`. To use `URLConnectionHttpClient` as the synchronous HTTP client instead of `apache-client`, set the `-DhttpClient` parameter to `url-connection-client`.

Build a native image

After you create the project, run the following command from your project directory, for example, `mynativeimageapp`:

```
mvn package -P native-image
```

This creates a native image application in the `target` directory, for example, `target/mynativeimageapp`.

Additional setup information

This topic supplements the information in [Setting up the AWS SDK for Java 2.x \(p. 9\)](#).

Set up credentials profiles

You can use more than one set of credentials in your application by setting up additional credentials profiles. Like the `[default]` profile, you can set up custom profiles to use programmatic access keys as credentials or to use temporary credentials.

To configure your own credentials profiles, use the shared `credentials` and `config` files. See the snippets below for example usage.

*A profile, `*cloudwatch_metrics`, configured in the `credentials` file to use a programmatic access key as credentials:

```
[cloudwatch_metrics]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
region = us-east-2
```

A profile, `devuser`, configured in the `config` file to use temporary credentials by assuming a role based on an Amazon Resource Name (ARN).

```
[profile devuser]
role_arn = {region-arn}iam::123456789012:role/developers
source_profile = dev-user
region = {region_api_default}
output = json
```

For an example using single sign-on (SSO) authentication, using AWS IAM Identity Center (successor to AWS Single Sign-On), see [SSO Credentials](#) in the *AWS SDKs and Tools Reference Guide*. In addition to `config` file changes, a dependency must be added to your project's `pom.xml` file:

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>sso</artifactId>
  <version>xxx</version>
</dependency>
```

For additional information about configuring the shared `credentials` and `config` files, see:

- [Set default credentials and Region \(p. 10\)](#)
- [Example config and credentials files](#)
- [The `.aws/credentials` and `.aws/config` files](#)
- [Credentials for an IAM role assumed as an IAM user](#)

Set an alternate credentials file location

By default, the AWS SDK for Java looks for the `credentials` file at `~/.aws/credentials`. To customize the location of the shared `credentials` file, set the `AWS_SHARED_CREDENTIALS_FILE` environment variable to an alternate location.

To set this variable on Linux, macOS, or Unix, use `export`:

```
export AWS_SHARED_CREDENTIALS_FILE=path/to/credentials_file
```

To set this variable on Windows, use `set`:

```
set AWS_SHARED_CREDENTIALS_FILE=path/to/credentials_file
```

Migrating from version 1.x to 2.x of the AWS SDK for Java

The AWS SDK for Java 2.x is a major rewrite of the 1.x code base built on top of Java 8+. It includes many updates, such as improved consistency, ease of use, and strongly enforced immutability. This section describes the major features that are new in version 2.x, and provides guidance on how to migrate your code to version 2.x from 1.x.

Topics

- [What's new \(p. 20\)](#)
- [What's different between the AWS SDK for Java 1.x and 2.x \(p. 21\)](#)
- [Using the SDK for Java 1.x and 2.x side-by-side \(p. 29\)](#)

What's new

- You can configure your own HTTP clients. See [HTTP Transport Configuration \(p. 41\)](#).
- Async clients are now truly nonblocking and return `CompletableFuture` objects. See [Asynchronous programming \(p. 50\)](#).
- Operations that return multiple pages have autopaginated responses. This enables you to focus your code on what to do with the response, without the need to check for and get subsequent pages. See the [Pagination \(p. 61\)](#)
- SDK start time performance for AWS Lambda functions is improved. See [SDK Start Time Performance Improvements \(p. 40\)](#)
- Version 2.x supports a new shorthand method for creating requests.

Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

For more details about the new features and to see specific code examples, refer to the other sections of this guide.

- [Quick Start \(p. 3\)](#)

- [Setting up \(p. 9\)](#)
- [Code examples for the AWS SDK for Java 2.x \(p. 72\)](#)
- [Using the SDK \(p. 31\)](#)
- [Security for the AWS SDK for Java \(p. 447\)](#)

What's different between the AWS SDK for Java 1.x and 2.x

This section describes the main changes to be aware of when converting an application from using the AWS SDK for Java version 1.x to version 2.x.

High-Level libraries

High-level libraries, such as the Amazon S3 Transfer Manager and the Amazon SQS Client-side Buffering, are not yet available in version 2.x. See the AWS SDK for Java 2.x [changelog](#) for a list of libraries that have been released.

If your application depends on libraries that have not yet been released in version 2.x, see [Using both SDKs side-by-side \(p. 29\)](#) to learn how to configure your pom.xml to use both 1.x and 2.x.

Adding version 2.x to Your Project

Maven is the recommended way to manage dependencies when using the AWS SDK for Java 2.x. To add version 2 components to your project, simply update your pom.xml file with a dependency on the SDK.

Example

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

Client builders

You must create all clients using the client builder method. Constructors are no longer available.

Example of creating a client in version 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
```

```
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example of creating a client in version 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();  
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

Client Configuration

In 1.x, SDK client configuration was modified by setting a `ClientConfiguration` instance on the client or client builder. In version 2.x, the client configuration is split into separate configuration classes. The separate configuration classes enable you to configure different HTTP clients for async versus synchronous clients but still use the same `ClientOverrideConfiguration` class.

Example of client configuration in version 1.x

```
AmazonDynamoDBClientBuilder.standard()  
.withClientConfiguration(clientConfiguration)  
.build()
```

Example of synchronous client configuration in version 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();  
  
ApacheHttpClient.Builder httpClientBuilder =  
    ApacheHttpClient.builder()  
        .proxyConfiguration(proxyConfig.build());  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
DynamoDbClient client =  
    DynamoDbClient.builder()  
        .httpClientBuilder(httpClientBuilder)  
        .overrideConfiguration(overrideConfig.build())  
        .build();
```

Example of asynchronous client configuration in version 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =  
    NettyNioAsyncHttpClient.builder();  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
ClientAsyncConfiguration.Builder asyncConfig =  
    ClientAsyncConfiguration.builder();  
  
DynamoDbAsyncClient client =  
    DynamoDbAsyncClient.builder()  
        .httpClientBuilder(httpClientBuilder)  
        .overrideConfiguration(overrideConfig.build())  
        .asyncConfiguration(asyncConfig.build())  
        .build();
```

For a complete mapping of client configuration methods between 1.x and 2.x, see the AWS SDK for Java 2.x [changelog](#).

Setter Methods

In the AWS SDK for Java 2.x, setter method names don't include the "set" or "with" prefix. For example, `*.withEndpoint()` is now just `*.endpoint()`.

Example of using setting methods in 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

Example of using setting methods in 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

Class Names

All client class names are now fully camel cased and no longer prefixed by "Amazon". These changes are aligned with names used in the AWS CLI. For a full list of client name changes, see the AWS SDK for Java 2.x [changelog](#).

Example of class names in 1.x

```
AmazonDynamoDB
AWSACMPCAAsyncClient
```

Example of class names in 2.x

```
DynamoDbClient
AcmAsyncClient
```

Region Class

The AWS SDK for Java version 1.x had multiple `Region` and `Regions` classes, both in the core package and in many of the service packages. `Region` and `Regions` classes in version 2.x are now collapsed into one core class, `Region`.

Example Region and Regions classes in 1.x

```
com.amazonaws.regions.Region
com.amazonaws.regions.Regions
com.amazonaws.services.ec2.model.Region
```

Example Region class in 2.x

```
software.amazon.awssdk.regions.Region
```

For more details about changes related to using the `Region` class, see [Region class name changes \(p. 28\)](#).

Immutable POJOs

Clients and operation request and response objects are now immutable and cannot be changed after creation. To reuse a request or response variable, you must build a new object to assign to it.

Example of updating a request object in 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Example of updating a request object in 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

Streaming Operations

Streaming operations such as the Amazon S3 `getObject` and `putObject` methods now support non-blocking I/O. As a result, the request and response POJOs no longer take `InputStream` as a parameter. Instead the request object accepts `RequestBody`, which is a stream of bytes. The asynchronous client accepts `AsyncRequestBody`.

Example of Amazon S3 `putObject` operation in 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example of Amazon S3 `putObject` operation in 2.x

```
s3client.putObject(PutObjectRequest.builder()
    .bucket(BUCKET)
    .key(KEY)
    .build(),
    RequestBody.of(Paths.get("myfile.in")));
```

In parallel, the response object accepts `ResponseTransformer` for synchronous clients and `AsyncResponseTransformer` for asynchronous clients.

Example of Amazon S3 `getObject` operation in 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

Example of Amazon S3 `getObject` operation in 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

Exception changes

Exception class names, and their structures and relationships, have also changed. `software.amazon.awssdk.core.exception.SdkException` is the new base `Exception` class that all the other exceptions extend.

For a full list of the 2.x exception class names mapped to the 1.x exceptions, see [Exception class name changes \(p. 29\)](#).

Service-Specific Changes

Amazon S3 Operation Name Changes

Many of the operation names for the Amazon S3 client have changed in the AWS SDK for Java 2.x. In version 1.x, the Amazon S3 client is not generated directly from the service API. This results in inconsistency between the SDK operations and the service API. In version 2.x, we now generate the Amazon S3 client to be more consistent with the service API.

Example of Amazon S3 client operation in 1.x

```
changeObjectStorageClass
```

Example of Amazon S3 client operation in 2.x

```
copyObject
```

Example of Amazon S3 client operation in the Amazon S3 service API

```
CopyObject
```

For a full list of the operation name mappings, see the AWS SDK for Java 2.x [changelog](#).

Cross-region access

For security best practices, cross-region access is no longer supported for single clients.

In version 1.x, services such as Amazon S3, Amazon SNS, and Amazon SQS allowed access to resources across Region boundaries. This is no longer allowed in version 2.x using the same client. If you need to access a resource in a different region, you must create a client in that region and retrieve the resource using the appropriate client.

Additional client changes

This topic describes additional changes to the default client in the AWS SDK for Java 2.x.

Default client changes

- The default credential provider chain for Amazon S3 no longer includes anonymous credentials. You must specify anonymous access to Amazon S3 manually by using the `AnonymousCredentialsProvider`.
- The following environment variables related to default client creation have been changed.

| 1.x | 2.x |
|--------------------------------|---------------------------|
| <code>AWS_CBOR_DISABLED</code> | <code>CBOR_ENABLED</code> |

| 1.x | 2.x |
|-------------------------------------|---------------------------------|
| <code>AWS_ION_BINARY_DISABLE</code> | <code>BINARY_ION_ENABLED</code> |

- The following system properties related to default client creation have been changed.

| 1.x | 2.x |
|---|---|
| <code>com.amazonaws.sdk.disableEc2Metadata</code> | <code>aws.disableEc2Metadata</code> |
| <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code> | <code>aws.ec2MetadataServiceEndpoint</code> |
| <code>com.amazonaws.sdk.disableCbor</code> | <code>aws.cborEnabled</code> |
| <code>com.amazonaws.sdk.disableIonBinary</code> | <code>aws.binaryIonEnabled</code> |

- The following system properties are no longer supported in 2.x.

| 1.x |
|---|
| <code>com.amazonaws.sdk.disableCertChecking</code> |
| <code>com.amazonaws.sdk.enableDefaultMetrics</code> |
| <code>com.amazonaws.sdk.enableThrottledRetry</code> |
| <code>com.amazonaws.regions.RegionUtils.fileOverride</code> |
| <code>com.amazonaws.regions.RegionUtils.disableRemote</code> |
| <code>com.amazonaws.services.s3.disableImplicitGlobalClients</code> |
| <code>com.amazonaws.sdk.enableInRegionOptimizedMode</code> |

- Loading Region configuration from a custom endpoints.json file is no longer supported.

Credentials provider changes

Credentials provider

This section provides a mapping of the name changes of credential provider classes and methods between versions 1.x and 2.x of the AWS SDK for Java. The following also lists some of the key differences in the way credentials are processed by the SDK in version 2.x:

- The default credentials provider loads system properties before environment variables in version 2.x. See [Using credentials \(p. 34\)](#) for more information.
- The constructor method is replaced with the `create` or `builder` methods.

Example

```
DefaultCredentialsProvider.create();
```

- Asynchronous refresh is no longer set by default. You must specify it with the `builder` of the credentials provider.

Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- You can specify a path to a custom profile file using the `ProfileCredentialsProvider.builder()`.

Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- Profile file format has changed to more closely match the AWS CLI. See [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide* for details.

Credentials provider changes mapped between versions 1.x and 2.x

Method name changes

| 1.x | 2.x |
|---|--|
| <code>AWSCredentialsProvider.getCredentials</code> | <code>AwsCredentialsProvider.resolveCredentials</code> |
| <code>DefaultAWSCredentialsProviderChain.getInstance</code> | Not Supported |
| <code>AWSCredentialsProvider.getInstance</code> | Not Supported |
| <code>AWSCredentialsProvider.refresh</code> | Not Supported |

Environment variable name changes

| 1.x | 2.x |
|---|--|
| <code>AWS_ACCESS_KEY</code> | <code>AWS_ACCESS_KEY_ID</code> |
| <code>AWS_SECRET_KEY</code> | <code>AWS_SECRET_ACCESS_KEY</code> |
| <code>AWS_CREDENTIAL_PROFILES_FILE</code> | <code>AWS_SHARED_CREDENTIALS_FILE</code> |

System property name changes

| 1.x | 2.x |
|---|---|
| <code>aws.secretKey</code> | <code>aws.secretAccessKey</code> |
| <code>com.amazonaws.sdk.disableEc2Metadata</code> | <code>aws.disableEc2Metadata</code> |
| <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code> | <code>aws.ec2MetadataServiceEndpoint</code> |

Region class name changes

This section describes the changes implemented in the AWS SDK for Java 2.x for using the `Region` and `Regions` classes.

Region configuration

- Some AWS services don't have Region specific endpoints. When using those services, you must set the Region as `Region.AWS_GLOBAL` or `Region.AWS_CN_GLOBAL`.

Example

```
Region region = Region.AWS_GLOBAL;
```

- `com.amazonaws.regions.Regions` and `com.amazonaws.regions.Region` classes are now combined into one class, `software.amazon.awssdk.regions.Region`.

Method and Class Name Mappings

The following tables map Region related classes between versions 1.x and 2.x of the AWS SDK for Java. You can create an instance of these classes using the `of()` method.

Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

Regions class method changes

| 1.x | 2.x |
|---------------------------------------|------------------------|
| <code>Regions.fromName</code> | <code>Region.of</code> |
| <code>Regions.getName</code> | <code>Region.id</code> |
| <code>Regions.getDescription</code> | Not Supported |
| <code>Regions.getCurrentRegion</code> | Not Supported |
| <code>Regions.DEFAULT_REGION</code> | Not Supported |
| <code>Regions.name</code> | Not Supported |

Region class method changes

| 1.x | 2.x |
|---|------------------------|
| <code>Region.getName</code> | <code>Region.id</code> |
| <code>Region.hasHttpsEndpoint</code> | Not Supported |
| <code>Region.hasHttpEndpoint</code> | Not Supported |
| <code>Region.getAvailableEndpoints</code> | Not Supported |
| <code>Region.createClient</code> | Not Supported |

RegionMetadata class method changes

| 1.x | 2.x |
|-----------------------------|--------------------------|
| RegionMetadata.getName | RegionMetadata.name |
| RegionMetadata.getDomain | RegionMetadata.domain |
| RegionMetadata.getPartition | RegionMetadata.partition |

ServiceMetadata class method changes

| 1.x | 2.x |
|---------------------------|--|
| Region.getServiceEndpoint | ServiceMetadata.endpointFor(Region) |
| Region.isServiceSupported | ServiceMetadata.regions().contains(Region) |

Exception class name changes

This topic contains a mapping of exception class-related name changes between versions 1.x and 2.x.

This table maps the exception class name changes.

| 1.x | 2.x |
|---|--|
| com.amazonaws.SdkBaseException com.amazonaws.AmazonClientException | software.amazon.awssdk.core.exception.SdkException |
| com.amazonaws.SdkClientException | software.amazon.awssdk.core.exception.SdkClientException |
| com.amazonaws.AmazonServiceException | software.amazon.awssdk.awscore.exception.AwsServiceException |

The following table maps the methods on exception classes between version 1.x and 2.x.

| 1.x | 2.x |
|--|---|
| AmazonServiceException.getRequestId | SdkServiceException.requestId |
| AmazonServiceException.getServiceName | AwsServiceException.awsErrorDetails().serviceName |
| AmazonServiceException.getErrorCode | AwsServiceException.awsErrorDetails().errorCode |
| AmazonServiceException.getErrorMessage | AwsServiceException.awsErrorDetails().errorMessage |
| AmazonServiceException.getStatusCode | AwsServiceException.awsErrorDetails().sdkHttpResponse |
| AmazonServiceException.getHttpHeaders | AwsServiceException.awsErrorDetails().sdkHttpResponse |
| AmazonServiceException.rawResponse | AwsServiceException.awsErrorDetails().rawResponse |

Using the SDK for Java 1.x and 2.x side-by-side

You can use both versions of the AWS SDK for Java in your projects.

The following shows an example of the pom.xml file for a project that uses Amazon S3 from version 1.x and DynamoDB from version 2.16.1.

Example Example of POM

This example shows a pom.xml file entry for a project that uses both 1.x and 2.x versions of the SDK.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

Using the AWS SDK for Java 2.x

After completing the steps in [Setting up the SDK \(p. 9\)](#), you are ready to make requests to AWS services such as Amazon S3, DynamoDB, IAM, Amazon EC2, and more.

Creating service clients

To make a request to an AWS service, you must first instantiate an object to serve as a client for that service using the static factory method `builder`. Then customize it by using the setters in the builder. The fluent setter methods return the `builder` object, so that you can chain the method calls for convenience and for more readable code. After you configure the properties you want, you can call the `build` method to create the client.

As an example, this code snippet instantiates an `Ec2Client` object as a service client for Amazon EC2:

```
Region region = Region.US_WEST_2;
Ec2Client ec2Client = Ec2Client.builder()
    .region(region)
    .build();
```

Note

Service clients in the SDK are thread-safe. For best performance, treat them as long-lived objects. Each client has its own connection pool resource that is released when the client is garbage collected.

A service client object is immutable, so you must create a new client for each service to which you make requests, or if you want to use a different configuration for making requests to the same service.

Specifying the `Region` in the service client builder is not required for all AWS services; however, it is a best practice to set the `Region` for the API calls you make in your applications. See [AWS region selection \(p. 38\)](#) for more information.

Using the default client

The client builders have another factory method named `create`. This method creates a service client with the default configuration. It uses the default provider chain to load credentials and the AWS Region. If credentials or the region can't be determined from the environment that the application is running in, the call to `create` fails. See [Using credentials \(p. 34\)](#) and [Region selection \(p. 38\)](#) for more information about how credentials and region are determined.

As an example, this code snippet instantiates a `DynamoDbClient` object as a service client for Amazon DynamoDB:

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
```

Making requests

You use the service client to make requests to that AWS service.

For example, this code snippet shows how to create a `RunInstancesRequest` object to create a new Amazon EC2 instance:

```
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()
    .imageId(amiId)
    .instanceType(InstanceType.T1_MICRO)
    .maxCount(1)
    .minCount(1)
    .build();

ec2Client.runInstances(runInstancesRequest);
```

Handling responses

You use a response handler to process the response back from the AWS service.

For example, this code snippet shows how to create a `RunInstancesResponse` object to handle the response from Amazon EC2 by printing out the `instanceId` for the new instance from the request above:

```
RunInstancesResponse runInstancesResponse = ec2Client.runInstances(runInstancesRequest);
System.out.println(runInstancesResponse.instances().get(0).instanceId());
```

Closing the client

When you no longer need the service client, close it.

```
ec2Client.close();
```

Note

Service clients extend the `AutoClosable` interface, but as a best practice - especially with short-lived code such as AWS Lambda functions - you should explicitly call the `close()` method.

Handling exceptions

The SDK uses runtime (or unchecked) exceptions, providing you fine-grained control over error handling and ensuring that exception handling will scale with your application.

An `SdkServiceException`, or one of its sub-classes, is the most common form of exception the SDK will throw. These exceptions represent responses from the AWS service. You can also handle an `SdkClientException`, which occurs when there's a problem on the client side (i.e., in your development or application environment), such as a network connection failure.

This code snippet demonstrates one way to handle service exceptions while uploading a file to Amazon S3.

```
Region region = Region.US_WEST_2;
s3Client = S3Client.builder()
    .region(region)
    .build();

try {

    PutObjectRequest putObjectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
```

```
        .key(key)
        .build();

s3Client.putObject(putObjectRequest, RequestBody.fromString("SDK for Java test"));
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

See [Handling exceptions \(p. 45\)](#) for more information.

Using waiters

Some requests take time to process, such as creating a new table in DynamoDB or creating a new Amazon S3 bucket. To ensure the resource is ready before your code continues to run, use a *Waiter*.

For example, this code snippet creates a new table ("myTable") in DynamoDB, waits for the table to be in an ACTIVE status, and then prints out the response:

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
DynamoDbWaiter dynamoDbWaiter = dynamoDbClient.waiter();

WaiterResponse<DescribeTableResponse> waiterResponse =
    dynamoDbWaiter.waitUntilTableExists(r -> r.tableName("myTable"));

waiterResponse.matched().response().ifPresent(System.out::println);
```

See [Using waiters \(p. 69\)](#) for more information.

Configuring service clients

To customize the configuration of a service client, use the setters on the factory method builder. For convenience and to create more readable code, you chain the methods to set multiple configuration options.

As an example, refer to the following code snippet.

```
ClientOverrideConfiguration clientOverrideConfiguration =
    ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(1))
        .retryPolicy(RetryPolicy.builder().numRetries(10).build())
        .addMetricPublisher(CloudWatchMetricPublisher.create())
        .build();

Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
    .region(region)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .overrideConfiguration(clientOverrideConfiguration)
    .httpClientBuilder(ApacheHttpClient.builder())

    .proxyConfiguration(proxyConfig.build(ProxyConfiguration.builder()))
    .build()
    .build();
```

HTTP clients

You can change the default configuration for HTTP clients in applications you build with the AWS SDK for Java. For information on how to configure HTTP clients and settings, see [HTTP configuration \(p. 41\)](#).

Retries

You can change the default settings for retries in your service clients, including the retry mode and back-off strategy. For more information, refer to [the `RetryPolicy` class](#) in the AWS SDK for Java API Reference.

For more information about retries in AWS services, see [Error retries and exponential backoff in AWS](#).

Timeouts

You can configure timeouts for each of your service clients using the `apiCallTimeout` and the `apiCallAttemptTimeout` setters. The `apiCallTimeout` setting is the amount of time to allow the client to complete the execution of an API call. The `apiCallAttemptTimeout` setting is the amount of time to wait for the HTTP request to complete before giving up.

For more information, see [apiCallTimeout](#) and [apiCallAttemptTimeout](#) in the AWS SDK for Java API Reference.

Execution interceptors

You can write code that intercepts the execution of your API requests and responses at different parts of the request/response lifecycle. This enables you to publish metrics, modify a request in-flight, debug request processing, view exceptions, and more. For more information, see [the `ExecutionInterceptor` interface](#) in the AWS SDK for Java API Reference.

Additional information

- For complete examples of the code snippets above, see [Working with Amazon DynamoDB \(p. 88\)](#), [Working with Amazon EC2 \(p. 105\)](#), and [Working with Amazon S3 \(p. 75\)](#).

Using credentials

To make requests to Amazon Web Services using the AWS SDK for Java, you must use cryptographically-signed credentials issued by AWS. You can use programmatic access keys or temporary security credentials such as AWS IAM Identity Center (successor to AWS Single Sign-On) or IAM roles to grant access to AWS resources.

For information on setting up credentials, see [Set default credentials and Region \(p. 10\)](#) and [Set up credentials profiles \(p. 19\)](#).

Topics

- [Use the default credential provider chain \(p. 35\)](#)
- [Use a specific credentials provider or provider chain \(p. 36\)](#)
- [Use credentials profiles \(p. 36\)](#)
- [Supply credentials explicitly \(p. 37\)](#)
- [Configuring IAM roles for Amazon EC2 \(p. 37\)](#)

Use the default credential provider chain

After you [Set default credentials and Region \(p. 10\)](#) for your environment, the AWS SDK for Java will automatically use those credentials when your application makes requests to AWS. The default credential provider chain, implemented by the [DefaultCredentialsProvider](#) class, checks sequentially each of places where you can set default credentials and selects the first one you set.

To use the default credential provider chain to supply credentials in your application, create a service client builder without specifying credentials provider configuration.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
```

Credential retrieval order

The default credential provider chain of the AWS SDK for Java 2.x searches for credentials in your environment using a predefined sequence.

1. Java system properties

- The SDK uses the [SystemPropertyCredentialsProvider](#) class to load credentials from the `aws.accessKeyId` and `aws.secretAccessKey` Java system properties. If `aws.sessionToken` is also specified, the SDK will use temporary credentials.

Note

For information on how to set Java system properties, see the [System Properties](#) tutorial on the official *Java Tutorials* website.

2. Environment variables

- The SDK uses the [EnvironmentVariableCredentialsProvider](#) class to load credentials from the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` system environment variables. If `AWS_SESSION_TOKEN` is also specified, the SDK will use temporary credentials.

3. Web identity token from AWS STS

- The SDK uses the [WebIdentityTokenFileCredentialsProvider](#) class to load credentials from Java system properties or environment variables.

4. The shared `credentials` and `config` files

- The SDK uses the [ProfileCredentialsProvider](#) to load credentials from the `[default]` credentials profile in the shared `credentials` and `config` files.

Note

The `credentials` and `config` files are shared by various AWS SDKs and Tools. For more information, see [The `.aws/credentials` and `.aws/config` files](#) in the [AWS SDKs and Tools Reference Guide](#).

5. Amazon ECS container credentials

- The SDK uses the [ContainerCredentialsProvider](#) class to load credentials from the `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` system environment variable.

6. Amazon EC2 instance profile credentials

- The SDK uses the [InstanceProfileCredentialsProvider](#) class to load credentials from the Amazon EC2 metadata service.

Use a specific credentials provider or provider chain

Alternatively, you can specify which credentials provider the SDK should use. For example, if you set your default credentials using environment variables, supply an [EnvironmentVariableCredentialsProvider](#) object to the `credentialsProvider` method on the service client builder, as in the following code snippet.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .build();
```

For a complete list of credential providers and provider chains, see [All Known Implementing Classes in AwsCredentialsProvider](#).

Note

You can use your own credential provider or provider chains by implementing the `AwsCredentialsProvider` interface.

Use credentials profiles

Using the shared `credentials` file, you can set up custom profiles which enables you to use multiple sets of credentials in your application. The `[default]` profile was mentioned above. The SDK uses the [ProfileCredentialsProvider](#) class to load credentials from profiles defined in the shared `credentials` file.

For information on how to set up custom profiles, see [Set up credentials profiles \(p. 19\)](#).

This code snippet demonstrates how to build a service client that uses the credentials defined as part of the `profile_name` profile.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile_name"))
    .build();
```

Set a custom profile as the default

To set a profile other than the `[default]` profile as the default for your application, set the `AWS_PROFILE` environment variable to the name of your custom profile.

To set this variable on Linux, macOS, or Unix, use `export`:

```
export AWS_PROFILE="other_profile"
```

To set these variables on Windows, use `set`:

```
set AWS_PROFILE="other_profile"
```

Alternatively, set the `aws.profile` Java system property to the name of the profile.

Supply credentials explicitly

If the default credential chain or a specific or custom provider or provider chain doesn't work for your application, you can supply the credentials that you want directly in code. These can be AWS account credentials, IAM credentials, or temporary credentials retrieved from AWS Security Token Service (AWS STS). If you've retrieved temporary credentials using AWS STS, use this method to specify the credentials for AWS access.

Important

For security, use *IAM account credentials* instead of the AWS account credentials when accessing AWS. For more information, see [AWS Security Credentials](#) in the Amazon Web Services General Reference.

1. Instantiate a class that provides the [AwsCredentials](#) interface, such as [AwsSessionCredentials](#). Supply it with the AWS access key and secret key to use for the connection.
2. Create a [StaticCredentialsProvider](#) object and supply it with the `AwsCredentials` object.
3. Configure the service client builder with the `StaticCredentialsProvider` and build the client.

The following example creates a new service client using credentials that you supply:

```
AwsBasicCredentials awsCreds = AwsBasicCredentials.create(
    "your_access_key_id",
    "your_secret_access_key");

S3Client s3 = S3Client.builder()
    .credentialsProvider(StaticCredentialsProvider.create(awsCreds))
    .build();
```

Configuring IAM roles for Amazon EC2

All requests to AWS services must be cryptographically signed using credentials issued by AWS. You can use *IAM roles* to conveniently grant secure access to AWS resources from your Amazon EC2 instances.

This topic provides information about how to use IAM roles with AWS SDK for Java applications running on Amazon EC2. For more information about IAM instances, see [IAM Roles for Amazon EC2](#) in the Amazon EC2 User Guide for Linux Instances.

Default provider chain and Amazon EC2 instance profiles

If your application creates an AWS client using the `create` method, the client searches for credentials using the *default credentials provider chain*, in the following order:

1. In the Java system properties: `aws.accessKeyId` and `aws.secretAccessKey`.
2. In system environment variables: `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
3. In the default credentials file (the location of this file varies by platform).
4. In the Amazon ECS environment variable: `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`.
5. In the *instance profile credentials*, which exist within the instance metadata associated with the IAM role for the Amazon EC2 instance.

The final step in the default provider chain is available only when running your application on an Amazon EC2 instance. However, it provides the greatest ease of use and best security when working with Amazon EC2 instances. You can also pass an [InstanceProfileCredentialsProvider](#) instance directly to

the client constructor to get instance profile credentials without proceeding through the entire default provider chain.

For example:

```
S3Client s3 = S3Client.builder()  
    .credentialsProvider(InstanceProfileCredentialsProvider.builder().build())  
    .build();
```

When you use this approach, the SDK retrieves temporary AWS credentials that have the same permissions as those associated with the IAM role that is associated with the Amazon EC2 instance in its instance profile. Although these credentials are temporary and would eventually expire, `InstanceProfileCredentialsProvider` periodically refreshes them for you so that the obtained credentials continue to allow access to AWS.

Walkthrough: Using IAM roles for EC2 instances

This walkthrough shows you how to retrieve an object from Amazon S3 using an IAM role to manage access.

Create an IAM role

Create an IAM role that grants read-only access to Amazon S3.

1. Open the [IAM console](#).
2. In the navigation pane, choose **Roles**, then **Create New Role**.
3. On the **Select Role Type** page, under **AWS service Roles**, choose **Amazon EC2**.
4. On the **Attach Policy** page, choose **Amazon S3 Read Only Access** from the policy list, then choose **Next Step**.
 - Enter a name for the role, then select **Next Step**. Remember this name
 - because you'll need it when you launch your Amazon EC2 instance.
5. On the **Review** page, choose **Create Role**.

Launch an EC2 instance and specify your IAM role

You can launch an Amazon EC2 instance with an IAM role using the Amazon EC2 console.

To launch an Amazon EC2 instance using the console, follow the directions in [Getting Started with Amazon EC2 Linux Instances](#) in the Amazon EC2 User Guide for Linux Instances.

When you reach the **Review Instance Launch** page, select **Edit instance details**. In **IAM role**, choose the IAM role that you created previously. Complete the procedure as directed.

Note

You need to create or use an existing security group and key pair to connect to the instance.

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

AWS region selection

Regions enable you to access AWS services that physically reside in a specific geographic area. This can be useful both for redundancy and to keep your data and applications running close to where you and your users will access them.

In AWS SDK for Java 2.x, all the different region related classes from version 1.x have been collapsed into one `Region` class. You can use this class for all region-related actions such as retrieving metadata about a region or checking whether a service is available in a region.

Choosing a region

You can specify a region name and the SDK will automatically choose an appropriate endpoint for you.

To explicitly set a region, we recommend that you use the constants defined in the `Region` class. This is an enumeration of all publicly available regions. To create a client with a region from the class, use the following code.

```
Ec2Client ec2 = Ec2Client.builder()
    .region(Region.US_WEST_2)
    .build();
```

If the region you are attempting to use isn't one of the constants in the `Region` class, you can create a new region using the `of` method. This feature allows you access to new Regions without upgrading the SDK.

```
Region newRegion = Region.of("us-east-42");
Ec2Client ec2 = Ec2Client.builder()
    .region(newRegion)
    .build();
```

Note

After you build a client with the builder, it's *immutable* and the region *cannot be changed*. If you are working with multiple AWS Regions for the same service, you should create multiple clients—one per region.

Choosing a specific endpoint

Each AWS client can be configured to use a *specific endpoint* within a region by calling the `endpointOverride` method.

For example, to configure the Amazon EC2 client to use the Europe (Ireland) Region, use the following code.

```
Ec2Client ec2 = Ec2Client.builder()
    .region(Region.EU_WEST_1)
    .endpointOverride(URI.create("https://ec2.eu-west-1.amazonaws.com"))
    .build();
```

See [Regions and Endpoints](#) for the current list of regions and their corresponding endpoints for all AWS services.

Automatically determine the Region from the environment

When running on Amazon EC2 or AWS Lambda, you might want to configure clients to use the same region that your code is running on. This decouples your code from the environment it's running in and makes it easier to deploy your application to multiple regions for lower latency or redundancy.

To use the default credential/region provider chain to determine the region from the environment, use the client builder's `create` method.

```
Ec2Client ec2 = Ec2Client.create();
```

If you don't explicitly set a region using the `region` method, the SDK consults the default region provider chain to try and determine the region to use.

Default region provider chain

The following is the region lookup process:

1. Any explicit region set by using `region` on the builder itself takes precedence over anything else.
2. The `AWS_REGION` environment variable is checked. If it's set, that region is used to configure the client.

Note

This environment variable is set by the Lambda container.

3. The SDK checks the AWS shared configuration file (usually located at `~/.aws/config`). If the `region` property is present, the SDK uses it.
 - The `AWS_CONFIG_FILE` environment variable can be used to customize the location of the shared config file.
 - The `AWS_PROFILE` environment variable or the `aws.profile` system property can be used to customize the profile that the SDK loads.
4. The SDK attempts to use the Amazon EC2 instance metadata service to determine the region of the currently running Amazon EC2 instance.
5. If the SDK still hasn't found a region by this point, client creation fails with an exception.

When developing AWS applications, a common approach is to use the *shared configuration file* (described in [Credential retrieval order \(p. 35\)](#)) to set the region for local development, and rely on the default region provider chain to determine the region when running on AWS infrastructure. This greatly simplifies client creation and keeps your application portable.

Checking for service availability in a Region

To see if a particular AWS service is available in a region, use the `serviceMetadata` and `region` method on the service that you'd like to check.

```
DynamoDbClient.serviceMetadata().regions().forEach(System.out::println);
```

See the [Region](#) class documentation for the regions you can specify, and use the endpoint prefix of the service to query.

Optimizing cold start performance for AWS Lambda

Among the improvements in the AWS SDK for Java 2.x is the SDK cold startup time for Java functions in Lambda. This is the time it takes for a Java Lambda function to start up and respond to its first request.

Version 2.x includes three primary changes that contribute to this improvement:

- Use of [jackson-jr](#), which is a serialization library that improves initialization time.
- Use of the [java.time](#) libraries for date and time objects.

- Use of [Slf4j](#) for a logging facade.

You can gain additional SDK startup time improvement by setting specific configuration values on the client builder. They each save some time at startup by reducing the amount of information your application needs to find for initialization.

In your client builder, specify a region, use Environment Variable credentials provider, and specify `URLConnectionClient` as the `httpClient`. See the code snippet below for an example.

- The [region lookup process](#) for the SDK takes time. By specifying a region, you can save up to 80ms of initialization time.

Note

By specifying an AWS region, the code will not run in other regions without modification.

- The process the SDK uses to look for credentials can take up to 90ms. By using the [EnvironmentVariableCredentialsProvider](#)

Note

Using this credentials provider enables the code to be used in Lambda functions, but may not work on Amazon EC2 or other systems.

- Instantiation time for JDK's [URLConnection](#) library is much lower than Apache HTTP Client or Netty. You can save up to 1 second by using this HTTP client.

Note

The JDK's [URLConnection](#), which is used by the [URLConnectionHttpClient](#) in the SDK for Java 2.x, does not support the HTTP PATCH method. PATCH is used by a few API operations such as [BatchUpdateFindings](#) in the Security Hub API ([BatchUpdateFindingsRequest](#) in the SDK for Java 2.x). If the AWS Lambda requires API methods that ultimately use HTTP's PATCH, you will need to use one of the other [available HTTP clients](#).

Example client configuration

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .httpClient(URLConnectionHttpClient.builder().build())
    .build();
```

HTTP configuration

You can change the default configuration for HTTP clients in applications you build with the AWS SDK for Java. This section discusses HTTP clients and settings for the AWS SDK for Java 2.x.

For asynchronous clients, you can use the [NettyNioAsyncHttpClient](#) or [AwsCrtAsyncHttpClient](#). For more information, see [Configuring the Netty-based HTTP client \(p. 44\)](#) or [Configuring the AWS CRT-based HTTP client \(p. 42\)](#).

For synchronous clients, you can use [URLConnectionHttpClient](#) or [ApacheHttpClient](#). For more information about Apache HttpClient, see [HttpClient Overview](#).

For a full list of options you can set with these clients, see the [AWS SDK for Java 2.x API Reference](#).

Topics

- [Configuring the AWS CRT-based HTTP client \(p. 42\)](#)

- [Configuring the Netty-based HTTP client \(p. 44\)](#)

Configuring the AWS CRT-based HTTP client

The AWS Common Runtime (CRT) HTTP client is a new HTTP client you can use with the AWS SDK for Java 2.x. The CRT-based HTTP client is an asynchronous, non-blocking HTTP client built on top of the Java bindings of the [AWS Common Runtime](#). You can use the CRT-based HTTP client to benefit from features such as improved performance, connection health checks, and post-quantum TLS support.

For asynchronous operations in the AWS SDK for Java 2.x, you can use Netty ([NettyNioAsyncHttpClient](#)) as the HTTP client or you can use the new AWS Common Runtime (CRT) HTTP client [AwsCrtAsyncHttpClient](#). This topics shows you how to configure the AWS CRT-based HTTP client.

Prerequisites

Before you can use use the AWS CRT client, you need to configure your project dependencies in your `pom.xml` or `build.gradle` file to do the following:

- Use version 2.14.13 or later of the AWS SDK for Java.
- Include version 2.14.13-PREVIEW of the `artifactId` `aws-crt-client`.

The following code example shows how to configure your project dependencies.

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.13</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
      <version>2.14.13-PREVIEW</version>
    </dependency>
  </dependencies>
</project>
```

Using the CRT-based HTTP client

You can use the CRT-based HTTP client for a specific service client, or you can create a single HTTP client to share across multiple service clients. These options are recommended for most use cases. Alternatively, you can set the CRT-based client as the default HTTP client for all asynchronous service clients and requests in your application.

The following code example shows how to use the CRT-based HTTP client for a specific service client.

```
S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient.builder()
        .maxConcurrency(50))
    .build();
```

The following code example shows how to use the CRT-based HTTP client as a shared HTTP client.

```
SdkAsyncHttpClient crtClient = AwsCrtAsyncHttpClient.create()
S3AsyncClient.builder()
    .httpClient(crtClient)
    .build();
```

Note

Your application must manage the lifecycle of an HTTP client instantiated outside of a service client builder. (A `builder` is static factory method used by the AWS SDK for Java to connect to Amazon Web Services such as Amazon S3 and Amazon Kinesis. For more information, see [Creating service clients \(p. 31\)](#).)

Setting the CRT-based HTTP client as the default

For asynchronous operations in the AWS SDK for Java 2.x, you can use Netty ([NettyNioAsyncHttpClient](#)) or the new AWS CRT-based HTTP client ([AwsCrtAsyncHttpClient](#)) as the default asynchronous HTTP client in the AWS SDK for Java 2.x.

Instead of using Netty as the asynchronous HTTP client, you can set the CRT-based HTTP client to be the default for your application. You can set this in your project's dependencies (for example, Maven `pom.xml` file) by explicitly excluding Netty. Alternatively, you can set the default HTTP client via Java system property when you run your app or in your application code.

Remove Netty from the project dependencies

Refer to the following snippet of a Maven `pom.xml` file.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <version>2.14.13</version>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
      <version>2.14.13-PREVIEW</version>
    </dependency>
  </dependencies>
</project>
```

Setting via Java system property

To use the CRT-based HTTP client as the default HTTP for your application, you can set the Java system property `software.amazon.awssdk.http.async.service.impl` to a value of `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService`.

To set during application startup, run a command similar to the following.

```
java app.jar -Dsoftware.amazon.awssdk.http.async.service.impl=\
software.amazon.awssdk.http.crt.AwsCrtSdkHttpService
```


Use the following code snippet to set in your application code.

```
System.setProperty("software.amazon.awssdk.http.async.service.impl",  
"software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient");
```

Configuring the CRT-based HTTP client

With the CRT-based HTTP client with in the AWS SDK for Java, you can configure various settings including connection health checks and maximum idle time. You can also configure post-quantum TLS support when you make requests to AWS Key Management Service (Amazon Kinesis).

Connection health checks

You can configure connection health checks for the CRT-based HTTP client using the `connectionHealthChecks` method on the HTTP client builder. Refer to the following example code snippet and the [API documentation](#).

```
AwsCrtAsyncHttpClient.builder()  
    .connectionHealthChecksConfiguration(  
        b -> b.minThroughputInBytesPerSecond(32000L)  
            .allowableThroughputFailureInterval(Duration.ofSeconds(3)))  
    .build();
```

Post-quantum TLS support

You can configure the CRT-based HTTP client to use post-quantum TLS when your application makes requests to Amazon Kinesis. Use the `tlsCipherPreference` method on the HTTP client builder. Refer to the following example code snippet and the [API documentation](#).

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()  
    .tlsCipherPreference(TlsCipherPreference.TLS_CIPHER_KMS_PQ_TLSv1_0_2019_06)  
    .build();  
KinesisAsyncClient kac = KinesisAsyncClient.builder()  
    .httpClient(awsCrtHttpClient)  
    .build();
```

Configuring the Netty-based HTTP client

For asynchronous operations in the AWS SDK for Java 2.x, you can use Netty ([NettyNioAsyncHttpClient](#)) as the HTTP client or you can use the new AWS Common Runtime (CRT) HTTP client [AwsCrtAsyncHttpClient](#). This topics shows you how to configure the Netty-based HTTP client.

For a full list of options you can set with these clients, see the [AWS SDK for Java API Reference version 2.x](#).

Prerequisite

Before you can use the Netty client, you need to configure your project dependencies in your `pom.xml` or `build.gradle` file to include version 2.0.0 or later of the `artifactId`netty-nio-client`.

The following code example shows how to configure your project dependencies.

```
<dependency>
```

```
<artifactId>netty-nio-client</artifactId>
<groupId>software.amazon.awssdk</groupId>
<version>2.0.0</version>
</dependency>
```

Configuring service clients

Use the HTTP client builder to have the SDK manage its lifecycle. The HTTP client will be closed for you when the service client is shut down.

Imports

```
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.services.kinesis.AmazonKinesisAsyncClient;
```

Code

```
AmazonKinesisAsyncClient client = AmazonKinesisAsyncClient.builder()
    .httpClientBuilder(NettyNioAsyncHttpClient.builder()
        .maxConcurrency(100)
        .maxPendingConnectionAcquires(10_000))
    .build();
```

You can also pass the HTTP client directly to the service client if you want to manage the lifecycle yourself.

Code

```
SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
    .maxConcurrency(100)
    .maxPendingConnectionAcquires(10_000)
    .build();

AmazonKinesisAsyncClient kinesisClient = AmazonKinesisAsyncClient.builder()
    .httpClient(httpClient)
    .build();

httpClient.close();
```

Exception handling for the AWS SDK for Java

Understanding how and when the AWS SDK for Java throws exceptions is important to building high-quality applications using the SDK. The following sections describe the different cases of exceptions that are thrown by the SDK and how to handle them appropriately.

Why unchecked exceptions?

The AWS SDK for Java uses runtime (or unchecked) exceptions instead of checked exceptions for these reasons:

- To allow developers fine-grained control over the errors they want to handle without forcing them to handle exceptional cases they aren't concerned about (and making their code overly verbose)
- To prevent scalability issues inherent with checked exceptions in large applications

In general, checked exceptions work well on small scales, but can become troublesome as applications grow and become more complex.

SdkServiceException (and subclasses)

`SdkServiceException` is the most common exception that you'll experience when using the AWS SDK for Java. This exception represents an error response from an AWS service. For example, if you try to terminate an Amazon EC2 instance that doesn't exist, Amazon EC2 will return an error response and all the details of that error response will be included in the `SdkServiceException` that's thrown. For some cases, a subclass of `SdkServiceException` is thrown to allow developers fine-grained control over handling error cases through catch blocks.

When you encounter an `SdkServiceException`, you know that your request was successfully sent to the AWS service but couldn't be successfully processed. This can be because of errors in the request's parameters or because of issues on the service side.

`SdkServiceException` provides you with information such as:

- Returned HTTP status code
- Returned AWS error code
- Detailed error message from the service
- AWS request ID for the failed request

SdkClientException

`SdkClientException` indicates that a problem occurred inside the Java client code, either while trying to send a request to AWS or while trying to parse a response from AWS. An `SdkClientException` is generally more severe than an `SdkServiceException`, and indicates a major problem that is preventing the client from making service calls to AWS services. For example, the AWS SDK for Java throws an `SdkClientException` if no network connection is available when you try to call an operation on one of the clients.

Logging AWS SDK for Java calls

The AWS SDK for Java is instrumented with `Slf4j`, which is an abstraction layer that enables the use of any one of several logging systems at runtime.

Supported logging systems include the Java Logging Framework and Apache Log4j, among others. This topic shows you how to use Log4j. You can use the SDK's logging functionality without making any changes to your application code.

To learn more about `Log4j`, see the [Apache website](#).

Add the Log4J JAR

To use Log4j with the SDK, you need to download the Log4j JAR from the [Log4j website](#) or use Maven by adding a dependency on Log4j in your pom.xml file. The SDK doesn't include the JAR.

Log4j configuration file

Log4j uses a configuration file, `log4j2.xml`. Example configuration files are shown below. To learn more about the values used in the configuration file, see the [manual for Log4j configuration](#).

Place your configuration file in a directory on your classpath. The Log4j JAR and the log4j2.xml file do not have to be in the same directory.

The log4j2.xml configuration file specifies properties such as [logging level](#), where logging output is sent (for example, [to a file or to the console](#)), and the [format of the output](#). The logging level is the granularity of output that the logger generates. Log4j supports the concept of multiple logging *hierarchies*. The logging level is set independently for each hierarchy. The following two logging hierarchies are available in the AWS SDK for Java:

- software.amazon.awssdk
- org.apache.http.wire

Setting the classpath

Both the Log4j JAR and the log4j2.xml file must be located on your classpath. To configure the log4j binding for SL4j in Maven you can add the following to your pom.xml:

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j-impl</artifactId>
</dependency>
```

If you're using the Eclipse IDE, you can set the classpath by opening the menu and navigating to **Project | Properties | Java Build Path**.

Service-specific errors and warnings

We recommend that you always leave the "software.amazon.awssdk" logger hierarchy set to "WARN" to catch any important messages from the client libraries. For example, if the Amazon S3 client detects that your application hasn't properly closed an `InputStream` and could be leaking resources, the S3 client reports it through a warning message to the logs. This also ensures that messages are logged if the client has any problems handling requests or responses.

The following log4j2.xml file sets the `rootLogger` to WARN, which causes warning and error messages from all loggers in the "software.amazon.awssdk" hierarchy to be included. Alternatively, you can explicitly set the software.amazon.awssdk logger to WARN.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
  </Loggers>
```

```
</Configuration>
```

Request/response summary logging

Every request to an AWS service generates a unique AWS request ID that is useful if you run into an issue with how an AWS service is handling a request. AWS request IDs are accessible programmatically through Exception objects in the SDK for any failed service call, and can also be reported through the DEBUG log level in the "software.amazon.awssdk.request" logger.

The following log4j2.xml file enables a summary of requests and responses.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender" />
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
    <Logger name="software.amazon.awssdk.request" level="DEBUG" />
  </Loggers>
</Configuration>
```

Here is an example of the log output:

```
2018-01-28 19:31:56 [main] DEBUG software.amazon.awssdk.request:Logger.java:78 - Sending
Request: software.amazon.awssdk.http.DefaultSdkHttpFullRequest@3a80515c
```

Verbose wire logging

In some cases, it can be useful to see the exact requests and responses that the AWS SDK for Java sends and receives. If you really need access to this information, you can temporarily enable it through the Apache HttpClient logger. Enabling the DEBUG level on the `apache.http.wire` logger enables logging for all request and response data.

Warning

We recommend you only use wire logging for debugging purposes. Disable it in your production environments because it can log sensitive data. It logs the full request or response without encryption, even for an HTTPS call. For large requests (e.g., to upload a file to Amazon S3) or responses, verbose wire logging can also significantly impact your application's performance.

The following log4j2.xml file turns on full wire logging in Apache HttpClient.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender" />
    </Root>
  </Loggers>
</Configuration>
```

```
<Logger name="software.amazon.awssdk" level="WARN" />
<Logger name="software.amazon.awssdk.request" level="DEBUG" />
<Logger name="org.apache.http.wire" level="DEBUG" />
</Loggers>
</Configuration>
```

Additional Maven dependency on log4j-1.2-api is required for wire-logging with Apache as it uses 1.2 under the hood. Add the following to the pom.xml file if you enable wire logging.

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-1.2-api</artifactId>
</dependency>
```

Setting the JVM TTL for DNS name lookups

The Java virtual machine (JVM) caches DNS name lookups. When the JVM resolves a hostname to an IP address, it caches the IP address for a specified period of time, known as the *time-to-live* (TTL).

Because AWS resources use DNS name entries that occasionally change, we recommend that you configure your JVM with a TTL value of no more than 60 seconds. This ensures that when a resource's IP address changes, your application will be able to receive and use the resource's new IP address by requerying the DNS.

On some Java configurations, the JVM default TTL is set so that it will *never* refresh DNS entries until the JVM is restarted. Thus, if the IP address for an AWS resource changes while your application is still running, it won't be able to use that resource until you *manually restart* the JVM and the cached IP information is refreshed. In this case, it's crucial to set the JVM's TTL so that it will periodically refresh its cached IP information.

Note

The default TTL can vary according to the version of your JVM and whether a [security manager](#) is installed. Many JVMs provide a default TTL less than 60 seconds. If you're using such a JVM and not using a security manager, you can ignore the remainder of this topic.

How to set the JVM TTL

To modify the JVM's TTL, set the `networkaddress.cache.ttl` property value. Use one of the following methods, depending on your needs:

- **globally, for all applications that use the JVM.** Set `networkaddress.cache.ttl` in the `$JAVA_HOME/jre/lib/security/java.security` file:

```
networkaddress.cache.ttl=60
```

- **for your application only,** set `networkaddress.cache.ttl` in your application's initialization code:

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

Features of the AWS SDK for Java 2.x

This section provides information about the features of the AWS SDK for Java 2.x.

Topics

- [Asynchronous programming \(p. 50\)](#)
- [Using the DynamoDB Enhanced Client in the AWS SDK for Java 2.x \(p. 56\)](#)
- [Working with HTTP/2 in the AWS SDK for Java \(p. 56\)](#)
- [Enabling SDK metrics for the AWS SDK for Java \(p. 56\)](#)
- [Retrieving paginated results using the AWS SDK for Java 2.x \(p. 61\)](#)
- [Amazon S3 Transfer Manager \(Preview\) \(p. 67\)](#)
- [Using waiters in the AWS SDK for Java 2.x \(p. 69\)](#)

Asynchronous programming

The AWS SDK for Java 2.x features truly nonblocking asynchronous clients that implement high concurrency across a few threads. The AWS SDK for Java 1.x has asynchronous clients that are wrappers around a thread pool and blocking synchronous clients that don't provide the full benefit of nonblocking I/O.

Synchronous methods block your thread's execution until the client receives a response from the service. Asynchronous methods return immediately, giving control back to the calling thread without waiting for a response.

Because an asynchronous method returns before a response is available, you need a way to get the response when it's ready. The methods for asynchronous client in 2.x of the AWS SDK for Java return *CompletableFuture* objects that allow you to access the response when it's ready.

Non-streaming operations

For non-streaming operations, asynchronous method calls are similar to synchronous methods. However, the asynchronous methods in the AWS SDK for Java return a *CompletableFuture* object that contains the results of the asynchronous operation *in the future*.

Call the *CompletableFuture* `whenComplete()` method with an action to complete when the result is available. *CompletableFuture* implements the *Future* interface, so you can also get the response object by calling the `get()` method.

The following is an example of an asynchronous operation that calls a Amazon DynamoDB function to get a list of tables, receiving a *CompletableFuture* that can hold a *ListTablesResponse* object. The action defined in the call to `whenComplete()` is done only when the asynchronous call is complete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
```

```
import java.util.concurrent.CompletableFuture;
```

Code

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
            client.listTables(ListTablesRequest.builder()
                .build());

        // Map the response to another CompletableFuture containing just the table names
        CompletableFuture<List<String>> tableNames =
            response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Lets the application shut down. Only close the client when you are
                // completely done with it.
                client.close();
            }
        });
        tableNames.join();
    }
}
```

The following code example shows you how to retrieve an Item from a table by using the Asynchronous client. Invoke the `getItem` method of the `DynamoDbAsyncClient` and pass it a [GetItemRequest](#) object with the table name and primary key value of the item you want. This is typically how you pass data that the operation requires. In this example, notice that a String value is passed.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```


Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String key,
String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
            client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
            returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Streaming operations

For streaming operations, you must provide an [AsyncRequestBody](#) to provide the content incrementally, or an [AsyncResponseTransformer](#) to receive and process the response.

The following example uploads a file to Amazon S3 asynchronously by using the `PutObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development environment,
 * including your AWS credentials.
 */
```

```
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example, bucket1).
\n\n" +
            "    key - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        // Put the object into the bucket
        CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
            AsyncRequestBody.fromFile(Paths.get(path))
        );
        future.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object uploaded. Details: " + resp);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Only close the client when you are completely done with it
                client.close();
            }
        });

        future.join();
    }
}
```

The following example gets a file from Amazon S3 asynchronously by using the `GetObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development environment,
 * including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example, bucket1).
\n\n" +
            "    objectKey - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        CompletableFuture<GetObjectResponse> futureGet = client.getObject(objectRequest,
            AsyncResponseTransformer.toFile(Paths.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                } else {
                    err.printStackTrace();
                }
            }
            finally {
                // Only close the client when you are completely done with it
                client.close();
            }
        })
    }
}
```

```

    });
    futureGet.join();
  }
}

```

Advanced operations

The AWS SDK for Java 2.x uses [Netty](#), an asynchronous event-driven network application framework, to handle I/O threads. The AWS SDK for Java 2.x creates an `ExecutorService` behind Netty, to complete the futures returned from the HTTP client request through to the Netty client. This abstraction reduces the risk of an application breaking the async process if developers choose to stop or sleep threads. By default, 50 Threads are generated for each asynchronous client, and managed in a queue within the `ExecutorService`.

Advanced users can specify their thread pool size when creating an asynchronous client using the following option when building.

Code

```

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10)
        )
    )
    .build();

```

To optimize performance, you can manage your own thread pool executor, and include it when configuring your client.

```

ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(10_000),
    new ThreadFactoryBuilder()
        .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            executor
        )
    )
    .build();

```

If you prefer to not use a thread pool, at all, use `Runnable::run` instead of using a thread pool executor.

```

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Runnable::run
        )
    )
    .build();

```

Using the DynamoDB Enhanced Client in the AWS SDK for Java 2.x

The Amazon DynamoDB enhanced client is a high-level library that is part of the AWS SDK for Java version 2 (v2). It offers a straightforward way to map client-side classes to DynamoDB tables. You define the relationships between tables and their corresponding model classes in your code. Then you can intuitively perform various create, read, update, or delete (CRUD) operations on tables or items in DynamoDB.

The AWS SDK for Java v2 includes a set of annotations that you can use with a Java bean to quickly generate a [TableSchema](#) for mapping your classes to tables. Alternatively, if you declare each [TableSchema](#) explicitly, you don't need to include annotations in your classes.

For information about how to use the DynamoDB Enhanced Client, refer to [Mapping items in DynamoDB tables \(p. 99\)](#).

Working with HTTP/2 in the AWS SDK for Java

HTTP/2 is a major revision of the HTTP protocol. This new version has several enhancements to improve performance:

- Binary data encoding provides more efficient data transfer.
- Header compression reduces the overhead bytes downloaded by the client, helping get the content to the client sooner. This is especially useful for mobile clients that are already constrained on bandwidth.
- Bidirectional asynchronous communication (multiplexing) allows multiple requests and response messages between the client and AWS to be in flight at the same time over a single connection, instead of over multiple connections, which improves performance.

Developers upgrading to the latest SDKs will automatically use HTTP/2 when it's supported by the service they're working with. New programming interfaces seamlessly take advantage of HTTP/2 features and provide new ways to build applications.

The AWS SDK for Java 2.x features new APIs for event streaming that implement the HTTP/2 protocol. For examples of how to use these new APIs, see [Working with Kinesis \(p. 151\)](#).

Enabling SDK metrics for the AWS SDK for Java

With the AWS SDK for Java 2.x, you can collect metrics about the service clients in your application, analyze the output in Amazon CloudWatch, and then act on it.

By default, metrics collection is disabled in the SDK. This topic helps you to enable and configure it.

Prerequisites

Before you can enable and use metrics, you must complete the following steps:

- Complete the steps in [Setting up \(p. 9\)](#).
- Configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version 2.14.0 or later of the AWS SDK for Java.

To enabling publishing of metrics to CloudWatch, also include the artifactId `cloudwatch-metric-publisher` with the version number `2.14.0` or later in your project's dependencies.

For example:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
    </dependency>
  </dependencies>
</project>
```

Note

To enhance the security of your application, you can use dedicated set of credentials for publishing metrics to CloudWatch. Create a separate IAM user with `cloudwatch:PutMetricData` permissions and then use that user's access key as credentials in the `MetricPublisher` configuration for your application.

For more information, see the [Amazon CloudWatch Permissions Reference](#) in the [Amazon CloudWatch Events User Guide](#) and [Adding and Removing IAM Identity Permissions](#) in the [IAM User Guide](#).

How to enable metrics collection

You can enable metrics in your application for a service client or on individual requests.

Enable metrics for a specific request

The following code snippet shows how to enable the CloudWatch metrics publisher for a request to Amazon DynamoDB. It uses the default metrics publisher configuration.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build());
```

Enable metrics for a specific service client

The following code snippet shows how to enable the CloudWatch metrics publisher for a service client.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
```

```
DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

The following snippet demonstrates how to use a custom configuration for the metrics publisher for a specific service client. The customizations include loading a separate credentials profile, specifying a different region than the service client, and customizing how often the publisher sends metrics to CloudWatch.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create("cloudwatch"))
    .region(Region.US_WEST_2)
    .publishFrequency(5, TimeUnit.MINUTES)
    .build();

Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

What information is collected?

Metrics collection includes the following:

- Number of API requests, including whether they succeed or fail
- Information about the AWS services you call in your API requests, including exceptions returned
- The duration for various operations such as Marshalling, Signing, and HTTP requests
- HTTP client metrics, such as the number of open connections, the number of pending requests, and the name of the HTTP client used

Note

The metrics available vary by HTTP client.

For a complete list, see [Service client metrics \(p. 58\)](#).

How can I use this information?

You can use the metrics the SDK collects to monitor the service clients in your application. You can look at overall usage trends, identify anomalies, review service client exceptions returned, or to dig in to understand a particular issue. Using Amazon CloudWatch, you can also create alarms to notify you as soon as your application reaches a condition that you define.

For more information, see [Using Amazon CloudWatch Metrics](#) and [Using Amazon CloudWatch Alarms](#) in the [Amazon CloudWatch User Guide](#).

Service client metrics

With the AWS SDK for Java version 2 (v2), you can collect metrics about the service clients in your application and then publish (output) those metrics to CloudWatch.

This topic contains the list and descriptions for the metrics that are collected.

For more information about enabling and configuring metrics for the SDK, see [Enabling SDK metrics \(p. 56\)](#).

Metrics collected with each request

| Metric name | Description | Type | Collected by default? |
|--------------------------|---|----------|-----------------------|
| ApiCallDuration | The total time taken to finish a request (inclusive of all retries) | Duration | Yes |
| ApiCallSuccessful | True if the API call was successful; false if not | Boolean | Yes |
| CredentialsFetchDuration | The time taken to fetch signing credentials for the request | Duration | Yes |
| MarshallingDuration | The time taken to marshall the request | Duration | Yes |
| OperationName | The name of the AWS API the request is made to | String | Yes |
| RetryCount | Number of times the SDK retried the API call | Integer | Yes |
| ServiceId | Service ID of the AWS service that the API request is made against | String | Yes |

Metrics collected for each request attempt

Each API call that your application makes may take multiple attempts before responded with a success or failure. These metrics are collected for each attempt.

| Metric name | Description | Type | Collected by default? |
|----------------------|--|----------|-----------------------|
| AvailableConcurrency | The number of remaining concurrent requests that can be supported by the HTTP client without needing to establish another connection | Integer | Yes |
| AwsExtendedRequestId | The extended request ID of the service request | String | Yes |
| AwsRequestId | The request ID of the service request | String | Yes |
| BackoffDelayDuration | The duration of time the SDK waited before this API call attempt | Duration | Yes |

| Metric name | Description | Type | Collected by default? |
|-------------------------------|--|----------|-----------------------|
| ConcurrencyAcquireDuration | The time taken to acquire a channel from the connection pool | Duration | Yes |
| HttpClientName | The name of the HTTP being use for the request | String | Yes |
| HttpStatusCode | The status code returned with the HTTP response | Integer | Yes |
| LeasedConcurrency | The number of request currently being executed by the HTTP client | Integer | Yes |
| LocalStreamWindowSize | The local HTTP/2 window size in bytes for the stream that this request was executed on | Integer | Yes |
| MarshallingDuration | The time it takes to marshal an SDK request to an HTTP request | Duration | Yes |
| MaxConcurrency | The max number of concurrent requests supported by the HTTP client | Integer | Yes |
| PendingConcurrencyAcquireTime | The number of requests that are blocked, waiting for another TCP connection or a new stream to be available from the connection pool | Integer | Yes |
| RemoteStreamWindowSize | The remote HTTP/2 window size in bytes for the stream that this request was executed on | Integer | Yes |
| ServiceCallDuration | The time it takes to connect to the service, send the request, and receive the HTTP status code and header from the response | Duration | Yes |
| SigningDuration | The time it takes to sign the HTTP request | Duration | Yes |

| Metric name | Description | Type | Collected by default? |
|-----------------------|---|----------|-----------------------|
| UnmarshallingDuration | The time it takes to unmarshall an HTTP response to an SDK response | Duration | Yes |

Retrieving paginated results using the AWS SDK for Java 2.x

Many AWS operations return paginated results when the response object is too large to return in a single response. In the AWS SDK for Java 1.0, the response contained a token you had to use to retrieve the next page of results. New in the AWS SDK for Java 2.x are autopagination methods that make multiple service calls to get the next page of results for you automatically. You only have to write code that processes the results. Additionally both types of methods have synchronous and asynchronous versions. See [examples-asynchronous](#) for more detail about asynchronous clients.

The following examples use Amazon S3 and Amazon DynamoDB operations to demonstrate the various methods of retrieving your data from paginated responses.

Note

These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in [setup-credentials](#).

Synchronous pagination

These examples use the synchronous pagination methods for listing objects in an Amazon S3 bucket.

Iterate over pages

Build a [ListObjectsV2Request](#) and provide a bucket name. Optionally you can provide the maximum number of keys to retrieve at one time. Pass it to the [S3Client's listObjectsV2Paginator](#) method. This method returns a [ListObjectsV2Iterable](#) object, which is an [Iterable](#) of the [ListObjectsV2Response](#) class.

The first example demonstrates using the paginator object to iterate through all the response pages with the [stream](#) method. You can directly stream over the response pages, convert the response stream to a stream of [S3Object](#) content, and then process the content of the Amazon S3 object.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
ListObjectsV2Request listReq = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
// Process response pages
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));
```

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

Iterate over objects

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response.

Use a stream

Use the `stream` method on the response content to iterate over the paginated item collection.

Code

```
// Helper method to work with paginated collection of items directly
listRes.contents().stream()
    .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));
```

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

Use a for loop

Use a standard `for` loop to iterate through the contents of the response.

Code

```
for (S3Object content : listRes.contents()) {
    System.out.println(" Key: " + content.key() + " size = " + content.size());
}
```

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

Manual pagination

If your use case requires it, manual pagination is still available. Use the next token in the response object for the subsequent requests. Here's an example using a `while` loop.

Code

```
ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

boolean done = false;
while (!done) {
    ListObjectsV2Response listObjResponse = s3.listObjectsV2(listObjectsReqManual);
    for (S3Object content : listObjResponse.contents()) {
        System.out.println(content.key());
    }

    if (listObjResponse.nextContinuationToken() == null) {
        done = true;
    }

    listObjectsReqManual = listObjectsReqManual.toBuilder()
        .continuationToken(listObjResponse.nextContinuationToken())
        .build();
}
```

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

Asynchronous pagination

These examples use the asynchronous pagination methods for listing tables in DynamoDB. A manual pagination example is available in the [basics-async](#) topic.

Iterate over pages of table names

First, create an asynchronous DynamoDB client. Then, call the `listTablesPaginator` method to get a [ListTablesPublisher](#). This is an implementation of the reactive streams Publisher interface. To learn more about the reactive streams model, see the [Reactive Streams Github repo](#).

Call the `subscribe` method on the [ListTablesPublisher](#) and pass a subscriber implementation. In this example, the subscriber has an `onNext` method that requests one item at a time from the publisher. This is the method that is called repeatedly until all pages are retrieved. The `onSubscribe` method calls the `Subscription.request` method to initiate requests for data from the publisher. This method must be called to start getting data from the publisher. The `onError` method is triggered if an error occurs while retrieving data. Finally, the `onComplete` method is called when all pages have been requested.

Use a subscriber

Imports

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
```

```
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;
import io.reactivex.Flowable;
import reactor.core.publisher.Flux;
```

Code

First create an async client

```
    // Creates a default client with credentials and regions loaded from the
environment
    final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

    ListTablesRequest listTablesRequest = ListTablesRequest.builder().limit(3).build();
```

Then use Subscriber to get results.

```
    // Or subscribe method should be called to create a new Subscription.
    // A Subscription represents a one-to-one life-cycle of a Subscriber subscribing to
a Publisher.
    publisher.subscribe(new Subscriber<ListTablesResponse>() {
        // Maintain a reference to the subscription object, which is required to
request data from the publisher
        private Subscription subscription;

        @Override
        public void onSubscribe(Subscription s) {
            subscription = s;
            // Request method should be called to demand data. Here we request a single
page
            subscription.request(1);
        }

        @Override
        public void onNext(ListTablesResponse response) {
            response.tableNames().forEach(System.out::println);
            // Once you process the current page, call the request method to signal
that you are ready for next page
            subscription.request(1);
        }

        @Override
        public void onError(Throwable t) {
            // Called when an error has occurred while processing the requests
        }

        @Override
        public void onComplete() {
            // This indicates all the results are delivered and there are no more pages
left
        }
    });
```

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

Use a for loop

Use a for loop to iterate through the pages for simple use cases when creating a new subscriber might be too much overhead. The response publisher object has a `forEach` helper method for this purpose.

Code

```
ListTablesPublisher publisher = asyncClient.listTablesPaginator(listTablesRequest);

// Use a for-loop for simple use cases
CompletableFuture<Void> future = publisher.subscribe(response ->
response.tableNames()

.forEach(System.out::println));
```

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

Iterate over table names

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response. Similar to the synchronous result, the asynchronous result class has a method to interact with the underlying item collection. The return type of the convenience method is a publisher that can be used to request items across all pages.

Use a subscriber

Code

First create an async client

```
System.out.println("running AutoPagination - iterating on item collection...\n");

// Creates a default client with credentials and regions loaded from the
environment
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest = ListTablesRequest.builder().limit(3).build();
```

Then use Subscriber to get results.

```
// Use subscriber
publisher.subscribe(new Subscriber<String>() {
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
        subscription = s;
        subscription.request(1);
    }

    @Override
    public void onNext(String tableName) {
        System.out.println(tableName);
        subscription.request(1);
    }

    @Override
    public void onError(Throwable t) { }

    @Override
    public void onComplete() { }
```

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

Use a for loop

Use the `forEach` convenience method to iterate through the results.

Code

```
// Use forEach
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

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

Use third-party library

You can use other third party libraries instead of implementing a custom subscriber. This example demonstrates using the RxJava implementation but any library that implements the reactive stream interfaces can be used. See the [RxJava wiki page on Github](#) for more information on that library.

To use the library, add it as a dependency. If using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.2.21</version>
</dependency>
```

Imports

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;

import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;
import io.reactivex.Flowable;
import reactor.core.publisher.Flux;
```

Code

```
System.out.println("running AutoPagination - using third party subscriber...\n");

DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()

.build());

// The Flowable class has many helper methods that work with any reactive streams
compatible publisher implementation
List<String> tables = Flowable.fromPublisher(publisher)
.flatMapIterable(ListTablesResponse::tableNames)
```

```
                .toList()  
                .blockingGet();  
System.out.println(tables);
```

Amazon S3 Transfer Manager (Preview)

The Amazon S3 Transfer Manager (Preview) is an open-source high level file transfer utility for the AWS SDK for Java 2.x that you can use to easily transfer files to and from Amazon Simple Storage Service (S3). It's built on top of the Java bindings of the [AWS Common Runtime S3 Client](#), benefiting from its enhanced throughput, performance, and reliability by leveraging Amazon S3 multipart upload and byte-range fetches for parallel transfers.

This topic helps you use the S3 Transfer Manager.

Prerequisites

Before you can use the Transfer Manager, you must do the following:

- Complete the steps in [Setting up \(p. 9\)](#).
- Configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version 2.17.16 or later of the SDK for Java.
- Include version 2.17.16-PREVIEW of the `artifactId` `s3-transfer-manager`.

The following code example shows how to configure your project dependencies.

```
<project>  
  <dependencyManagement>  
    <dependencies>  
      <dependency>  
        <groupId>software.amazon.awssdk</groupId>  
        <artifactId>bom</artifactId>  
        <version>2.17.16</version>  
        <type>pom</type>  
        <scope>import</scope>  
      </dependency>  
    </dependencies>  
  </dependencyManagement>  
  <dependencies>  
    <dependency>  
      <groupId>software.amazon.awssdk</groupId>  
      <artifactId>s3-transfer-manager</artifactId>  
      <version>2.17.16-PREVIEW</version>  
    </dependency>  
  </dependencies>  
</project>
```

Imports

To make use of the code snippets in this topic, include the following imports:

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.transfer.s3.S3ClientConfiguration;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.CompletedDownload;  
import software.amazon.awssdk.transfer.s3.CompletedUpload;
```



```
import software.amazon.awssdk.transfer.s3.Download;
import software.amazon.awssdk.transfer.s3.Upload;
import software.amazon.awssdk.transfer.s3.UploadRequest;
```

Using the Transfer Manager (Preview)

With the Preview of the Amazon S3 Transfer Manager, you can upload or download one file per request.

To upload or download a file, first instantiate an `S3TransferManager` object to use as a service client.

To instantiate a service client using the default settings, use the `create()` method of `S3TransferManager`.

```
S3TransferManager s3TransferManager = S3TransferManager.create();
```

To customize the configuration of the service client, such as to select a region or to use a specific credentials provider for the request, build an `S3ClientConfiguration` object and then specify that configuration with the `s3ClientConfiguration()` method on the service client builder.

```
Region region = Region.US_WEST_2;
S3ClientConfiguration s3ClientConfiguration =
    S3ClientConfiguration.builder()
        .region(region)
        .minimumPartSizeInBytes(10 * MB)
        .targetThroughputInGbps(20.0)
        .build();
```

Upload a file to S3

To upload a file to Amazon S3 using the Transfer Manager (Preview), first build a `PutObjectRequest`, specifying the Amazon S3 bucket and key to which you want to upload with the `bucket()` and `key()` methods. Next, instantiate an `UploadRequest` object, passing the `PutObjectRequest` object using the `putObjectRequest()` method. Set the path to the file via the `source()` method. Then build an `Upload` object, passing in the `UploadRequest` object via the `upload()` method.

With the Transfer Manager, you can complete all of the above steps using short-hand (Java lambda) notation, so that all you have to do is specify the path to the file you are uploading and the bucket and key to which you want to upload the file.

```
Upload upload =
    s3TransferManager.upload(b -> b.putObjectRequest(r -> r.bucket(bucket).key(key))
        .source(Paths.get("fileToUpload.txt")));
```

To capture the response, use a `CompletedUpload` object.

```
CompletedUpload completedUpload = upload.completionFuture().join();
System.out.println("PutObjectResponse: " + completedUpload.response());
```

Download a file from S3

To download a file from Amazon S3 using the Transfer Manager (Preview), build a `Download` object. Using short-hand notation, you can specify the Amazon S3 bucket and key using the `getObjectRequest()` method and use the `destination()` to set where the file will be saved.

```
Download download =  
    s3TransferManager.download(b -> b.getObjectRequest(r -> r.bucket(bucket).key(key))  
        .destination(Paths.get("downloadedFile.txt")));
```

To capture the response, use a [CompletedDownload](#) object.

```
CompletedDownload completedDownload = download.completionFuture().join();  
System.out.println("Content length: " + completedDownload.response().contentLength());
```

Using waiters in the AWS SDK for Java 2.x

The waiters utility of the AWS SDK for Java 2.x enables you to validate that AWS resources are in a specified state before performing operations on those resources.

A *waiter* is an abstraction used to poll AWS resources, such as DynamoDB tables or Amazon S3 buckets, until a desired state is reached (or until a determination is made that the resource won't ever reach the desired state). Instead of writing logic to continuously poll your AWS resources, which can be cumbersome and error-prone, you can use waiters to poll a resource and have your code continue to run after the resource is ready.

Prerequisites

Before you can use waiters in a project with the AWS SDK for Java, you must complete the steps in [Setting up the AWS SDK for Java 2.x \(p. 9\)](#).

You must also configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version 2.15.0 or later of the AWS SDK for Java.

For example:

```
<project>  
  <dependencyManagement>  
    <dependencies>  
      <dependency>  
        <groupId>software.amazon.awssdk</groupId>  
        <artifactId>bom</artifactId>  
        <version>2.15.0</version>  
        <type>pom</type>  
        <scope>import</scope>  
      </dependency>  
    </dependencies>  
  </dependencyManagement>  
</project>
```

Using waiters

To instantiate a waiters object, first create a service client. Set the service client's `waiter()` method as the value of the waiter object. Once the waiter instance exists, set its response options to execute the appropriate code.

Synchronous programming

The following code snippet shows how to wait for a DynamoDB table to exist and be in an **ACTIVE** state.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();

WaiterResponse<DescribeTableResponse> waiterResponse =
    waiter.waitForTableExists(r -> r.tableName("myTable"));

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

Asynchronous programming

The following code snippet shows how to wait for a DynamoDB table to no longer exist.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
    asyncWaiter.waitForTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
    if (t == null) {
        // print out the matched ResourceNotFoundException
        r.matched().exception().ifPresent(System.out::println);
    }
}).join();
```

Configuring waiters

You can customize the configuration for a waiter by using the `overrideConfiguration()` on its builder. For some operations, you can apply a custom configuration when you make the request.

Configure a waiter

The following code snippet shows how to override the configuration on a waiter.

```
// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();

// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();
```

Override configuration for a specific request

The following code snippet shows how to override the configuration for a waiter on a per-request basis. Note that only some operations have customizable configurations.

```
waiter.waitForTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));
```

```
asyncWaiter.waitForTableExists(b -> b.tableName("myTable"),  
    o -> o.waitForTimeout(Duration.ofMinutes(1)));
```

Code examples

For a complete example using waiters with DynamoDB, see [CreateTable.java](#) in the AWS Code Examples Repository.

For a complete example using waiters with Amazon S3, see [S3BucketOps.java](#) in the AWS Code Examples Repository.

Code examples for the AWS SDK for Java 2.x

This section provides programming examples you can use with the AWS SDK for Java 2.x for specific features, use cases, and AWS services.

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

- [Guided code examples for the AWS SDK for Java 2.x \(p. 72\)](#)
- [SDK for Java 2.x code examples \(p. 183\)](#)

Guided code examples for the AWS SDK for Java 2.x

This section provides guided code examples you can use to build applications that use AWS services.

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

- [AWS database services and AWS SDK for Java 2.x \(p. 73\)](#)
- [Working with Amazon S3 \(p. 75\)](#)
- [Working with DynamoDB \(p. 88\)](#)
- [Working with Amazon EC2 \(p. 105\)](#)
- [Working with IAM \(p. 119\)](#)
- [Working with Amazon Athena \(p. 136\)](#)
- [Working with CloudWatch \(p. 136\)](#)
- [Working with AWS CloudTrail \(p. 146\)](#)
- [Working with Amazon Cognito \(p. 146\)](#)
- [Working with Amazon Comprehend \(p. 151\)](#)
- [Working with Amazon EventBridge \(p. 151\)](#)
- [Working with Amazon Kinesis Data Firehose \(p. 151\)](#)
- [Working with Amazon Forecast \(p. 151\)](#)
- [Working with AWS Glue \(p. 151\)](#)

- [Working with Kinesis](#) (p. 151)
- [Working with AWS Key Management Service](#) (p. 158)
- [Invoke, list, and delete AWS Lambda functions](#) (p. 158)
- [Working with AWS Elemental MediaConvert](#) (p. 160)
- [Working with AWS Elemental MediaStore](#) (p. 161)
- [Working with AWS Migration Hub](#) (p. 161)
- [Working with Amazon Personalize](#) (p. 161)
- [Working with Amazon Pinpoint](#) (p. 161)
- [Working with Amazon Polly](#) (p. 168)
- [Working with Amazon Rekognition](#) (p. 169)
- [Working with Amazon SageMaker](#) (p. 169)
- [Working with AWS Secrets Manager](#) (p. 169)
- [Working with Amazon Simple Email Service](#) (p. 169)
- [Working with Amazon Simple Notification Service](#) (p. 169)
- [Working with Amazon Simple Queue Service](#) (p. 173)
- [Working with AWS Systems Manager](#) (p. 178)
- [Working with Amazon Simple Workflow Service](#) (p. 178)
- [Working with Amazon Textract](#) (p. 178)
- [Working with Amazon Transcribe](#) (p. 178)
- [Working with Amazon Translate](#) (p. 182)
- [Working with Amazon WorkDocs](#) (p. 182)

AWS database services and AWS SDK for Java 2.x

AWS offers several database types: relational, key-value, in-memory, document, and [several others](#). The SDK for Java 2.x support varies depending the nature of the database service in AWS.

Some database services, for example [Amazon DynamoDB](#) service, have web service APIs to manage the AWS resource (database) as well as web service APIs to interact with the data. In the SDK for Java 2.x these types of services have dedicated service clients, for example [DynamoDBClient](#).

Other database services have web service APIs that interact with the resource, such the [Amazon DocumentDB](#) API (for cluster, instance and resource management), but do not have a web service API for working with the data. The SDK for Java 2.x has a corresponding [DocDbClient](#) interface for working with the resource. However, you need another Java API, such as [MongoDB for Java](#) to work with the data.

Use the examples below to learn how you use the SDK for Java 2.x service clients with the different types of databases.

Amazon DynamoDB examples

Working with the data

SDK service client: [DynamoDBClient](#)

Example: [React/Spring REST application using DynamoDB](#)

Examples: [Several DynamoDB examples](#)

Working with the database

SDK service client: [DynamoDBClient](#)

Examples: [CreateTable](#), [ListTables](#), [DeleteTable](#)

| Working with the data | Working with the database |
|--|---------------------------|
| SDK service client: DynamoDBEnhancedClient | |
| Example: React/Spring REST application using DynamoDB | |
| Examples: Several DynamoDB examples (names starting with 'Enhanced') | |

See [additional DynamoDB examples \(p. 88\)](#) in the guided code examples section of this guide.

Amazon RDS examples

| Working with the data | Working with the database |
|--|--|
| Non-SDK API: JDBC, database-specific SQL flavor | SDK service client: RdsClient |
| Example: React/Spring REST application using MySQL | Examples: Several RdsClient examples |

Amazon Redshift examples

| Working with the data | Working with the database |
|---|---|
| SDK service client: RedshiftDataClient | SDK service client: RedshiftClient |
| Examples: Several RedshiftDataClient examples | Examples: Several RedshiftClient examples |
| Example: React/Spring REST application using RedshiftDataClient | |

Amazon Aurora Serverless v1 examples

| Working with the data | Working with the database |
|--|--|
| SDK service client: RdsDataClient | SDK service client: RdsClient |
| Example: React/Spring REST application using RdsDataClient | Examples: Several RdsClient examples |

Amazon DocumentDB examples

| Working with the data | Working with the database |
|--|---|
| Non-SDK API: MongoDB-specific Java library (for example MongoDB for Java) | SDK service client: DocDbClient |
| Examples: DocumentDB (Mongo) Developer Guide (select 'Java' tab) | |

Working with Amazon S3

This section provides examples of programming with [Amazon Simple Storage Service \(S3\)](#) using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Creating, listing, and deleting Amazon S3 buckets \(p. 75\)](#)
- [Working with Amazon S3 objects \(p. 79\)](#)
- [Working with Amazon S3 presigned URLs \(p. 85\)](#)
- [Working with S3 Glacier \(p. 88\)](#)

Creating, listing, and deleting Amazon S3 buckets

Every object (file) in Amazon S3 must reside within a *bucket*. A bucket represents a collection (container) of objects. Each bucket must have a unique *key* (name). For detailed information about buckets and their configuration, see [Working with Amazon S3 Buckets](#) in the Amazon Simple Storage Service User 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 User Guide.

Note

These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in [the section called "Set default credentials and Region" \(p. 10\)](#).

Topics

- [Create a bucket \(p. 75\)](#)
- [List the buckets \(p. 76\)](#)
- [Delete a bucket \(p. 77\)](#)

Create a bucket

Build a [CreateBucketRequest](#) and provide a bucket name. Pass it to the `S3Client`'s `createBucket` method. Use the `S3Client` to do additional operations such as listing or deleting buckets as shown in later examples.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
```



```
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an `S3Client`.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Make a Create Bucket Request.

```
// Create a bucket by using a S3Waiter object
public static void createBucket( S3Client s3Client, String bucketName) {

    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

List the buckets

Build a `ListBucketsRequest`. Use the `S3Client`'s `listBuckets` method to retrieve the list of buckets. If the request succeeds a `ListBucketsResponse` is returned. Use this response object to retrieve the list of buckets.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Make a List Buckets Request.

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x -> System.out.println(x.name()));
```

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

Delete a bucket

Before you can delete an Amazon S3 bucket, you must ensure that the bucket is empty or the service will return an error. If you have a [versioned bucket](#), you must also delete any versioned objects that are in the bucket.

Topics

- [Delete objects in a bucket \(p. 77\)](#)
- [Delete an empty bucket \(p. 78\)](#)

Delete objects in a bucket

Build a [ListObjectsV2Request](#) and use the S3Client's `listObjects` method to retrieve the list of objects in the bucket. Then use the `deleteObject` method on each object to delete it.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.*;
```

Code

First create an S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
```

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Delete all objects in the bucket.

```
public static void deleteObjectsInBucket (S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;
        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
    }
}
```

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

Delete an empty bucket

Build a [DeleteBucketRequest](#) with a bucket name and pass it to the S3Client's deleteBucket method.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Delete the bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

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

Working with Amazon S3 objects

An Amazon S3 object represents a file or collection of data. Every object must be contained in a [bucket](#) (p. 75).

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 User Guide.

Note

These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in [the section called "Set default credentials and Region"](#) (p. 10).

Topics

- [Upload an object](#) (p. 79)
- [Upload objects in multiple parts](#) (p. 80)
- [Download an object](#) (p. 82)
- [Delete an object](#) (p. 82)
- [Copy an object](#) (p. 83)
- [List objects](#) (p. 84)

Upload an object

Build a [PutObjectRequest](#) and supply a bucket name and key name. Then use the `S3Client`'s `putObject` method with a [RequestBody](#) that contains the object content and the `PutObjectRequest` object. *The bucket must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
Region region = Region.US_WEST_2;
s3 = S3Client.builder()
    .region(region)
    .build();

createBucket(s3, bucketName, region);

PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

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

Upload objects in multiple parts

Use the `S3Client`'s `createMultipartUpload` method to get an upload ID. Then use the `uploadPart` method to upload each part. Finally, use the `S3Client`'s `completeMultipartUpload` method to tell Amazon S3 to merge all the uploaded parts and finish the upload operation.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
```

```
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
        CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
        s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
        RequestBody.fromByteBuffer(getRandomByteBuffer(5 * MB))).eTag();

        CompletedPart part1 = CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
        UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
        RequestBody.fromByteBuffer(getRandomByteBuffer(3 * MB))).eTag();
        CompletedPart part2 = CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Finally call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
        CompletedMultipartUpload.builder()
            .parts(part1, part2)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
        CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
```

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

Download an object

Build a `GetObjectRequest` and supply a bucket name and key name. Use the `S3Client`'s `getObject` method, passing it the `GetObjectRequest` object and a `ResponseTransformer` object. The `ResponseTransformer` creates a response handler that writes the response content to the specified file or stream.

The following example specifies a file name to write the object content to.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.getObject(getObjectRequest);
```

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

Delete an object

Build a `DeleteObjectRequest` and supply a bucket name and key name. Use the `S3Client`'s `deleteObject` method, and pass it the name of a bucket and object to delete. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
```

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

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

Copy an object

Build a [CopyObjectRequest](#) and supply a bucket name that the object is copied into, a URL encoded string value (see the `URLEncoder.encode` method), and the key name of the object. Use the `S3Client`'s `copyObject` method, and pass the [CopyObjectRequest](#) object. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
```

Code

```
public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

    String encodedUrl = "";
```



```

    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());

    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }

    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySourceIfMatch(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

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

List objects

Build a [ListObjectsRequest](#) and supply the bucket name. Then invoke the `S3Client`'s `listObjects` method and pass the [ListObjectsRequest](#) object. This method returns a [ListObjectsResponse](#) that contains all of the objects in the bucket. You can invoke this object's `contents` method to get a list of objects. You can iterate through this list to display the objects, as shown in the following code example.

Imports

```

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

```

Code

```

public static void listBucketObjects(S3Client s3, String bucketName ) {

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calcKb(myValue.size()) + " KBs");
            System.out.print("\n The owner is " + myValue.owner());
        }
    }
}

```

```
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
//convert bytes to kbs.  
private static long calKb(Long val) {  
    return val/1024;  
}
```

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

Working with Amazon S3 presigned URLs

You can use a [S3Presigner](#) object to sign an Amazon S3 `SdkRequest` so that it's executed without requiring authentication on the part of the caller. For example, assume Alice has access to an S3 object, and she wants to temporarily share access to that object with Bob. Alice can generate a pre-signed [GetObjectRequest](#) object to secure share with Bob so that he can download the object without requiring access to Alice's credentials.

Topics

- [Generate a Presigned URL and Upload an Object \(p. 85\)](#)
- [Get a Presigned Object \(p. 86\)](#)

Generate a Presigned URL and Upload an Object

Build a [S3Presigner](#) object that represents the client object. Next create a [PresignedPutObjectRequest](#) object that can be executed at a later time without requiring additional signing or authentication. When you create this object, you can specify the bucket name and the key name. In addition, you can also specify the time in minutes that the bucket can be accessed without using credentials by invoking the `signatureDuration` method (as shown in the following code example).

You can use the [PresignedPutObjectRequest](#) object to obtain the URL by invoking its `url` method.

Imports

```
import java.io.IOException;  
import java.io.OutputStreamWriter;  
import java.net.HttpURLConnection;  
import java.net.URL;  
import java.time.Duration;  
import java.util.HashMap;  
import java.util.Map;  
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;  
import software.amazon.awssdk.services.s3.presigner.S3Presigner;  
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;
```

Code

The following Java code example uploads content to a presigned S3 bucket.

```
public static void signBucket(S3Presigner presigner, String bucketName, String keyName)  
{
```

```

try {
    Map<String, String> metadata = new HashMap<>();
    metadata.put("author", "Mary Doe");
    metadata.put("version", "1.0.0.0");

    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(keyName)
        .contentType("text/plain")
        .metadata(metadata)
        .build();

    PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10))
        .putObjectRequest(objectRequest)
        .build();

    PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
    System.out.println("Presigned URL to upload a file to: " +
presignedRequest.url());
    System.out.println("Which HTTP method needs to be used when uploading a file: "
+ presignedRequest.httpRequest().method());

    // Upload content to the Amazon S3 bucket by using this URL.
    URL url = presignedRequest.url();

    // Create the connection and use it to upload the new object.
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setDoOutput(true);
    connection.setRequestProperty("Content-Type", "text/plain");
    connection.setRequestProperty("x-amz-meta-author", "Mary Doe");
    connection.setRequestProperty("x-amz-meta-version", "1.0.0.0");
    connection.setRequestMethod("PUT");
    OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
    out.write("This text was uploaded as an object by using a presigned URL.");
    out.close();

    connection.getResponseCode();
    System.out.println("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}

```

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

Get a Presigned Object

Build a [S3Presigner](#) object that represents the client object. Next, create a [GetObjectRequest](#) object and specify the bucket name and key name. In addition, create a [GetObjectPresignRequest](#) object that can be executed at a later time without requiring additional signing or authentication. When you create this object, you can specify the time in minutes that the bucket can be accessed without using credentials by invoking the `signatureDuration` method (as shown in the following code example).

Invoke the `presignGetObject` method that belongs to the [S3Presigner](#) object to create a [PresignedPutObjectRequest](#) object. You can invoke this object's `url` method to obtain the URL to use. Once you have the URL, you can use standard HTTP Java logic to read the contents of the bucket, as shown in the following Java code example.

Imports

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;
```

Code

The following Java code example reads content from a presigned S3 bucket.

```
public static void getPresignedUrl(S3Presigner presigner, String bucketName, String
keyName ) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(60))
            .getObjectRequest(getObjectRequest)
            .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header, values) -
> {
            values.forEach(value -> {
                connection.setRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream = connection.getOutputStream()) {
                IoUtils.copy(signedPayload, httpOutputStream);
            }
        }

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            System.out.println("Service returned response: ");
            IoUtils.copy(content, System.out);
        }
    }
}
```

```
    } catch (S3Exception | IOException e) {  
        e.printStackTrace();  
    }  
}
```

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

Working with S3 Glacier

S3 Glacier is an extremely low-cost storage service that provides secure, durable, and flexible storage for data backup and archival. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with DynamoDB

This section provides examples that show you how to program [DynamoDB](#) by using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Working with tables in DynamoDB \(p. 88\)](#)
- [Working with items in DynamoDB \(p. 94\)](#)
- [Mapping items in DynamoDB tables \(p. 99\)](#)

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* are 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 as your needs change.

Create a table

Use the `DynamoDbClient`'s `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.

Note

If a table with the name you chose already exists, an [DynamoDbException](#) is thrown.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Create a table with a simple primary key

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

Code

```
public static String createTable(DynamoDbClient ddb, String tableName, String key) {

    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created
        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
    }
```

```
        newTable = response.tableDescription().tableName();
        return newTable;

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

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

Create a table with a composite primary key

Add another `AttributeDefinition` and `KeySchemaElement` to `CreateTableRequest`.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

Code

```
public static String createTableComKey(DynamoDbClient ddb, String tableName) {
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(
            AttributeDefinition.builder()
                .attributeName("Language")
                .attributeType(ScalarAttributeType.S)
                .build(),
            AttributeDefinition.builder()
                .attributeName("Greeting")
                .attributeType(ScalarAttributeType.S)
                .build()
        )
        .keySchema(
            KeySchemaElement.builder()
                .attributeName("Language")
                .keyType(KeyType.HASH)
                .build(),
            KeySchemaElement.builder()
                .attributeName("Greeting")
                .keyType(KeyType.RANGE)
                .build()
        )
        .provisionedThroughput(
            ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10)).build()
        )
        .tableName(tableName)
        .build();

    String tableId = "";

    try {
        CreateTableResponse result = ddb.createTable(request);
```

```
        tableId = result.tableDescription().tableId();
        return tableId;
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

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

List tables

You can list the tables in a particular region by calling the `DynamoDbClient`'s `listTables` method.

Note

If the named table doesn't exist for your account and region, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

Code

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();

            if (tableNames.size() > 0) {
                for (String curName : tableNames) {
                    System.out.format("* %s\n", curName);
                }
            } else {
                System.out.println("No tables found!");
                System.exit(0);
            }

            lastName = response.lastEvaluatedTableName();
            if (lastName == null) {
                moreTables = false;
            }
        } catch (DynamoDbException e) {
```



```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

By default, up to 100 tables are returned per call—use `lastEvaluatedTableName` on the returned [ListTablesResponse](#) 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](#) on GitHub.

Describe (get information about) a table

Call the `DynamoDbClient`'s `describeTable` method.

Note

If the named table doesn't exist for your account and region, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

Code

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo =
            ddb.describeTable(request).table();

        if (tableInfo != null) {
            System.out.format("Table name   : %s\n",
                tableInfo.tableName());
            System.out.format("Table ARN   : %s\n",
                tableInfo.tableArn());
            System.out.format("Status      : %s\n",
                tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
                tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
                throughputInfo.readCapacityUnits().longValue());
            System.out.format("  Write Capacity: %d\n",
```

```
        throughputInfo.writeCapacityUnits().longValue());

    List<AttributeDefinition> attributes =
        tableInfo.attributeDefinitions();
    System.out.println("Attributes");

    for (AttributeDefinition a : attributes) {
        System.out.format("  %s (%s)\n",
            a.attributeName(), a.attributeType());
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

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

Modify (update) a table

You can modify your table's provisioned throughput values at any time by calling the `DynamoDbClient`'s `updateTable` method.

Note

If the named table doesn't exist for your account and region, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void updateDynamoDBTable(DynamoDbClient ddb,
                                       String tableName,
                                       Long readCapacity,
                                       Long writeCapacity) {

    System.out.format(
        "Updating %s with new provisioned throughput values\n",
        tableName);
    System.out.format("Read capacity : %d\n", readCapacity);
    System.out.format("Write capacity : %d\n", writeCapacity);

    ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
        .readCapacityUnits(readCapacity)
        .writeCapacityUnits(writeCapacity)
        .build();

    UpdateTableRequest request = UpdateTableRequest.builder()
        .provisionedThroughput(tableThroughput)
        .tableName(tableName)
        .build();

    try {
        ddb.updateTable(request);
    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

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

Delete a table

Call the `DynamoDbClient`'s `deleteTable` method and pass it the table's name.

Note

If the named table doesn't exist for your account and region, a `ResourceNotFoundException` is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

Code

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

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

More information

- [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 (get) an item from a table

Call the `DynamoDbClient`'s `getItem` method and pass it a `GetItemRequest` object with the table name and primary key value of the item you want. It returns a `GetItemResponse` object with all of the

attributes for that item. You can specify one or more [projection expressions](#) in the `GetItemRequest` to retrieve specific attributes.

You can use the returned `GetItemResponse` object's `item()` method to retrieve a [Map](#) of key (String) and value ([AttributeValue](#)) pairs that are associated with the item.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

Code

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1, returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", key);
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Retrieve (get) an item from a table using the asynchronous client

Invoke the `getItem` method of the `DynamoDbAsyncClient` and pass it a [GetItemRequest](#) object with the table name and primary key value of the item you want.

You can return a [Collection](#) instance with all of the attributes for that item (refer to the following example).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String key,
String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
            client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
            returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Add a new item to a table

Create a [Map](#) of key-value pairs that represent the item's attributes. 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.

Note

If the named table doesn't exist for your account and region, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

Code

```
public static void putItemInTable(DynamoDbClient ddb,
                                String tableName,
                                String key,
                                String keyVal,
                                String albumTitle,
                                String albumTitleValue,
                                String awards,
                                String awardVal,
                                String songTitle,
                                String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle, AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        ddb.putItem(request);
        System.out.println(tableName + " was successfully updated");
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.\n",
            tableName);
        System.err.println("Be sure that it exists and that you've typed its name
            correctly!");
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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 a map of fields to update.

Note

If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

Code

```
public static void updateTableItem(DynamoDbClient ddb,
                                   String tableName,
                                   String key,
                                   String keyVal,
                                   String name,
                                   String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

    HashMap<String,AttributeValueUpdate> updatedValues =
        new HashMap<String,AttributeValueUpdate>();

    // Update the column specified by name with updatedVal
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

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

Delete an existing item in a table

You can delete an item that exists in a table by using the `DynamoDbClient`'s `deleteItem` method and providing a table name as well as the primary key value.

Note

If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a [ResourceNotFoundException](#) is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

Code

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String key,
String keyVal) {

    HashMap<String,AttributeValue> keyToGet =
        new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

More information

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

Mapping items in DynamoDB tables

The Amazon DynamoDB enhanced client is a high-level library that is part of the AWS SDK for Java version 2 (v2). It offers a straightforward way to map client-side classes to DynamoDB tables. You define the relationships between tables and their corresponding model classes in your code. Then you can intuitively perform various create, read, update, or delete (CRUD) operations on tables or items in DynamoDB.

The AWS SDK for Java v2 includes a set of annotations that you can use with a Java bean to quickly generate a [TableSchema](#) for mapping your classes to tables. Alternatively, if you declare each [TableSchema](#) explicitly, you don't need to include annotations in your classes.

To work with items in a DynamoDB table using the enhanced client, first create a [DynamoDbEnhancedClient](#) from an existing [DynamoDbClient](#) object.

```
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
```



```
        .dynamoDbClient(ddb)
        .build();
createDynamoDBTable(enhancedClient);
```

Create a table using the enhanced client

To easily create a [TableSchema](#) using the enhanced client, start by creating a Java data class that includes a default public constructor and standardized names of getters and setters for each property in the class. Include a class-level annotation to indicate it is a `DynamoDbBean` and, at a minimum, include a `DynamoDbPartitionKey` annotation on the getter or setter for the primary key of the table record.

Once this data class has been defined, call `TableSchema`'s `fromBean()` with that data class to create the table schema.

See the code snippet below for an example of how to do this.

Imports

```
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
```

Code

```
public static void putRecord(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));

        // Create an Instant
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        // Populate the Table
        Customer custRecord = new Customer();
        custRecord.setCustName("Susan red");
        custRecord.setId("id146");
        custRecord.setEmail("sred@noserver.com");
        custRecord.setRegistrationDate(instant);

        // Put the customer data into a DynamoDB table
        custTable.putItem(custRecord);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("done");
}
```

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

Retrieve (get) an item from a table

To get an item from a DynamoDB table, create a [DynamoDbTable](#) object and call `getItem()` with a [GetItemEnhancedRequest](#) object to get the actual item.

For example, the following code snippet demonstrates one way to use the enhanced client to get information from an item in a DynamoDB table.

Imports

```
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
```

Code

```
public static String getItem(DynamoDbEnhancedClient enhancedClient) {
    try {
        //Create a DynamoDbTable object
        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));

        //Create a KEY object
        Key key = Key.builder()
            .partitionValue("id146")
            .build();

        // Get the item by using the key
        Customer result = mappedTable.getItem(r->r.key(key));
        return "The email value is "+result.getEmail();

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return "";
}
```

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

Batch create (put) and delete items

You can batch a series of put requests ([PutItemEnhancedRequest](#)) and delete requests ([DeleteItemEnhancedRequest](#)) to one or more tables, and then send all of the changes in a single request.

In the following code snippet, a [DynamoDbTable](#) object is created, two items are queued up to be added to the table, and then the items are written to the table in a single call. Include multiple entries of `addDeleteItem()` and `addPutItem()` ([part of WriteBatch.Builder](#)) in each batch, as needed. To

queue up changes to a different table, add another instance of `WriteBatch.builder()` and provide a corresponding `DynamoDbTable` object in `mappedTableResource()`.

Imports

```
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
```

Code

```
public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
    try {

        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));

        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        Customer record2 = new Customer();
        record2.setCustName("Fred Pink");
        record2.setId("id110");
        record2.setEmail("fredp@noserver.com");
        record2.setRegistrationDate(instant) ;

        Customer record3 = new Customer();
        record3.setCustName("Susan Pink");
        record3.setId("id120");
        record3.setEmail("spink@noserver.com");
        record3.setRegistrationDate(instant) ;

        // Create a BatchWriteItemEnhancedRequest object
        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =
            BatchWriteItemEnhancedRequest.builder()
                .writeBatches(
                    WriteBatch.builder(Customer.class)
                        .mappedTableResource(mappedTable)
                        .addPutItem(r -> r.item(record2))
                        .addPutItem(r -> r.item(record3))
                        .build())
                .build();

        // Add these two items to the table
        enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
        System.out.println("done");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

```

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

Use a filtered query to get items from a table

You can get items from a table based on filterable queries, and then perform operations (for example, return item values) on one or more of the items in the query results.

In the following code snippet, you build a filter by first defining the value or values you're searching for as an [AttributeValue](#) object. Then you put this into a `HashMap` and build an [Expression](#) from the `HashMap`. Build a [QueryConditional](#) object to specify the primary key to match against in the query, and then execute the query on your [DynamoDbTable](#) object.

Note

The [QueryConditional](#) interface has several methods you can use to build your queries, including common conditional statements like greater than, less than, and in between.

Imports

```
import java.time.Instant;
import java.util.Map;
import java.util.Iterator;
import java.util.HashMap;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Expression;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.enhanced.dynamodb.model.QueryConditional;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void queryTableFilter(DynamoDbEnhancedClient enhancedClient) {
    try{
        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));

        AttributeValue att = AttributeValue.builder()
            .s("sblue@noserver.com")
            .build();

        Map<String, AttributeValue> expressionValues = new HashMap<>();
        expressionValues.put(":value", att);

        Expression expression = Expression.builder()
            .expression("email = :value")
            .expressionValues(expressionValues)
            .build();

        // Create a QueryConditional object that is used in the query operation.
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder().partitionValue("id103")
                .build());
    }
}
```

```
// Get items in the Customer table and write out the ID value.
Iterator<Customer> results = mappedTable.query(r ->
r.queryConditional(queryConditional).filterExpression(expression)).items().iterator();

while (results.hasNext()) {

    Customer rec = results.next();
    System.out.println("The record id is "+rec.getId());
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

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

Retrieve (get) all items from a table

When you want to get all of the records in a given DynamoDB table, use the `scan()` method of your [DynamoDbTable](#) object and the `items()` method to create a set of results against which you can execute various item operations. For example, the following code snippet prints out the ID value of each item in the **Record** table.

Imports

```
import java.time.Instant;
import java.util.Iterator;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void scan( DynamoDbEnhancedClient enhancedClient) {

    try{
        // Create a DynamoDbTable object
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        Iterator<Customer> results = custTable.scan().items().iterator();
        while (results.hasNext()) {

            Customer rec = results.next();
            System.out.println("The record id is "+rec.getId());
            System.out.println("The name is " +rec.getCustName());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

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

For more information, see [Working with items in DynamoDB](#) in the Amazon DynamoDB Developer Guide.

Working with Amazon EC2

This section provides examples of programming [Amazon EC2](#) that use the AWS SDK for Java 2.x.

Topics

- [Manage Amazon EC2 instances](#) (p. 105)
- [Use elastic IP addresses in Amazon EC2](#) (p. 109)
- [Use regions and availability zones](#) (p. 112)
- [Work with Amazon EC2 key pairs](#) (p. 114)
- [Work with security groups in Amazon EC2](#) (p. 116)

Manage Amazon EC2 instances

Create an instance

Create a new Amazon EC2 instance by calling the `Ec2Client`'s `runInstances` method, providing it with a `RunInstancesRequest` containing the [Amazon Machine Image \(AMI\)](#) to use and an [instance type](#).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();

    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
        .build();

    try {
```

```

        ec2.createTags(tagRequest);
        System.out.printf(
            "Successfully started EC2 Instance %s based on AMI %s",
            instanceId, amiId);

        return instanceId;

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

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

Start an instance

To start an Amazon EC2 instance, call the `Ec2Client`'s `startInstances` method, providing it with a [StartInstancesRequest](#) containing the ID of the instance to start.

Imports

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;

```

Code

```

public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}

```

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

Stop an instance

To stop an Amazon EC2 instance, call the `Ec2Client`'s `stopInstances` method, providing it with a [StopInstancesRequest](#) containing the ID of the instance to stop.

Imports

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;

```

Code

```

public static void stopInstance(Ec2Client ec2, String instanceId) {

    StopInstancesRequest request = StopInstancesRequest.builder()

```

```
        .instanceIds(instanceId)
        .build();

ec2.stopInstances(request);
System.out.printf("Successfully stopped instance %s", instanceId);
}
```

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

Reboot an instance

To reboot an Amazon EC2 instance, call the `Ec2Client`'s `rebootInstances` method, providing it with a [RebootInstancesRequest](#) containing the ID of the instance to reboot.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

Code

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {
    try {
        RebootInstancesRequest request = RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        ec2.rebootInstances(request);
        System.out.printf(
            "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Describe instances

To list your instances, create a [DescribeInstancesRequest](#) and call the `Ec2Client`'s `describeInstances` method. 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' `reservations` method, and then call `instances` on each returned [Reservation](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```


Code

```
public static void describeEC2Instances( Ec2Client ec2){

    boolean done = false;
    String nextToken = null;

    try {

        do {

            DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
            DescribeInstancesResponse response = ec2.describeInstances(request);

            for (Reservation reservation : response.reservations()) {
                for (Instance instance : reservation.instances()) {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " + instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("monitoring information is " +
instance.monitoring().state());

                }
            }

            nextToken = response.nextToken();
        } while (nextToken != null);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Results are paged; you can get further results by passing the value returned from the result object's `nextToken` method to a new request object's `nextToken` method, then using the new request object in your next call to `describeInstances`.

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

Monitor an instance

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` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {
```

```
MonitorInstancesRequest request = MonitorInstancesRequest.builder()
    .instanceIds(instanceId).build();

ec2.monitorInstances(request);
System.out.printf(
    "Successfully enabled monitoring for instance %s",
    instanceId);
}
```

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

Stop 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` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);

    System.out.printf(
        "Successfully disabled monitoring for instance %s",
        instanceId);
}
```

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

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
- [MonitorInstances](#) in the Amazon EC2 API Reference
- [UnmonitorInstances](#) in the Amazon EC2 API Reference

Use elastic IP addresses in Amazon EC2

EC2-Classic is retiring

Warning

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

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` method with an `AllocateAddressRequest` object containing the network type (classic Amazon EC2 or VPC).

The returned `AllocateAddressResponse` 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` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static String getAllocateAddress( Ec2Client ec2, String instanceId) {
    try {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
            ec2.allocateAddress(allocateRequest);

        String allocationId = allocateResponse.allocationId();

        AssociateAddressRequest associateRequest =
            AssociateAddressRequest.builder()
                .instanceId(instanceId)
                .allocationId(allocationId)
                .build();

        AssociateAddressResponse associateResponse =
            ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Describe elastic IP addresses

To list the Elastic IP addresses assigned to your account, call the `Ec2Client`'s `describeAddresses` method. It returns a `DescribeAddressesResponse` which you can use to get a list of `Address` objects that represent the Elastic IP addresses on your account.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Address;
import software.amazon.awssdk.services.ec2.model.DescribeAddressesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2Address(Ec2Client ec2 ) {

    try {
        DescribeAddressesResponse response = ec2.describeAddresses();

        for(Address address : response.addresses()) {
            System.out.printf(
                "Found address with public IP %s, " +
                "domain %s, " +
                "allocation id %s " +
                "and NIC id %s",
                address.publicIp(),
                address.domain(),
                address.allocationId(),
                address.networkInterfaceId());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Release an elastic IP address

To release an Elastic IP address, call the Ec2Client's `releaseAddress` method, passing it a [ReleaseAddressRequest](#) containing the allocation ID of the Elastic IP address you want to release.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
```

Code

```
public static void releaseEC2Address(Ec2Client ec2,String allocId) {

    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId).build();

        ReleaseAddressResponse response = ec2.releaseAddress(request);

        System.out.printf(
            "Successfully released elastic IP address %s", allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

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 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` method.

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](#) on GitHub.

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

Use regions and availability zones

Describe regions

To list the Regions available to your account, call the `Ec2Client`'s `describeRegions` method. It returns a [DescribeRegionsResponse](#). Call the returned object's `regions` method to get a list of [Region](#) objects that represent each Region.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.Region;
import software.amazon.awssdk.services.ec2.model.AvailabilityZone;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
```

Code

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    for(Region region : regionsResponse.regions()) {
        System.out.printf(
            "Found Region %s " +
            "with endpoint %s",
            region.regionName(),
            region.endpoint());
        System.out.println();
    }
}
```

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

Describe availability zones

To list each Availability Zone available to your account, call the `Ec2Client`'s `describeAvailabilityZones` method. It returns a `DescribeAvailabilityZonesResponse`. Call its `availabilityZones` method to get a list of `AvailabilityZone` objects that represent each Availability Zone.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.Region;
import software.amazon.awssdk.services.ec2.model.AvailabilityZone;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
```

Code

Create the `Ec2Client`.

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
```

Then call `describeAvailabilityZones()` and retrieve results.

```
DescribeAvailabilityZonesResponse zonesResponse =
    ec2.describeAvailabilityZones();

for(AvailabilityZone zone : zonesResponse.availabilityZones()) {
    System.out.printf(
        "Found Availability Zone %s " +
        "with status %s " +
        "in region %s",
        zone.zoneName(),
        zone.state(),
        zone.regionName());
    System.out.println();
}
```

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

Describe accounts

To describe your account, call the `Ec2Client`'s `describeAccountAttributes` method. This method returns a `DescribeAccountAttributesResponse` object. Invoke this object's `accountAttributes` method to get a list of `AccountAttribute` objects. You can iterate through the list to retrieve an `AccountAttribute` object.

You can get your account's attribute values by invoking the `AccountAttribute` object's `attributeValues` method. This method returns a list of `AccountAttributeValue` objects. You can iterate through this second list to display the value of attributes (see the following code example).

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2Account(Ec2Client ec2) {
    try{
        DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
        accountResults.accountAttributes().forEach(attribute -> {
            System.out.print("\n The name of the attribute is
"+attribute.attributeName());

                attribute.attributeValues().forEach(myValue ->
                    System.out.print("\n The value of the attribute is
"+myValue.attributeValue()));
            }
        );
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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

Work with Amazon EC2 key pairs

Create a key pair

To create a key pair, call the `Ec2Client`'s `createKeyPair` method with a [CreateKeyPairRequest](#) that contains the key's name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void createEC2KeyPair(Ec2Client ec2,String keyName ) {
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName).build();

        ec2.createKeyPair(request);
    }
```

```
        System.out.printf(
            "Successfully created key pair named %s",
            keyName);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Describe key pairs

To list your key pairs or to get information about them, call the `Ec2Client`'s `describeKeyPairs` method. It returns a [DescribeKeyPairsResponse](#) that you can use to access the list of key pairs by calling its `keyPairs` method, which returns a list of [KeyPairInfo](#) objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.KeyPairInfo;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2Keys( Ec2Client ec2){
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();

        for(KeyPairInfo keyPair : response.keyPairs()) {
            System.out.printf(
                "Found key pair with name %s " +
                "and fingerprint %s",
                keyPair.keyName(),
                keyPair.keyFingerprint());
        }

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Delete a key pair

To delete a key pair, call the `Ec2Client`'s `deleteKeyPair` method, passing it a [DeleteKeyPairRequest](#) that contains the name of the key pair to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
```



```
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {  
    try {  
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()  
            .keyName(keyPair)  
            .build();  
  
        DeleteKeyPairResponse response = ec2.deleteKeyPair(request);  
        System.out.printf(  
            "Successfully deleted key pair named %s", keyPair);  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

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

Work with security groups in Amazon EC2

Create a security group

To create a security group, call the `Ec2Client`'s `createSecurityGroup` method with a [CreateSecurityGroupRequest](#) that contains the key's name.

Imports

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.ec2.Ec2Client;  
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;  
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;  
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;  
import software.amazon.awssdk.services.ec2.model.Ec2Exception;  
import software.amazon.awssdk.services.ec2.model.IpPermission;  
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;  
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

```
CreateSecurityGroupRequest createRequest = CreateSecurityGroupRequest.builder()  
    .groupName(groupName)  
    .description(groupDesc)  
    .vpcId(vpcId)  
    .build();
```

```
CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

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

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` method, 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.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

First, create an `Ec2Client`

```
Region region = Region.US_WEST_2;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Then use the `Ec2Client`'s `authorizeSecurityGroupIngress` method,

```
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
    .fromPort(80)
    .ipRanges(ipRange)
    .build();

IpPermission ipPerm2 = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(22)
    .fromPort(22)
    .ipRanges(ipRange)
    .build();

AuthorizeSecurityGroupIngressRequest authRequest =
    AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

AuthorizeSecurityGroupIngressResponse authResponse =
```

```
ec2.authorizeSecurityGroupIngress(authRequest);

System.out.printf(
    "Successfully added ingress policy to Security Group %s",
    groupName);

return resp.groupId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

To add an egress rule to the security group, provide similar data in an [AuthorizeSecurityGroupEgressRequest](#) to the `Ec2Client`'s `authorizeSecurityGroupEgress` method.

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

Describe security groups

To describe your security groups or get information about them, call the `Ec2Client`'s `describeSecurityGroups` method. It returns a [DescribeSecurityGroupsResponse](#) that you can use to access the list of security groups by calling its `securityGroups` method, which returns a list of [SecurityGroup](#) objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId).build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);

        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Delete a security group

To delete a security group, call the `Ec2Client`'s `deleteSecurityGroup` method, passing it a `DeleteSecurityGroupRequest` that contains the ID of the security group to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {
    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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

Working with IAM

This section provides examples of programming [IAM](#) by using the AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. For a complete guide to IAM, visit the [IAM User Guide](#).

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Managing IAM access keys \(p. 120\)](#)
- [Managing IAM Users \(p. 124\)](#)
- [Using IAM account aliases \(p. 127\)](#)
- [Working with IAM policies \(p. 129\)](#)
- [Working with IAM server certificates \(p. 133\)](#)

Managing IAM access keys

Create an access key

To create an IAM access key, call the `IamClient`'s `createAccessKey` method with a `CreateAccessKeyRequest` object.

Note

You must set the region to **AWS_GLOBAL** for `IamClient` calls to work because IAM is a global service.

Imports

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static String createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

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` method.

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.

Imports

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listKeys( IamClient iam,String userName ){
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

The results of `listAccessKeys` are paged (with a default maximum of 100 records per call). You can call `isTruncated` on the returned [ListAccessKeysResponse](#) object to see if the query returned fewer results than are available. If so, then call `marker` on the `ListAccessKeysResponse` and use it when creating a new request. Use that new request in the next invocation of `listAccessKeys`.

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

Retrieve an access key's last used time

To get the time an access key was last used, call the `IamClient`'s `getAccessKeyLastUsed` method with the access key's ID (which can be passed in using a [GetAccessKeyLastUsedRequest](#) object).

You can then use the returned [GetAccessKeyLastUsedResponse](#) object to retrieve the key's last used time.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

    try {
        GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
            .accessKeyId(accessId).build();

        GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

        System.out.println("Access key was last used at: " +
            response.accessKeyLastUsed().lastUsedDate());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

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

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](#), then passing the request object to the `IamClient`'s `updateAccessKey` method.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void updateKey(IamClient iam, String username, String accessId, String
status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
```

```
        .status(statusType)
        .build();

iam.updateAccessKey(request);

System.out.printf(
    "Successfully updated the status of access key %s to" +
    "status %s for user %s", accessId, status, username);
} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Delete an access key

To permanently delete an access key, call the `IamClient`'s `deleteKey` method, 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. 122) method instead.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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

Creating a User

Create a new IAM user by providing the user name to the `IamClient`'s `createUser` method using a [CreateUserRequest](#) object containing the user name.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

Code

```
public static String createIAMUser(IamClient iam, String username ) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Listing Users

To list the IAM users for your account, create a new [ListUsersRequest](#) and pass it to the `IamClient`'s `listUsers` method. You can retrieve the list of users by calling `users` on the returned [ListUsersResponse](#) object.

The list of users returned by `listUsers` is paged. You can check to see there are more results to retrieve by calling the response object's `isTruncated` method. If it returns `true`, then call the response object's `marker()` method. Use the marker value to create a new request object. Then call the `listUsers` method again with the new request.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Updating a User

To update a user, call the `IamClient` object's `updateUser` method, which takes a `UpdateUserRequest` object that you can use to change the user's *name* or *path*.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

Code

```
public static void updateIAMUser(IamClient iam, String curName,String newName ) {
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
            newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Deleting a User

To delete a user, call the `IamClient`'s `deleteUser` request with a `UpdateUserRequest` object set with the user name to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

More Information

- [IAM Users](#) in the IAM User Guide
- [Managing IAM Users](#) in the IAM User Guide
- [CreateUser](#) in the IAM API Reference

- [ListUsers](#) in the IAM API Reference
- [UpdateUser](#) in the IAM API Reference
- [DeleteUser](#) in the IAM API Reference

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.

Create an account alias

To create an account alias, call the `IamClient`'s `createAccountAlias` method with a [CreateAccountAliasRequest](#) object that contains the alias name.

Imports

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void createIAMAccountAlias(IamClient iam, String alias) {
    try {
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

List account aliases

To list your account's alias, if any, call the `IamClient`'s `listAccountAliases` method.

Note

The returned [ListAccountAliasesResponse](#) supports the same `isTruncated` and `marker` methods as other AWS SDK for Java *list* methods, but an S account can have only *one* account alias.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listAliases(IamClient iam) {  
    try {  
        ListAccountAliasesResponse response = iam.listAccountAliases();  
  
        for (String alias : response.accountAliases()) {  
            System.out.printf("Retrieved account alias %s", alias);  
        }  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Delete an account alias

To delete your account's alias, call the `IamClient`'s `deleteAccountAlias` method. When deleting an account alias, you must supply its name using a [DeleteAccountAliasRequest](#) object.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteIAMAccountAlias(IamClient iam, String alias ) {  
    try {  
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()  
            .accountAlias(alias)  
            .build();  
  
        iam.deleteAccountAlias(request);  
        System.out.println("Successfully deleted account alias " + alias);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

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

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

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` method.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

Code

```
public static String createIAMPolicy(IamClient iam, String policyName ) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Get a policy

To retrieve an existing policy, call the `IamClient`'s `getPolicy` method, providing the policy's ARN within a [GetPolicyRequest](#) object.

Imports

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getIAMPolicy(IamClient iam, String policyArn) {
    try {
        GetPolicyRequest request = GetPolicyRequest.builder()
            .policyArn(policyArn).build();

        GetPolicyResponse response = iam.getPolicy(request);
        System.out.format("Successfully retrieved policy %s",
            response.policy().policyName());
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Attach a role policy

You can attach a policy to an IAM [role](#) by calling the `IamClient`'s `attachRolePolicy` method, providing it with the role name and policy ARN in an [AttachRolePolicyRequest](#).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
        }
    }
}
```

```
        if (polArn.compareTo(policyArn)==0) {
            System.out.println(roleName +
                " policy is already attached to this role.");
            return;
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

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

List attached role policies

List attached policies on a role by calling the `IamClient`'s `listAttachedRolePolicies` method. It takes a [ListAttachedRolePoliciesRequest](#) object that contains the role name to list the policies for.

Call `getAttachedPolicies` on the returned [ListAttachedRolePoliciesResponse](#) object to get the list of attached policies. Results may be truncated; if the `ListAttachedRolePoliciesResponse` object's `isTruncated` method returns `true`, call the `ListAttachedRolePoliciesResponse` object's `marker` method. Use the marker returned to create a new request and use it to call `listAttachedRolePolicies` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
    .roleName(roleName)
    .build();
```



```
ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

// Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy: attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn)==0) {
        System.out.println(roleName +
            " policy is already attached to this role.");
        return;
    }
}

AttachRolePolicyRequest attachRequest =
    AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

iam.attachRolePolicy(attachRequest);

System.out.println("Successfully attached policy " + policyArn +
    " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done");
}
```

See the [\[https---github-com-awsdocs-aws-doc-sdk-examples-blob-master-javav2-example-code-iam-src-main-java-com-example-iam-AttachRolePolicy-java\]](https---github-com-awsdocs-aws-doc-sdk-examples-blob-master-javav2-example-code-iam-src-main-java-com-example-iam-AttachRolePolicy-java)[\[complete example\]](#) on GitHub.

Detach a role policy

To detach a policy from a role, call the `IamClient`'s `detachRolePolicy` method, providing it with the role name and policy ARN in a [DetachRolePolicyRequest](#).

Imports

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn ) {
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    }
}
```

```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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
- [AttachRolePolicy](#) in the IAM API Reference
- [ListAttachedRolePolicies](#) in the IAM API Reference
- [DetachRolePolicy](#) 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](#).

Get a server certificate

You can retrieve a server certificate by calling the `IamClient`'s `getServerCertificate` method, passing it a [GetServerCertificateRequest](#) with the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getCertificate(IamClient iam, String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.serverCertificate().certificateBody());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

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

List server certificates

To list your server certificates, call the `IamClient`'s `listServerCertificates` method with a [ListServerCertificatesRequest](#). It returns a [ListServerCertificatesResponse](#).

Call the returned `ListServerCertificateResponse` object's `serverCertificateMetadataList` method to get a list of [ServerCertificateMetadata](#) objects that you can use to get information about each certificate.

Results may be truncated; if the `ListServerCertificateResponse` object's `isTruncated` method returns `true`, call the `ListServerCertificatesResponse` object's `marker` method and use the marker to create a new request. Use the new request to call `listServerCertificates` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listCertificates(IamClient iam) {
    try {
        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListServerCertificatesResponse response;

            if (newMarker == null) {
                ListServerCertificatesRequest request =
                    ListServerCertificatesRequest.builder().build();
                response = iam.listServerCertificates(request);
            } else {
                ListServerCertificatesRequest request =
                    ListServerCertificatesRequest.builder()
                        .marker(newMarker).build();
                response = iam.listServerCertificates(request);
            }

            for(ServerCertificateMetadata metadata :
                response.serverCertificateMetadataList()) {
                System.out.printf("Retrieved server certificate %s",
                    metadata.serverCertificateName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    }
}
```

```
    }  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Update a server certificate

You can update a server certificate's name or path by calling the `IamClient`'s `updateServerCertificate` method. It takes a [UpdateServerCertificateRequest](#) object set with the server certificate's current name and either a new name or new path to use.

Imports

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;  
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

Code

```
public static void updateCertificate(IamClient iam, String curName, String newName) {  
    try {  
        UpdateServerCertificateRequest request =  
            UpdateServerCertificateRequest.builder()  
                .serverCertificateName(curName)  
                .newServerCertificateName(newName)  
                .build();  
  
        UpdateServerCertificateResponse response =  
            iam.updateServerCertificate(request);  
  
        System.out.printf("Successfully updated server certificate to name %s",  
            newName);  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Delete a server certificate

To delete a server certificate, call the `IamClient`'s `deleteServerCertificate` method with a [DeleteServerCertificateRequest](#) containing the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteCert(IamClient iam,String certName ) {  
    try {  
        DeleteServerCertificateRequest request =  
            DeleteServerCertificateRequest.builder()  
                .serverCertificateName(certName)  
                .build();  
  
        iam.deleteServerCertificate(request);  
        System.out.println("Successfully deleted server certificate " +  
            certName);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

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](#)

Working with Amazon Athena

Amazon Athena is a serverless, interactive query service to query data and analyze big data in Amazon S3 by using standard SQL. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with CloudWatch

This section provides examples of programming [CloudWatch](#) by using the AWS SDK for Java 2.x.

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](#).

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Getting metrics from CloudWatch \(p. 137\)](#)

- [Publishing custom metric data to CloudWatch \(p. 138\)](#)
- [Working with CloudWatch alarms \(p. 139\)](#)
- [Using alarm actions in CloudWatch \(p. 142\)](#)
- [Sending events to CloudWatch \(p. 143\)](#)

Getting metrics from CloudWatch

Listing metrics

To list CloudWatch metrics, create a [ListMetricsRequest](#) and call the `CloudWatchClient`'s `listMetrics` method. 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.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

Code

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
        while(!done) {

            ListMetricsResponse response;

            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf(
                    "Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if(response.nextToken() == null) {
```

```
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

The metrics are returned in a [ListMetricsResponse](#) by calling its `getMetrics` method.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `listMetrics` method again with the new request.

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

More information

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

Publishing custom metric data to CloudWatch

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 ").

Publish custom metric data

To publish your own metric data, call the `CloudWatchClient`'s `putMetricData` method 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.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

Code

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
```

```

        .value("URLS")
        .build();

    // Set an Instant object
    String time =
    ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
    Instant instant = Instant.parse(time);

    MetricDatum datum = MetricDatum.builder()
        .metricName("PAGES_VISITED")
        .unit(StandardUnit.NONE)
        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension).build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum).build();

    cw.putMetricData(request);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Successfully put data point %f", dataPoint);
}

```

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

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

Create an alarm

To create an alarm based on a CloudWatch metric, call the `CloudWatchClient`'s `putMetricAlarm` method with a `PutMetricAlarmRequest` filled with the alarm conditions.

Imports

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;

```

Code

```

public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

```



```
try {
    Dimension dimension = Dimension.builder()
        .name("InstanceId")
        .value(instanceId).build();

    PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
        .alarmName(alarmName)
        .comparisonOperator(
            ComparisonOperator.GREATER_THAN_THRESHOLD)
        .evaluationPeriods(1)
        .metricName("CPUUtilization")
        .namespace("AWS/EC2")
        .period(60)
        .statistic(Statistic.AVERAGE)
        .threshold(70.0)
        .actionsEnabled(false)
        .alarmDescription(
            "Alarm when server CPU utilization exceeds 70%")
        .unit(StandardUnit.SECONDS)
        .dimensions(dimension)
        .build();

    cw.putMetricAlarm(request);
    System.out.printf(
        "Successfully created alarm with name %s", alarmName);
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

List alarms

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

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

Code

```
public static void descCWAlarms( CloudWatchClient cw) {

    try {

        boolean done = false;
        String newToken = null;

        while(!done) {
            DescribeAlarmsResponse response;

            if (newToken == null) {
                DescribeAlarmsRequest request =
                DescribeAlarmsRequest.builder().build();
```

```
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

The list of alarms can be obtained by calling `MetricAlarms` on the [DescribeAlarmsResponse](#) that is returned by `describeAlarms`.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `describeAlarms` method again with the new request.

Note

You can also retrieve alarms for a specific metric by using the `CloudWatchClient`'s `describeAlarmsForMetric` method. Its use is similar to `describeAlarms`.

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

Delete alarms

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

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

Code

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);
    }
```

```
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

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.

Note

Alarm actions can be added to an alarm by using the [PutMetricAlarmRequest](#)'s `alarmActions` method when [creating an alarm](#) (p. 139).

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.

Imports

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;  
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;  
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;  
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsResponse;
```

Code

```
public static void enableActions(CloudWatchClient cw, String alarm) {  
    try {  
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()  
            .alarmNames(alarm).build();  
  
        cw.enableAlarmActions(request);  
        System.out.printf(  
            "Successfully enabled actions on alarm %s", alarm);  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

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.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;
```

Code

```
public static void disableActions(CloudWatchClient cw, String alarmName) {
    try {
        DisableAlarmActionsRequest request = DisableAlarmActionsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.disableAlarmActions(request);
        System.out.printf(
            "Successfully disabled actions on alarm %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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.

Add events

To add custom CloudWatch events, call the `CloudWatchEventsClient`'s `putEvents` method 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`.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

Code

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

    try {

        final String EVENT_DETAILS =
            "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Add rules

To create or update a rule, call the `CloudWatchEventsClient`'s `putRule` method 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.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

Code

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
```

```
        .scheduleExpression("rate(5 minutes)")
        .state(RuleState.ENABLED)
        .build();

    PutRuleResponse response = cwe.putRule(request);
    System.out.printf(
        "Successfully created CloudWatch events rule %s with arn %s",
        roleArn, response.ruleArn());
} catch (
    CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

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` method with a [PutTargetsRequest](#) containing the rule to update and a list of targets to add to the rule.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

Code

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

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

Working with AWS CloudTrail

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Cognito

With Amazon Cognito, you can quickly add user sign-up or sign-in capability to your web or mobile app. The examples here demonstrate some of the basic functionality of Amazon Cognito.

Create a user pool

A user pool is a directory of users that you can configure for your web or mobile app.

To create a user pool, start by building a [CreateUserPoolRequest](#) object, with the name of the user pool as the value of its `poolName()`. Call the `createUserPool()` method of your [CreateUserPoolRequest](#), passing in the `CreateUserPoolRequest` object. You can capture the result of this request as a [CreateUserPoolResponse](#) object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;
```

Code

```
public static String createPool(CognitoIdentityProviderClient cognitoClient, String
userPoolName ) {

    try {
        CreateUserPoolResponse response = cognitoClient.createUserPool(
            CreateUserPoolRequest.builder()
                .poolName(userPoolName)
                .build()
        );
    }
}
```

```
    );  
    return response.userPool().id();  
  
    } catch (CognitoIdentityProviderException e){  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

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

List users from a user pool

To list users from your user pools, start by building a [ListUserPoolsRequest](#) object, with the number of maximum results as the value of its `maxResults()`. Call the `listUserPools()` method of your `CognitoIdentityProviderClient`, passing in the `ListUserPoolsRequest` object. You can capture the result of this request as a [ListUserPoolsResponse](#) object, as demonstrated in the following code snippet. Create a [UserPoolDescriptionType](#) object to easily iterate over the results and pull out the attributes of each user.

Imports

```
import software.amazon.awssdk.regions.Region;  
import  
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;  
import  
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;  
import software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;  
import software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

Code

```
public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient ) {  
  
    try {  
        ListUserPoolsRequest request = ListUserPoolsRequest.builder()  
            .maxResults(10)  
            .build();  
  
        ListUserPoolsResponse response = cognitoClient.listUserPools(request);  
        response.userPools().forEach(userpool -> {  
            System.out.println("User pool " + userpool.name() + ", User ID " +  
userpool.id() );  
        }  
    );  
  
    } catch (CognitoIdentityProviderException e){  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Create an identity pool

An identity pool is a container that organizes the IDs from your external identity provider, keeping a unique identifier for each user. To create an identity pool, start by building a [CreateIdentityPoolRequest](#) with the name of the user pool as the value of its `identityPoolName()`.

Set `allowUnauthenticatedIdentities()` to `true` or `false`. Call the `createIdentityPool()` method of your `CognitoIdentityClient` object, passing in the `CreateIdentityPoolRequest` object. You can capture the result of this request as a `CreateIdentityPoolResponse` object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
```

Code

```
public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName ) {

    try {
        CreateIdentityPoolRequest poolRequest = CreateIdentityPoolRequest.builder()
            .allowUnauthenticatedIdentities(false)
            .identityPoolName(identityPoolName)
            .build() ;

        CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
        return response.identityPoolId();

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Add an app client

To enable the hosted web sign-up or sign-in UI for your app, create an app client. To create an app client, start by building a `CreateUserPoolClientRequest` object, with the name of the client as the value of its `clientName()`. Set `userPoolId()` to the ID of the user pool to which you want to attach this app client. Call the `createUserPoolClient()` method of your `CognitoIdentityProviderClient`, passing in the `CreateUserPoolClientRequest` object. You can capture the result of this request as a `CreateUserPoolClientResponse` object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import
software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;
```

Code

```

public static void createPoolClient ( CognitoIdentityProviderClient cognitoClient,
                                     String clientName,
                                     String userPoolId ) {

    try {

        CreateUserPoolClientResponse response = cognitoClient.createUserPoolClient(
            CreateUserPoolClientRequest.builder()
                .clientName(clientName)
                .userPoolId(userPoolId)
                .build()

        );

        System.out.println("User pool " + response.userPoolClient().clientName() + "
created. ID: " + response.userPoolClient().clientId());

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

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

Add a third-party identity provider

Adding an external identity provider (IdP) enables your users to log into your app using that service's login mechanism. To add a third-party IdP, start by building an [UpdateIdentityPoolRequest](#) object, with the name of the identity pool as the value of its `identityPoolName()`. Set `allowUnauthenticatedIdentities()` to true or false, specify the `identityPoolId()`, and define which login providers will be supported with `supportedLoginProviders()`. Call the `updateIdentityPool()` method of your `CognitoIdentityClient`, passing in the `UpdateIdentityPoolRequest` object. You can capture the result of this request as an [UpdateIdentityPoolResponse](#) object, as demonstrated in the following code snippet.

Imports

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import software.amazon.awssdk.services.cognitoidentity.model.CognitoIdentityProvider;
import software.amazon.awssdk.services.cognitoidentity.model.UpdateIdentityPoolRequest;
import software.amazon.awssdk.services.cognitoidentity.model.UpdateIdentityPoolResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import java.util.ArrayList;
import java.util.List;

```

Code

```

public static void createNewUser(CognitoIdentityProviderClient cognitoClient,
                                String userPoolId,
                                String name,
                                String email,
                                String password){

    try{

        AttributeType userAttrs = AttributeType.builder()
            .name("email")
            .value(email)
            .build();
    }
}

```

```
AdminCreateUserRequest userRequest = AdminCreateUserRequest.builder()
    .userPoolId(userPoolId)
    .username(name)
    .temporaryPassword(password)
    .userAttributes(userAttrs)
    .messageAction("SUPPRESS")
    .build();

AdminCreateUserResponse response = cognitoClient.adminCreateUser(userRequest);
System.out.println("User " + response.user().username() + "is created. Status:
" + response.user().userStatus());

} catch (CognitoIdentityProviderException e){
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Get credentials for an ID

To get the credentials for an identity in an identity pool, first build a [GetCredentialsForIdentityRequest](#) with the identity ID as the value of its `identityId()`. Call the `getCredentialsForIdentity()` method of your `CognitoIdentityClient`, passing in the `GetCredentialsForIdentityRequest`. You can capture the result of this request as a [GetCredentialsForIdentityResponse](#) object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
```

Code

```
public static void getCredsForIdentity(CognitoIdentityClient cognitoClient, String
identityId) {
    try {
        GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest.builder()
            .identityId(identityId)
            .build();

        GetCredentialsForIdentityResponse response =
cognitoClient.getCredentialsForIdentity(getCredentialsForIdentityRequest);
        System.out.println("Identity ID " + response.identityId() + ", Access key ID "
+ response.credentials().accessKeyId());
    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

For more information, see the [Amazon Cognito Developer Guide](#).

Working with Amazon Comprehend

Amazon Comprehend is a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon EventBridge

Amazon EventBridge delivers a stream of real-time data from event sources, such as Zendesk, Datadog, or Pagerduty, and routes that data to targets like AWS Lambda. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Kinesis Data Firehose

Amazon Kinesis Data Firehose provides a simple way to capture, transform, and load streaming data. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Forecast

Amazon Forecast is a fully managed service for time-series forecasting. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with AWS Glue

With AWS Glue, you can fully manage, extract, transform, and load (ETL) your data for analytics. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Kinesis

This section provides examples of programming [Amazon Kinesis](#) using the AWS SDK for Java 2.x.

For more information about Kinesis, see the [Amazon Kinesis Developer Guide](#).

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Subscribing to Amazon Kinesis Data Streams \(p. 152\)](#)

Subscribing to Amazon Kinesis Data Streams

The following examples show you how to retrieve and process data from Amazon Kinesis Data Streams using the `subscribeToShard` method. Kinesis Data Streams now employs the enhanced fanout feature and a low-latency HTTP/2 data retrieval API, making it easier for developers to run multiple low-latency, high-performance applications on the same Kinesis Data Stream.

Set up

First, create an asynchronous Kinesis client and a [SubscribeToShardRequest](#) object. These objects are used in each of the following examples to subscribe to Kinesis events.

Imports

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

Code

```
Region region = Region.US_EAST_1;
KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();

SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
    .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/StockTradeStream")
    .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

Use the builder interface

You can use the `builder` method to simplify the creation of the [SubscribeToShardResponseHandler](#).

Using the builder, you can set each lifecycle callback with a method call instead of implementing the full interface.

Code

```
private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
```

```
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .onComplete(() -> System.out.println("All records stream successfully"))
        // Must supply some type of subscriber
        .subscriber(e -> System.out.println("Received event - " + e))
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

For more control of the publisher, you can use the `publisherTransformer` method to customize the publisher.

Code

```
    private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
        .subscriber(e -> System.out.println("Received event - " + e))
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

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

Use a custom response handler

For full control of the subscriber and publisher, implement the [SubscribeToShardResponseHandler](#) interface.

In this example, you implement the `onEventStream` method, which allows you full access to the publisher. This demonstrates how to transform the publisher to event records for printing by the subscriber.

Code

```
    private static CompletableFuture<Void> responseHandlerBuilderClassic(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream> publisher)
{
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
                .flatMapIterable(SubscribeToShardEvent::records)
                // Limit to 1000 total records
                .limit(1000)
        }
    }
}
```

```
        // Batch records into lists of 25
        .buffer(25)
        // Print out each record batch
        .subscribe(batch -> System.out.println("Record Batch - " + batch));
    }

    @Override
    public void complete() {
        System.out.println("All records stream successfully");
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("Error during stream - " + throwable.getMessage());
    }
};
return client.subscribeToShard(request, responseHandler);
}
```

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

Use the visitor interface

You can use a [Visitor](#) object to subscribe to specific events you're interested in watching.

Code

```
private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
    .builder()
    .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
    .build();
    SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .subscriber(visitor)
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

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

Use a custom subscriber

You can also implement your own custom subscriber to subscribe to the stream.

This code snippet shows an example subscriber.

Code

```
private static class MySubscriber implements Subscriber<SubscribeToShardEventStream> {

    private Subscription subscription;
    private AtomicInteger eventCount = new AtomicInteger(0);

    @Override
    public void onSubscribe(Subscription subscription) {
```

```
        this.subscription = subscription;
        this.subscription.request(1);
    }

    @Override
    public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
        System.out.println("Received event " + shardSubscriptionEventStream);
        if (eventCount.incrementAndGet() >= 100) {
            // You can cancel the subscription at any time if you wish to stop
            receiving events.
            subscription.cancel();
        }
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
        System.err.println("Error occurred while stream - " + throwable.getMessage());
    }

    @Override
    public void onComplete() {
        System.out.println("Finished streaming all events");
    }
}
```

You can pass that custom subscriber to the subscribe method, similarly to preview examples. The following code snippet shows this example.

Code

```
private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(MySubscriber::new)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

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

Write data records into a Kinesis data stream

You can use the [AmazonKinesisClient](#) object to write data records into a Kinesis data stream by using the `putRecords` method. To successfully invoke this method, create a [PutRecordsRequest](#) object. You pass the name of the data stream to the `streamName` method. Also you must pass the data by using the `putRecords` method (as shown in the following code example).

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```


In the following Java code example, notice that **StockTrade** object is used as the data to write to the Kinesis data stream. Before running this example, ensure that you have created the data stream.

Code

```

public static void setStockData( KinesisClient kinesisClient, String streamName) {
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x=0; x<index; x++){
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
                                   String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by the
    Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as the
        partition key, explained in the Supplemental Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        e.getMessage();
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName) {
    try {
        DescribeStreamRequest describeStreamRequest = DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
            kinesisClient.describeStream(describeStreamRequest);

        if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE")) {
            System.err.println("Stream " + streamName + " is not active. Please wait a
few moments and try again.");
        }
    }
}

```

```
        System.exit(1);
    }

    }catch (KinesisException e) {
        System.err.println("Error found while describing the stream " + streamName);
        System.err.println(e);
        System.exit(1);
    }
}
```

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

Use a third-party library

You can use other third-party libraries instead of implementing a custom subscriber. This example demonstrates using the RxJava implementation, but you can use any library that implements the Reactive Streams interfaces. See the [RxJava wiki page on Github](#) for more information on that library.

To use the library, add it as a dependency. If you're using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

Imports

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;

import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

This example uses RxJava in the `onEventStream` lifecycle method. This gives you full access to the publisher, which can be used to create an Rx Flowable.

Code

```
SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " + t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)
        .flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000))
```

```
                .buffer(25)
                .subscribe(e -> System.out.println("Record batch = " + e)))
            .build();
```

You can also use the `publisherTransformer` method with the `Flowable` publisher. You must adapt the `Flowable` publisher to an `SdkPublisher`, as shown in the following example.

Code

```
SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " + t.getMessage()))
    .publisherTransformer(p ->
        SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

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

More information

- [SubscribeToShardEvent](#) in the Amazon Kinesis API Reference
- [SubscribeToShard](#) in the Amazon Kinesis API Reference

Working with AWS Key Management Service

Amazon Kinesis is a secure and resilient service that uses hardware security modules that have been validated under FIPS 140-2, or are in the process of being validated, to protect your keys. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Invoke, list, and delete AWS Lambda functions

This section provides examples of programming with the Lambda service client by using the AWS SDK for Java 2.x.

Topics

- [Invoke a Lambda function \(p. 158\)](#)
- [List Lambda functions \(p. 159\)](#)
- [Delete a Lambda function \(p. 160\)](#)

Invoke a Lambda function

You can invoke a Lambda function by creating a `LambdaClient` object and invoking its `invoke` method. Create an `InvokeRequest` object to specify additional information such as the function name and the payload to pass to the Lambda function. Function names appear as `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction`. You can retrieve the value by looking at the function in the AWS Management Console.

To pass payload data to a function, create a `SdkBytes` object that contains information. For example, in the following code example, notice the JSON data passed to the Lambda function.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

The following code example demonstrates how to invoke a Lambda function.

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null ;
    try {
        //Need a SdkBytes instance for the payload
        String json = "{\"Hello \": \"Paris\"}";
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        //Setup an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

List Lambda functions

Build a [LambdaClient](#) object and invoke its `listFunctions` method. This method returns a [ListFunctionsResponse](#) object. You can invoke this object's `functions` method to return a list of [FunctionConfiguration](#) objects. You can iterate through the list to retrieve information about the functions. For example, the following Java code example shows how to get each function name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

Code

The following Java code example demonstrates how to retrieve a list of function names.

```
public static void listFunctions(LambdaClient awsLambda) {
```

```
try {
    ListFunctionsResponse functionResult = awsLambda.listFunctions();
    List<FunctionConfiguration> list = functionResult.functions();

    for (FunctionConfiguration config: list) {
        System.out.println("The function name is "+config.functionName());
    }

} catch(LambdaException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

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

Delete a Lambda function

Build a [LambdaClient](#) object and invoke its `deleteFunction` method. Create a [DeleteFunctionRequest](#) object and pass it to the `deleteFunction` method. This object contains information such as the name of the function to delete. Function names appear as *arn:aws:lambda:us-east-1:123456789012:function:HelloFunction*. You can retrieve the value by looking at the function in the AWS Management Console.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

The following Java code demonstrates how to delete a Lambda function.

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String functionName ) {
    try {
        //Setup an DeleteFunctionRequest
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Working with AWS Elemental MediaConvert

AWS Elemental MediaConvert is a file-based video processing service that allows video providers to transcode content for broadcast and multiscreen delivery. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with AWS Elemental MediaStore

AWS Elemental MediaStore is an AWS storage service optimized for media. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with AWS Migration Hub

AWS Migration Hub provides a single place to monitor migrations in any AWS region where your migration tools are available. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Personalize

Amazon Personalize is a machine learning service that makes it easy for developers to create individualized recommendations for customers. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Pinpoint

You can use Amazon Pinpoint to send relevant, personalized messages to your customers via multiple communication channels, such as push notifications, SMS, and email.

Create a project

A project (or application) in Amazon Pinpoint is a collection of settings, customer data, segments, and campaigns.

To create a project, start by building a `CreateApplicationRequest` object with the name of the project as the value of its `name()`. Then build a `CreateAppRequest` object, passing in the `CreateApplicationRequest` object as the value of its `createApplicationRequest()` method. Call the `createApp()` method of your `PinpointClient`, passing in the `CreateAppRequest` object. Capture the result of this request as a `CreateAppResponse` object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
public static String createApplication(PinpointClient pinpoint, String appName) {
    try {
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();
    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Create a dynamic segment

A segment is a set of customers who share specific attributes, such as the city they live in or how frequently they visit your website. A dynamic segment is one that's based on attributes that you define, and can change over time.

To create a dynamic segment, first build all of the dimensions you want for this segment. For example, the following code snippet is set to include customers who were active on the site in the last 30 days. You can do this by first building a [RecencyDimension](#) object with the `duration()` and `recencyType()` you want (that is, `ACTIVE` or `INACTIVE`), and then passing this object to a [SegmentBehaviors](#) builder object as the value of `recency()`.

When you have defined your segment attributes, build them into a [SegmentDimensions](#) object. Then build a [WriteSegmentRequest](#) object, passing in the `SegmentDimensions` object as the value of its `dimensions()`. Next, build a [CreateSegmentRequest](#) object, passing in the `WriteSegmentRequest` object as the value of its `writeSegmentRequest()`. Finally, call the `createSegment()` method of your `PinpointClient`, passing in the `CreateSegmentRequest` object. Capture the result of this request as a [CreateSegmentResponse](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;
```

Code

```
public static SegmentResponse createSegment(PinpointClient client, String appId) {
    try {
        Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
        segmentAttributes.put("Team", AttributeDimension.builder()
            .attributeType(AttributeType.INCLUSIVE)
            .values("Lakers")
            .build());

        RecencyDimension recencyDimension = RecencyDimension.builder()
            .duration("DAY_30")
            .recencyType("ACTIVE")
            .build();

        SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
            .recency(recencyDimension)
            .build();

        SegmentDemographics segmentDemographics = SegmentDemographics
            .builder()
            .build();

        SegmentLocation segmentLocation = SegmentLocation
            .builder()
            .build();

        SegmentDimensions dimensions = SegmentDimensions
            .builder()
            .attributes(segmentAttributes)
            .behavior(segmentBehaviors)
            .demographic(segmentDemographics)
            .location(segmentLocation)
            .build();

        WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
            .name("MySegment")
            .dimensions(dimensions)
            .build();

        CreateSegmentRequest createSegmentRequest = CreateSegmentRequest.builder()
            .applicationId(appId)
            .writeSegmentRequest(writeSegmentRequest)
            .build();

        CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
        System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
        System.out.println("Done");
        return createSegmentResult.segmentResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```


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

Import a static segment

A static segment is one you create and import from outside of Amazon Pinpoint. The following example code shows how to create a static segment by importing it from Amazon S3.

Prerequisite

Before you can complete this example, you need to create an IAM role that grants Amazon Pinpoint access to Amazon S3. For more information, see [IAM role for importing endpoints or segments](#) in the Amazon Pinpoint Developer Guide.

To import a static segment, start by building an [ImportJobRequest](#) object. In the builder, specify the `s3Url()`, `roleArn()`, and `format()`.

Note

For more information about the properties of an `ImportJobRequest`, see [the ImportJobRequest section of Import Jobs](#) in the Amazon Pinpoint API Reference.

Then build a [CreateImportJobRequest](#) object, passing in the `ImportJobRequest` object as the value of its `importJobRequest()`, and the ID of your project as the `applicationId()`. Call the `createImportJob()` method of your `PinpointClient`, passing in the `CreateImportJobRequest` object. Capture the result of this request as a `CreateImportJobResponse` object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
public static ImportJobResponse createImportSegment(PinpointClient client,
                                                    String appId,
                                                    String bucket,
                                                    String key,
                                                    String roleArn) {

    try {
        ImportJobRequest importRequest = ImportJobRequest.builder()
            .defineSegment(true)
            .registerEndpoints(true)
            .roleArn(roleArn)
            .format(Format.JSON)
            .s3Url("s3://" + bucket + "/" + key)
            .build();

        CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
            .importJobRequest(importRequest)
            .applicationId(appId)
            .build();

        CreateImportJobResponse jobResponse = client.createImportJob(jobRequest);
```

```
        return jobResponse.importJobResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

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

List segments for your project

To list the segments associated with a particular project, start by building a [GetSegmentsRequest](#) object, with the ID of the project as the value of its `applicationId()`. Next, call the `getSegments()` method of your `PinpointClient`, passing in the `GetSegmentsRequest` object. Capture the result of this request as a [GetSegmentsResponse](#) object. Finally, instantiate a `List` object upcasted to the [SegmentResponse](#) class. Then call the `segmentsResponse().item()` of `GetSegmentsResponse`, as demonstrated in the following code snippet. From there, you can iterate through the results.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;
```

Code

```
public static void listSegs( PinpointClient pinpoint, String appId) {

    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();

        for(SegmentResponse segment: segments) {
            System.out.println("Segment " + segment.id() + " " + segment.name() + " "
+ segment.lastModifiedDate());
        }
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Create a campaign

A campaign is an initiative meant to engage a particular audience segment by sending messages to those customers.

To create a campaign, first build all of the settings you want for this campaign. In the following code snippet, for example, the campaign will start immediately because the `startTime()` of the [Schedule](#) is set to `IMMEDIATE`. To set it to start at a specific time instead, specify a time in ISO 8601 format.

Note

For more information about the settings available for campaigns, see the [Schedule](#) section of [Campaigns](#) in the Amazon Pinpoint API Reference.

After you define your campaign configuration, build it into a [WriteCampaignRequest](#) object. None of the methods of the `builder()` of the `WriteCampaignRequest` are required. But you do need to include any of the configuration settings ([MessageConfiguration](#)) that you set for the campaign. We also recommend that you include a name and a description for your campaign so you can easily distinguish it from other campaigns. Call the `createCampaign()` method of your `PinpointClient`, passing in the `WriteCampaignRequest` object. Capture the result of this request as a [CreateCampaignResponse](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
public static void createPinCampaign(PinpointClient pinpoint, String appId, String
segmentId) {

    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());

}

public static CampaignResponse createCampaign(PinpointClient client, String appId,
String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration = MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
```

```
        .name("MyCampaign")
        .segmentId(segmentID)
        .messageConfiguration(messageConfiguration)
        .build();

    CreateCampaignResponse result = client.createCampaign(
        CreateCampaignRequest.builder()
            .applicationId(appID)
            .writeCampaignRequest(request).build()
    );

    System.out.println("Campaign ID: " + result.campaignResponse().id());

    return result.campaignResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return null;
}
```

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

Send a message

To send an SMS text message through Amazon Pinpoint, first build an [AddressConfiguration](#) object to specify the `channelType()`. (In the following example, it's set to `ChannelType.SMS` to indicate the message will be sent via SMS.) Initialize a [HashMap](#) to store the destination phone number and the [AddressConfiguration](#) object. Next, build an [SMSMessage](#) object containing the relevant values. These include the `originationNumber`, the type of message (`messageType`), and the body of the message itself.

When you have created the message, build the [SMSMessage](#) object into a [DirectMessageConfiguration](#) object. Build your [Map](#) object and [DirectMessageConfiguration](#) object into a [MessageRequest](#) object. Build a [SendMessageRequest](#) object, including your project ID (`applicationId`) and your [MessageRequest](#) object. Call the `sendMessages()` method of your [PinpointClient](#), passing in the [SendMessageRequest](#) object. Capture the result of this request as a [SendMessageResponse](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;
```

Code

```
public static void sendSMSMessage(PinpointClient pinpoint, String message, String
appId, String originationNumber, String destinationNumber) {
```

```
try {  
  
    Map<String, AddressConfiguration> addressMap =  
        new HashMap<String, AddressConfiguration>();  
  
    AddressConfiguration addConfig = AddressConfiguration.builder()  
        .channelType(ChannelType.SMS)  
        .build();  
  
    addressMap.put(destinationNumber, addConfig);  
  
    SMSMessage smsMessage = SMSMessage.builder()  
        .body(message)  
        .messageType(messageType)  
        .originationNumber(originationNumber)  
        .senderId(senderId)  
        .keyword(registeredKeyword)  
        .build();  
  
    // Create a DirectMessageConfiguration object  
    DirectMessageConfiguration direct = DirectMessageConfiguration.builder()  
        .smsMessage(smsMessage)  
        .build();  
  
    MessageRequest msgReq = MessageRequest.builder()  
        .addresses(addressMap)  
        .messageConfiguration(direct)  
        .build();  
  
    // create a SendMessagesRequest object  
    SendMessagesRequest request = SendMessagesRequest.builder()  
        .applicationId(appId)  
        .messageRequest(msgReq)  
        .build();  
  
    SendMessagesResponse response= pinpoint.sendMessages(request);  
  
    MessageResponse msg1 = response.messageResponse();  
    Map map1 = msg1.result();  
  
    //Write out the result of sendMessage  
    map1.forEach((k, v) -> System.out.println((k + ":" + v)));  
  
} catch (PinpointException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

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

For more information, see the [Amazon Pinpoint Developer Guide](#).

Working with Amazon Polly

Amazon Polly is a service that turns text into lifelike speech, allowing you to create applications that talk, and build entirely new categories of speech-enabled functionality. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Rekognition

With Amazon Rekognition, you can perform fast and accurate face searches, allowing you to identify a person in a photo or video using your private repository of face images. You can also verify identity by analyzing a face image against images you have stored for comparison. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon SageMaker

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with AWS Secrets Manager

AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Simple Email Service

Amazon Simple Email Service (Amazon SES) is a cost-effective, flexible, and scalable email service that enables developers to send mail from within any application. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Simple Notification Service

With Amazon Simple Notification Service, you can easily push real-time notification messages from your applications to subscribers over multiple communication channels. This topic describes how to perform some of the basic functions of Amazon SNS.

Create a topic

A **topic** is a logical grouping of communication channels that defines which systems to send a message to, for example, fanning out a message to AWS Lambda and an HTTP webhook. You send messages to Amazon SNS, then they're distributed to the channels defined in the topic. This makes the messages available to subscribers.

To create a topic, first build a [CreateTopicRequest](#) object, with the name of the topic set using the `name()` method in the builder. Then, send the request object to Amazon SNS by using the `createTopic()` method of the [SnsClient](#). You can capture the result of this request as a [CreateTopicResponse](#) object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

List your Amazon SNS topics

To retrieve a list of your existing Amazon SNS topics, build a [ListTopicsRequest](#) object. Then, send the request object to Amazon SNS by using the `listTopics()` method of the `SnsClient`. You can capture the result of this request as a [ListTopicsResponse](#) object.

The following code snippet prints out the HTTP status code of the request and a list of Amazon Resource Names (ARNs) for your Amazon SNS topics.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() + "\n"
            + "\nTopics\n\n" + result.topics());
    } catch (SnsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Subscribe an endpoint to a topic

After you create a topic, you can configure which communication channels will be endpoints for that topic. Messages are distributed to these endpoints after Amazon SNS receives them.

To configure a communication channel as an endpoint for a topic, subscribe that endpoint to the topic. To start, build a [SubscribeRequest](#) object. Specify the communication channel (for example, `lambda` or `email`) as the `protocol()`. Set the `endpoint()` to the relevant output location (for example, the ARN of a Lambda function or an email address), and then set the ARN of the topic to which you want to subscribe as the `topicArn()`. Send the request object to Amazon SNS by using the `subscribe()` method of the `SnsClient`. You can capture the result of this request as a [SubscribeResponse](#) object.

The following code snippet shows how to subscribe an email address to a topic.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

Code

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
            + "Status is " + result.sdkHttpResponse().statusCode());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Publish a message to a topic

After you have a topic and one or more endpoints configured for it, you can publish a message to it. To start, build a [PublishRequest](#) object. Specify the `message()` to send, and the ARN of the topic (`topicArn()`) to send it to. Then, send the request object to Amazon SNS by using the `publish()` method of the `SnsClient`. You can capture the result of this request as a [PublishResponse](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status is " +
            result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Unsubscribe an endpoint from a topic

You can remove the communication channels configured as endpoints for a topic. After doing that, the topic itself continues to exist and distribute messages to any other endpoints configured for that topic.

To remove a communication channel as an endpoint for a topic, unsubscribe that endpoint from the topic. To start, build an [UnsubscribeRequest](#) object and set the ARN of the topic you want to unsubscribe from as the `subscriptionArn()`. Then send the request object to SNS by using the `unsubscribe()` method of the `SnsClient`. You can capture the result of this request as an [UnsubscribeResponse](#) object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

Code

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);

        System.out.println("\n\nStatus was " + result.sdkHttpResponse().statusCode())
```

```
        + "\n\nSubscription was removed for " + request.subscriptionArn());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Delete a topic

To delete an Amazon SNS topic, first build a [DeleteTopicRequest](#) object with the ARN of the topic set as the `topicArn()` method in the builder. Then send the request object to Amazon SNS by using the `deleteTopic()` method of the `SnsClient`. You can capture the result of this request as a [DeleteTopicResponse](#) object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

For more information, see the [Amazon Simple Notification Service Developer Guide](#).

Working with Amazon Simple Queue Service

This section provides examples of programming [Amazon Simple Queue Service](#) using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Working with Amazon Simple Queue Service message queues \(p. 174\)](#)
- [Sending, receiving, and deleting Amazon Simple Queue Service messages \(p. 176\)](#)

Working with Amazon Simple Queue Service message queues

A *message queue* is the logical container used for sending messages reliably in Amazon Simple Queue Service. 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](#).

This topic describes how to create, list, delete, and get the URL of an Amazon Simple Queue Service queue by using the AWS SDK for Java.

Create a queue

Use the `SqsClient`'s `createQueue` method, and provide a `CreateQueueRequest` object that describes the queue parameters.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);
```

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

List queues

To list the Amazon Simple Queue Service queues for your account, call the `SqsClient`'s `listQueues` method with a `ListQueuesRequest` object.

Using the `listQueues` overload without any parameters returns *all queues*, up to 1,000 queues. You can supply a queue name prefix to the `ListQueuesRequest` object to limit the results to queues that match that prefix.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
                ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
                sqsClient.listQueues(listQueuesRequest);

            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }
        }
```

```
    }  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Get the URL for a queue

Call the `SqsClient`'s `getQueueUrl` method. with a `GetQueueUrlRequest` object.

Imports

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.*;  
import java.util.List;
```

Code

```
    GetQueueUrlResponse getQueueUrlResponse =  
    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());  
    String queueUrl = getQueueUrlResponse.queueUrl();  
    return queueUrl;  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    }  
    return "";
```

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

Delete a queue

Provide the queue's [URL \(p. 175\)](#) to the `DeleteMessageRequest` object. Then call the `SqsClient`'s `deleteQueue` method.

Imports

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.*;  
import java.util.List;
```

Code

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {  
    try {  
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()  
            .queueName(queueName)  
            .build();  
        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();  
    }  
}
```

```
DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
    .queueUrl(queueUrl)
    .build();

sqsClient.deleteQueue(deleteQueueRequest);

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

More information

- [How Amazon Simple Queue Service 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 Simple Queue Service messages

A message is a piece of data that can be sent and received by distributed components. Messages are always delivered using an [SQS Queue](#) (p. 174).

Send a message

Add a single message to an Amazon Simple Queue Service queue by calling the `SqsClient` client `sendMessage` method. Provide a [SendMessageRequest](#) object that contains the queue's [URL](#) (p. 175), message body, and optional delay value (in seconds).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());
```

Send multiple messages in a request

Send more than one message in a single request by using the `SqsClient` `sendMessageBatch` method. This method takes a [SendMessageBatchRequest](#) that contains the queue URL and a list of messages to send. (Each message is a [SendMessageBatchRequestEntry](#).) You can also delay sending a specific message by setting a delay value on the message.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),
            SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

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

Retrieve Messages

Retrieve any messages that are currently in the queue by calling the `SqsClient receiveMessage` method. This method takes a [ReceiveMessageRequest](#) that contains the queue URL. You can also specify the maximum number of messages to return. Messages are returned as a list of [Message](#) objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
ReceiveMessageRequest receiveMessageRequest = ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
return messages;
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

Delete a message after receipt

After receiving a message and processing its contents, delete the message from the queue by sending the message's receipt handle and queue URL to the `SqsClient deleteMessage` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

Code

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest = DeleteMessageRequest.builder()
            .queueUrl(queueUrl)
            .receiptHandle(message.receiptHandle())
            .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

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

More Info

- [How Amazon Simple Queue Service 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

Working with AWS Systems Manager

AWS Systems Manager is Amazon software that can be installed and configured on an Amazon EC2 instance, an on-premises server, or a virtual machine (VM). See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Simple Workflow Service

The Amazon Simple Workflow Service makes it easy to build applications that coordinate work across distributed components. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Textract

Amazon Textract is a fully managed machine learning service that automatically extracts text and data from scanned documents. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon Transcribe

This section provides examples of programming [Amazon Transcribe](#) using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The [complete example code is available on GitHub](#). From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- [Working with Amazon Transcribe \(p. 179\)](#)

Working with Amazon Transcribe

The following example shows how bidirectional streaming works using Amazon Transcribe. Bidirectional streaming implies that there's both a stream of data going to the service and being received back in real time. The example uses Amazon Transcribe streaming transcription to send an audio stream and receive a stream of transcribed text back in real time.

See [Streaming Transcription](#) in the Amazon Transcribe Developer Guide to learn more about this feature.

See [Getting Started](#) in the Amazon Transcribe Developer Guide to get started using Amazon Transcribe.

Set up the microphone

This code uses the `javax.sound.sampled` package to stream audio from an input device.

Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

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

Create a publisher

This code implements a publisher that publishes audio data from the Amazon Transcribe audio stream.

Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
```



```

import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (TranscribeStreamingException e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {

```

```

    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

        int len = 0;
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
        return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
    }
}

```

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

Create the client and start the stream

In the main method, create a request object, start the audio input stream and instantiate the publisher with the audio input.

You must also create a [StartStreamTranscriptionResponseHandler](#) to specify how to handle the response from Amazon Transcribe.

Then, use the `TranscribeStreamingAsyncClient`'s `startStreamTranscription` method to start the bidirectional streaming.

Imports

```

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;

```

Code

```
public static void convertAudio(TranscribeStreamingAsyncClient client) throws Exception
{
    try {
        StartStreamTranscriptionRequest request =
        StartStreamTranscriptionRequest.builder()
            .mediaEncoding(MediaEncoding.PCM)
            .languageCode(LanguageCode.EN_US)
            .mediaSampleRateHertz(16_000).build();

        TargetDataLine mic = Microphone.get();
        mic.start();

        AudioStreamPublisher publisher = new AudioStreamPublisher(new
        AudioInputStream(mic));

        StartStreamTranscriptionResponseHandler response =
            StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                TranscriptEvent event = (TranscriptEvent) e;
                event.transcript().results().forEach(r ->
                r.alternatives().forEach(a -> System.out.println(a.transcript())));
            }).build();

        // Keeps Streaming until you end the Java program
        client.startStreamTranscription(request, publisher, response);

    } catch (TranscribeStreamingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

More information

- [How It Works](#) in the Amazon Transcribe Developer Guide.
- [Getting Started With Streaming Audio](#) in the Amazon Transcribe Developer Guide.
- [Guidelines and Limits](#) in the Amazon Transcribe Developer Guide.

Working with Amazon Translate

Amazon Translate removes the complexity of building real-time and batch translation capabilities into your applications. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

Working with Amazon WorkDocs

Amazon WorkDocs is a fully managed, secure content creation, storage, and collaboration service. See the following resources for complete code examples with instructions.

[Link to Github](#)

[Link to AWS Code Sample Catalog](#)

SDK for Java 2.x code examples

The code examples in this topic show you how to use the AWS SDK for Java 2.x with AWS.

The examples are divided into the following categories:

Single-service actions

Code excerpts that show you how to call individual service functions.

Single-service scenarios

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

Cross-service examples

Sample applications that work across multiple AWS services.

Examples

- [Single-service actions and scenarios using SDK for Java 2.x \(p. 183\)](#)
- [Cross-service examples using SDK for Java 2.x \(p. 442\)](#)

Single-service actions and scenarios using SDK for Java 2.x

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS services.

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

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

Services

- [API Gateway examples using SDK for Java 2.x \(p. 184\)](#)
- [Application Recovery Controller examples using SDK for Java 2.x \(p. 186\)](#)
- [CloudFront examples using SDK for Java 2.x \(p. 188\)](#)
- [CloudWatch examples using SDK for Java 2.x \(p. 190\)](#)
- [CloudWatch Events examples using SDK for Java 2.x \(p. 195\)](#)
- [CloudWatch Logs examples using SDK for Java 2.x \(p. 197\)](#)
- [Amazon Cognito Identity Provider examples using SDK for Java 2.x \(p. 199\)](#)
- [Amazon Comprehend examples using SDK for Java 2.x \(p. 210\)](#)
- [DynamoDB examples using SDK for Java 2.x \(p. 214\)](#)
- [Amazon EC2 examples using SDK for Java 2.x \(p. 243\)](#)
- [Amazon EC2 Auto Scaling examples using SDK for Java 2.x \(p. 249\)](#)
- [EventBridge examples using SDK for Java 2.x \(p. 257\)](#)
- [AWS Glue examples using SDK for Java 2.x \(p. 260\)](#)
- [IAM examples using SDK for Java 2.x \(p. 269\)](#)
- [AWS KMS examples using SDK for Java 2.x \(p. 284\)](#)
- [Kinesis examples using SDK for Java 2.x \(p. 290\)](#)
- [Lambda examples using SDK for Java 2.x \(p. 294\)](#)

- [Amazon Personalize examples using SDK for Java 2.x \(p. 300\)](#)
- [Amazon Personalize Events examples using SDK for Java 2.x \(p. 318\)](#)
- [Amazon Personalize Runtime examples using SDK for Java 2.x \(p. 321\)](#)
- [Amazon Pinpoint examples using SDK for Java 2.x \(p. 324\)](#)
- [Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x \(p. 337\)](#)
- [Amazon RDS examples using SDK for Java 2.x \(p. 338\)](#)
- [Amazon Redshift examples using SDK for Java 2.x \(p. 342\)](#)
- [Amazon Rekognition examples using SDK for Java 2.x \(p. 345\)](#)
- [Amazon S3 examples using SDK for Java 2.x \(p. 371\)](#)
- [S3 Glacier examples using SDK for Java 2.x \(p. 397\)](#)
- [Amazon SES examples using SDK for Java 2.x \(p. 404\)](#)
- [Amazon SES API v2 examples using SDK for Java 2.x \(p. 408\)](#)
- [Amazon SNS examples using SDK for Java 2.x \(p. 409\)](#)
- [Amazon SQS examples using SDK for Java 2.x \(p. 425\)](#)
- [Secrets Manager examples using SDK for Java 2.x \(p. 431\)](#)
- [Step Functions examples using SDK for Java 2.x \(p. 434\)](#)
- [Amazon Textract examples using SDK for Java 2.x \(p. 438\)](#)

API Gateway examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with API Gateway.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 184\)](#)

Actions

Create a REST API

The following code example shows how to create an API Gateway REST API.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createAPI( ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
```

```
        .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is "+response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateRestApi](#) in *AWS SDK for Java 2.x API Reference*.

Delete a REST API

The following code example shows how to delete an API Gateway REST API.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteAPI( ApiGatewayClient apiGateway, String restApiId) {

    try {
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()
            .restApiId(restApiId)
            .build();

        apiGateway.deleteRestApi(request);
        System.out.println("The API was successfully deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteRestApi](#) in *AWS SDK for Java 2.x API Reference*.

Delete a deployment

The following code example shows how to delete a deployment.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

    try {
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
            .restApiId(restApiId)
```

```
        .deploymentId(deploymentId)
        .build();

    apiGateway.deleteDeployment(request);
    System.out.println("Deployment was deleted" );

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDeployment](#) in *AWS SDK for Java 2.x API Reference*.

Deploy a REST API

The following code example shows how to deploy an API Gateway REST API.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response = apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is "+response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- For API details, see [CreateDeployment](#) in *AWS SDK for Java 2.x API Reference*.

Application Recovery Controller examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Application Recovery Controller.

Actions are code excerpts that show you how to call individual Application Recovery Controller functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 187\)](#)

Actions

Get the state of a routing control

The following code example shows how to get the state of an Application Recovery Controller routing control.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                                                                String
routingControlArn) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- For API details, see [GetRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

Update the state of a routing control

The following code example shows how to update the state of an Application Recovery Controller routing control.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                                                                String
routingControlArn,
                                                                String
routingControlState) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
```



```
        Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
            .endpointOverride(URI.create(clusterEndpoint.endpoint()))
            .region(Region.of(clusterEndpoint.region()))
            .build();
        return client.updateRoutingControlState(
            UpdateRoutingControlStateRequest.builder()

                .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
    } catch (Exception exception) {
        System.out.println(exception);
    }
}
return null;
}
```

- For API details, see [UpdateRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

CloudFront examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudFront.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 188\)](#)

Actions

Create a function

The following code example shows how to create an Amazon CloudFront function.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {

    try {

        InputStream is = new FileInputStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(is);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
```

```
        .name(functionName)
        .functionCode(functionCode)
        .functionConfig(config)
        .build();

    CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
    return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException | FileNotFoundException e){
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

Update a distribution

The following code example shows how to update an Amazon CloudFront distribution.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void modDistribution(CloudFrontClient cloudFrontClient, String idVal)
{
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to comment
and aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
            .callerReference(config.callerReference())
            .logging(config.logging())
            .originGroups(config.originGroups())
            .origins(config.origins())
            .restrictions(config.restrictions())
            .defaultRootObject(config.defaultRootObject())
            .webACLId(config.webACLId())
            .httpVersion(config.httpVersion())
            .viewerCertificate(config.viewerCertificate())
            .customErrorResponses(config.customErrorResponses())
            .build();
    }
```

```
UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
    .distributionConfig(config1)
    .id(disObject.id())
    .ifMatch(response.eTag())
    .build();

cloudFrontClient.updateDistribution(updateDistributionRequest);

} catch (CloudFrontException e){
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [UpdateDistribution](#) in *AWS SDK for Java 2.x API Reference*.

CloudWatch examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 190\)](#)

Actions

Create an alarm that watches a metric

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
```

```
        .statistic(Statistic.AVERAGE)
        .threshold(70.0)
        .actionsEnabled(false)
        .alarmDescription("Alarm when server CPU utilization exceeds 70%")
        .unit(StandardUnit.SECONDS)
        .dimensions(dimension)
        .build();

        cw.putMetricAlarm(request);
        System.out.printf("Successfully created alarm with name %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Java 2.x API Reference*.

Delete alarms

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Java 2.x API Reference*.

Describe alarms for a metric

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void descCWAlarms( CloudWatchClient cw) {
    try {
```

```
boolean done = false;
String newToken = null;

while(!done) {
    DescribeAlarmsResponse response;
    if (newToken == null) {
        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Java 2.x API Reference*.

Disable alarm actions

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void disableActions(CloudWatchClient cw, String alarmName) {

    try {
        DisableAlarmActionsRequest request = DisableAlarmActionsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.disableAlarmActions(request);
        System.out.printf("Successfully disabled actions on alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

Enable alarm actions

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void enableActions(CloudWatchClient cw, String alarm) {  
  
    try {  
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()  
            .alarmNames(alarm)  
            .build();  
  
        cw.enableAlarmActions(request);  
        System.out.printf("Successfully enabled actions on alarm %s", alarm);  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

List metrics

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listMets( CloudWatchClient cw, String namespace) {  
  
    boolean done = false;  
    String nextToken = null;  
  
    try {  
        while(!done) {  
  
            ListMetricsResponse response;  
            if (nextToken == null) {  
                ListMetricsRequest request = ListMetricsRequest.builder()  
                    .namespace(namespace)  
                    .build();  
  
                response = cw.listMetrics(request);  
            } else {  
                ListMetricsRequest request = ListMetricsRequest.builder()  
                    .namespace(namespace)  
                    .nextToken(nextToken)  
                    .build();  
  
            }  
  
        }  
  
    }  
  
}
```

```
        response = cw.listMetrics(request);
    }

    for (Metric metric : response.metrics()) {
        System.out.printf("Retrieved metric %s", metric.metricName());
        System.out.println();
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for Java 2.x API Reference*.

Put data into a metric

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
        ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension).build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum).build();

        cw.putMetricData(request);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        System.out.printf("Successfully put data point %f", dataPoint);  
    }
```

- For API details, see [PutMetricData](#) in *AWS SDK for Java 2.x API Reference*.

CloudWatch Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Events.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 195\)](#)

Actions

Adding a Lambda function target

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String  
functionArn, String targetId ) {  
  
    try {  
        Target target = Target.builder()  
            .arn(functionArn)  
            .id(targetId)  
            .build();  
  
        PutTargetsRequest request = PutTargetsRequest.builder()  
            .targets(target)  
            .rule(ruleName)  
            .build();  
  
        cwe.putTargets(request);  
        System.out.printf(  
            "Successfully created CloudWatch events target for rule %s",  
            ruleName);  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```


- For API details, see [PutTargets](#) in *AWS SDK for Java 2.x API Reference*.

Create a scheduled rule

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            ruleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

Send events

The following code example shows how to send Amazon CloudWatch Events events.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {
    try {

        final String EVENT_DETAILS =
            "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
```

```
        .entries(requestEntry)
        .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

CloudWatch Logs examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Logs.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 197\)](#)

Actions

Create a subscription filter

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putSubFilters(CloudWatchLogsClient cwl,
                                String filter,
                                String pattern,
                                String logGroup,
                                String functionArn) {

    try {
        PutSubscriptionFilterRequest request =
        PutSubscriptionFilterRequest.builder()
            .filterName(filter)
            .filterPattern(pattern)
            .logGroupName(logGroup)
            .destinationArn(functionArn)
            .build();

        cwl.putSubscriptionFilter(request);
        System.out.printf(
```

```
        "Successfully created CloudWatch logs subscription filter %s",
        filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

Delete a subscription filter

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSubFilter(CloudWatchLogsClient logs, String filter, String
logGroup) {

    try {
        DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
        .filterName(filter)
        .logGroupName(logGroup)
        .build();

        logs.deleteSubscriptionFilter(request);
        System.out.printf("Successfully deleted CloudWatch logs subscription filter
%s", filter);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

Describe existing subscription filters

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {

    try {
        boolean done = false;
        String newToken = null;
```

```
        while(!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                .logGroupName(logGroup)
                .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                .nextToken(newToken)
                .logGroupName(logGroup)
                .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }

            for(SubscriptionFilter filter : response.subscriptionFilters()) {
                System.out.printf("Retrieved filter with name %s, " + "pattern %s "
+ "and destination arn %s",
                    filter.filterName(),
                    filter.filterPattern(),
                    filter.destinationArn());
            }

            if(response.nextToken() == null) {
                done = true;
            } else {
                newToken = response.nextToken();
            }
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Done");
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Cognito Identity Provider examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Cognito Identity Provider.

Actions are code excerpts that show you how to call individual Amazon Cognito Identity Provider functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 200\)](#)

- [Scenarios \(p. 205\)](#)

Actions

Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for Java 2.x API Reference*.

Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
    .session(session)
    .build();

    AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

```
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {

    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [AdminGetUser](#) in *AWS SDK for Java 2.x API Reference*.

List users

The following code example shows how to list Amazon Cognito users.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient, String
userPoolId ) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created " + user.userCreateDate() );
        });

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId ) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus() + " Created " + user.userCreateDate() );
        });

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {

    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Java 2.x API Reference*.

Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {

    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Java 2.x API Reference*.

Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);

    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
```



```
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up ");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SignUp](#) in *AWS SDK for Java 2.x API Reference*.

Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String,String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();

        InitiateAuthResponse response =
identityProviderClient.initiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName() );
        return response;

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Java 2.x API Reference*.

Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development
 * Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
 * cognito_scenario_user_pool_with_mfa.
```

```
*
* This code example performs the following operations:
*
* 1. Invokes the signUp method to sign up a user.
* 2. Invokes the adminGetUser method to get the user's confirmation status.
* 3. Invokes the ResendConfirmationCode method if the user requested another code.
* 4. Invokes the confirmSignUp method.
* 5. Invokes the initiateAuth to sign in. This results in being prompted to set up
TOTP (time-based one-time password). (The response is "ChallengeName": "MFA_SETUP").
* 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
This can be used with Google Authenticator.
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted to
submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/

public class CognitoMVP {

    public static void main(String[] args) throws NoSuchAlgorithmException,
InvalidKeyException {

        final String usage = "\n" +
            "Usage:\n" +
            "  <clientId> <poolId>\n\n" +
            "Where:\n" +
            "  clientId - The app client Id value that you can get from the AWS CDK
script.\n\n" +
            "  poolId - The pool Id that you can get from the AWS CDK script. \n\n" ;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];

        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        // Use the console to get data from the user.
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("*** Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);

        System.out.println("*** Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out.println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
```

```

        resendConfirmationCode(identityProviderClient, clientId, userName);
        System.out.println("*** Sending a new confirmation code");
    }

    System.out.println("*** Enter confirmation code that was emailed");
    String code = in.nextLine();
    confirmSignUp(identityProviderClient, clientId, code, userName);

    System.out.println("*** Rechecking the status of " + userName + " in the user
pool");
    getAdminUser(identityProviderClient, userName, poolId);

    InitiateAuthResponse authResponse = initiateAuth(identityProviderClient,
clientId, userName, password) ;
    String mySession = authResponse.session() ;
    String newSession = getSecretForAppMFA(identityProviderClient, mySession);

    System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
    String myCode = in.nextLine();

    // Verify the TOTP and register for MFA.
    verifyTOTP(identityProviderClient, newSession, myCode);
    System.out.println("*** Re-enter a 6-digit code displayed in Google
Authenticator");
    String mfaCode = in.nextLine();
    InitiateAuthResponse authResponse1 = initiateAuth(identityProviderClient,
clientId, userName, password);
    String session2 = authResponse1.session();
    adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
}

// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {

    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)

```

```
        .session(session)
        .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String,String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();

        InitiateAuthResponse response =
identityProviderClient.initiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName() );
        return response;

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}

public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();
```

```
        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {

    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);

    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {

    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
```

```
        .userPoolId(poolId)
        .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

        } catch (CognitoIdentityProviderException e){
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Amazon Comprehend examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Comprehend.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 210\)](#)

Actions

Create a document classifier

The following code example shows how to create an Amazon Comprehend document classifier.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri, String documentClassifierName){

    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient.createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDocumentClassifier](#) in *AWS SDK for Java 2.x API Reference*.

Detect entities in a document

The following code example shows how to detect entities in a document with Amazon Comprehend.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectAllEntities(ComprehendClient comClient,String text ) {

    try {
        DetectEntitiesRequest detectEntitiesRequest =
DetectEntitiesRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectEntitiesResponse detectEntitiesResult =
comClient.detectEntities(detectEntitiesRequest);
        List<Entity> entList = detectEntitiesResult.entities();
        for (Entity entity : entList) {
            System.out.println("Entity text is " + entity.text());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}
```

- For API details, see [DetectEntities](#) in *AWS SDK for Java 2.x API Reference*.

Detect key phrases in a document

The following code example shows how to detect key phrases in a document with Amazon Comprehend.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectAllKeyPhrases(ComprehendClient comClient, String text) {  
  
    try {  
        DetectKeyPhrasesRequest detectKeyPhrasesRequest =  
DetectKeyPhrasesRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectKeyPhrasesResponse detectKeyPhrasesResult =  
comClient.detectKeyPhrases(detectKeyPhrasesRequest);  
        List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();  
        for (KeyPhrase keyPhrase : phraseList) {  
            System.out.println("Key phrase text is " + keyPhrase.text());  
        }  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for Java 2.x API Reference*.

Detect syntactical elements of a document

The following code example shows how to detect syntactical elements of a document with Amazon Comprehend.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectAllSyntax(ComprehendClient comClient, String text){  
  
    try {  
        DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectSyntaxResponse detectSyntaxResult =  
comClient.detectSyntax(detectSyntaxRequest);  
  
    }  
}
```

```
List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
for (SyntaxToken token : syntaxTokens) {
    System.out.println("Language is " + token.text());
    System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
}

} catch (ComprehendException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DetectSyntax](#) in *AWS SDK for Java 2.x API Reference*.

Detect the dominant language in a document

The following code example shows how to detect the dominant language in a document with Amazon Comprehend.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectTheDominantLanguage(ComprehendClient comClient, String
text){

    try {
        DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
            .text(text)
            .build();

        DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
        List<DominantLanguage> allLanList = resp.languages();
        for (DominantLanguage lang : allLanList) {
            System.out.println("Language is " + lang.languageCode());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectDominantLanguage](#) in *AWS SDK for Java 2.x API Reference*.

Detect the sentiment of a document

The following code example shows how to detect the sentiment of a document with Amazon Comprehend.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectSentiments(ComprehendClient comClient, String text){  
  
    try {  
        DetectSentimentRequest detectSentimentRequest =  
DetectSentimentRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectSentimentResponse detectSentimentResult =  
comClient.detectSentiment(detectSentimentRequest);  
        System.out.println("The Neutral value is "  
+detectSentimentResult.sentimentScore().neutral() );  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectSentiment](#) in *AWS SDK for Java 2.x API Reference*.

DynamoDB examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with DynamoDB.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 214\)](#)
- [Scenarios \(p. 224\)](#)

Actions

Create a table

The following code example shows how to create a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)  
{  
    DynamoDbWaiter dbWaiter = ddb.waiter();  
    CreateTableRequest request = CreateTableRequest.builder()  
        .attributeDefinitions(AttributeDefinition.builder()  

```

```
        .attributeName(key)
        .attributeType(ScalarAttributeType.S)
        .build()
    .keySchema(KeySchemaElement.builder()
        .attributeName(key)
        .keyType(KeyType.HASH)
        .build())
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .tableName(tableName)
    .build();

String newTable = "";
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateTable](#) in *AWS SDK for Java 2.x API Reference*.

Delete a table

The following code example shows how to delete a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for Java 2.x API Reference*.

Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for Java 2.x API Reference*.

Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Gets an item from a table by using the `DynamoDbClient`.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal, String partitionAlias) {

    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String,String> attrNameAlias = new HashMap<String,String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();

    attrValues.put(":"+partitionKeyName, AttributeValue.builder()
        .s(partitionKeyVal)
```

```
        .build());

    QueryRequest queryReq = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
        .expressionAttributeNames(attrNameAlias)
        .expressionAttributeValues(attrValues)
        .build();

    try {
        QueryResponse response = ddb.query(queryReq);
        return response.count();
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return -1;
}
```

- For API details, see [GetItem](#) in *AWS SDK for Java 2.x API Reference*.

Get information about a table

The following code example shows how to get information about a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {

    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo = ddb.describeTable(request).table();
        if (tableInfo != null) {
            System.out.format("Table name   : %s\n", tableInfo.tableName());
            System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
            System.out.format("Status      : %s\n", tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits().longValue());
            System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits().longValue());

            List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
            System.out.println("Attributes");

            for (AttributeDefinition a : attributes) {
```

```
        System.out.format(" %s (%s)\n", a.attributeName(),
a.attributeType());
    }
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for Java 2.x API Reference*.

List tables

The following code example shows how to list DynamoDB tables.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();
            if (tableNames.size() > 0) {
                for (String curName : tableNames) {
                    System.out.format("* %s\n", curName);
                }
            } else {
                System.out.println("No tables found!");
                System.exit(0);
            }

            lastName = response.lastEvaluatedTableName();
            if (lastName == null) {
                moreTables = false;
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
    System.out.println("\nDone!");
}
```

- For API details, see [ListTables](#) in *AWS SDK for Java 2.x API Reference*.

Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Puts an item into a table by using the enhanced client.

```
public static void putRecord(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));

        // Create an Instant value.
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        // Populate the Table.
        Customer custRecord = new Customer();
        custRecord.setCustName("Tom red");
        custRecord.setId("id101");
        custRecord.setEmail("tred@noserver.com");
        custRecord.setRegistrationDate(instant);

        // Put the customer data into an Amazon DynamoDB table.
        custTable.putItem(custRecord);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Customer data added to the table with id id101");
}
```

- For API details, see [PutItem](#) in *AWS SDK for Java 2.x API Reference*.

Query a table

The following code example shows how to query a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Queries a table by using the enhanced client.

```
public static String queryTable(DynamoDbEnhancedClient enhancedClient) {

    try{
        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));
```



```

        QueryConditional queryConditional =
QueryConditional.keyEqualTo(Key.builder()
        .partitionValue("id101")
        .build());

        // Get items in the table and write out the ID value.
        Iterator<Customer> results =
mappedTable.query(queryConditional).items().iterator();
        String result="";

        while (results.hasNext()) {
            Customer rec = results.next();
            result = rec.getId();
            System.out.println("The record id is "+result);
        }
        return result;

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

Queries a table by using the enhanced client and a secondary index.

```

public static void queryIndex(DynamoDbClient ddb, String tableName) {

    try {
        // Create a DynamoDbEnhancedClient and use the DynamoDbClient object.
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        //Create a DynamoDbTable object based on Movies.
        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        String dateVal = "2013";

        DynamoDbIndex<Movies> secIndex = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class)) .index("year-index");
        AttributeValue attVal = AttributeValue.builder()
            .n(dateVal)
            .build();

        // Create a QueryConditional object that's used in the query operation.
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder().partitionValue(attVal)
            .build());

        // Get items in the table.
        SdkIterable<Page<Movies>> results =
secIndex.query(QueryEnhancedRequest.builder()
            .queryConditional(queryConditional)
            .limit(300)
            .build());

        // Display the results.
        results.forEach(page -> {
            List<Movies> allMovies = page.items();
            for (Movies myMovies: allMovies) {
                System.out.println("The movie title is " + myMovies.getTitle() + ".
The year is " + myMovies.getYear());
            }
        }
    }
}

```

```
    });  
  
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Queries a table by using the `DynamoDbClient`.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String  
partitionKeyName, String partitionKeyVal, String partitionAlias) {  
  
    // Set up an alias for the partition key name in case it's a reserved word.  
    HashMap<String,String> attrNameAlias = new HashMap<String,String>();  
    attrNameAlias.put(partitionAlias, partitionKeyName);  
  
    // Set up mapping of the partition name with the value.  
    HashMap<String, AttributeValue> attrValues = new HashMap<>();  
  
    attrValues.put(":"+partitionKeyName, AttributeValue.builder()  
        .s(partitionKeyVal)  
        .build());  
  
    QueryRequest queryReq = QueryRequest.builder()  
        .tableName(tableName)  
        .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)  
        .expressionAttributeNames(attrNameAlias)  
        .expressionAttributeValues(attrValues)  
        .build();  
  
    try {  
        QueryResponse response = ddb.query(queryReq);  
        return response.count();  
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    return -1;  
}
```

Queries a table by using the `DynamoDbClient` and a secondary index.

```
public static void queryIndex(DynamoDbClient ddb, String tableName) {  
  
    try {  
        Map<String, String> expressionAttributeNames = new HashMap<>();  
        expressionAttributeNames.put("#year", "year");  
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();  
        expressionAttributeValues.put(":yearValue",  
AttributeValue.builder().n("2013").build());  
  
        QueryRequest request = QueryRequest.builder()  
            .tableName(tableName)  
            .indexName("year-index")  
            .keyConditionExpression("#year = :yearValue")  
            .expressionAttributeNames(expressionAttributeNames)  
            .expressionAttributeValues(expressionAttributeValues)  
            .build();  
  
    }  
}
```

```
        System.out.println("=== Movie Titles ===");
        QueryResponse response = ddb.query(request);
        response.items()
            .forEach(movie -> System.out.println(movie.get("title").s()));

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Query](#) in *AWS SDK for Java 2.x API Reference*.

Scan a table

The following code example shows how to scan a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Scans an Amazon DynamoDB table by using the enhanced client.

```
public static void scan( DynamoDbEnhancedClient enhancedClient) {
    try{
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
            TableSchema.fromBean(Customer.class));
        Iterator<Customer> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Customer rec = results.next();
            System.out.println("The record id is "+rec.getId());
            System.out.println("The name is " +rec.getCustName());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [Scan](#) in *AWS SDK for Java 2.x API Reference*.

Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Updates an item located in a table by using the enhanced client.

```
public static String modifyItem(DynamoDbEnhancedClient enhancedClient, String
keyVal, String email) {
```

```
try {
    DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
    Key key = Key.builder()
        .partitionValue(keyVal)
        .build();

    // Get the item by using the key and update the email value.
    Customer customerRec = mappedTable.getItem(r->r.key(key));
    customerRec.setEmail(email);
    mappedTable.updateItem(customerRec);
    return customerRec.getEmail();

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
```

- For API details, see [UpdateItem](#) in *AWS SDK for Java 2.x API Reference*.

Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Inserts many items into a table by using the enhanced client.

```
public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
        DynamoDbTable<Music> musicMappedTable = enhancedClient.table("Music",
TableSchema.fromBean(Music.class));
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        Customer record2 = new Customer();
        record2.setCustName("Fred Pink");
        record2.setId("id110");
        record2.setEmail("fredp@noserver.com");
        record2.setRegistrationDate(instant);

        Customer record3 = new Customer();
        record3.setCustName("Susan Pink");
        record3.setId("id120");
        record3.setEmail("spink@noserver.com");
        record3.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =
BatchWriteItemEnhancedRequest.builder()
            .writeBatches(
                WriteBatch.builder(Customer.class) // add items to
the Customer table
```

```
        .mappedTableResource(customerMappedTable)
        .addPutItem(builder -> builder.item(record2))
        .addPutItem(builder -> builder.item(record3))
        .build(),
        WriteBatch.builder(Music.class)           // delete an
item from the Music table
        .mappedTableResource(musicMappedTable)
        .addDeleteItem(builder -> builder.key(
            Key.builder().partitionValue("Famous
Band").build()))
        .build();

// Add two items to the Customer table and delete one item from the Music
table
enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

System.out.println("done");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create a DynamoDB table.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {

    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
```

```

        .attributeType("N")
        .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

```

Create a helper function to download and extract the sample JSON file.

```

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {

    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

```

```
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    while (iter.hasNext()) {

        // Only add 200 Movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}
```

Get an item from a table.

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

Full example.

```
public class Scenario {  
  
    public static void main(String[] args) throws IOException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <fileName>\n\n" +  
            "Where:\n" +  
            "  fileName - The path to the moviedata.json file that you can download  
from the Amazon DynamoDB Developer Guide.\n" ;  
  
        if (args.length != 1) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String tableName = "Movies";  
        String fileName = args[0];  
        ProfileCredentialsProvider credentialsProvider =  
ProfileCredentialsProvider.create();  
        Region region = Region.US_EAST_1;  
        DynamoDbClient ddb = DynamoDbClient.builder()  
            .region(region)  
            .credentialsProvider(credentialsProvider)  
            .build();  
  
        System.out.println("***** Creating an Amazon DynamoDB table named Movies with  
a key named year and a sort key named title.");  
        createTable(ddb, tableName);  
  
        System.out.println("***** Loading data into the Amazon DynamoDB table.");  
        loadData(ddb, tableName, fileName);  
  
        System.out.println("***** Getting data from the Movie table.");  
        getItem(ddb) ;  
  
        System.out.println("***** Putting a record into the Amazon DynamoDB table.");  
        putRecord(ddb);  
  
        System.out.println("***** Updating a record.");  
        updateTableItem(ddb, tableName);  
  
        System.out.println("***** Scanning the Amazon DynamoDB table.");  
        scanMovies(ddb, tableName);  
  
        System.out.println("***** Querying the Movies released in 2013.");  
        queryTable(ddb);  
  
        System.out.println("***** Deleting the Amazon DynamoDB table.");  
        deleteDynamoDBTable(ddb, tableName);  
        ddb.close();  
    }  
  
    // Create a table with a Sort key.  
    public static void createTable(DynamoDbClient ddb, String tableName) {  
  
        DynamoDbWaiter dbWaiter = ddb.waiter();  
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();
```



```

// Define attributes.
attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("year")
    .attributeType("N")
    .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()

```

```
        .partitionValue(2013)
        .build());

    // Get items in the table and write out the ID value.
    Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
    String result="";

    while (results.hasNext()) {
        Movies rec = results.next();
        System.out.println("The title of the movie is "+rec.getTitle());
        System.out.println("The movie information is "+rec.getInfo());
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {

    System.out.println("***** Scanning all movies.\n");
    try{
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is "+rec.getTitle());
            System.out.println("The movie year is " +rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {

    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    while (iter.hasNext()) {

        // Only add 200 Movies to the table.
        if (t == 200)
            break ;
    }
}
```

```

        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName){

    HashMap<String,AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String,AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s({"\directors\":["Merian C. Cooper\"",
"\Ernest B. Schoedsack\"]').build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println(tableName + " was successfully deleted!");
}

```

```

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {
    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [BatchWriteItem](#)

- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class ScenarioPartiQLBatch {

    public static void main(String [] args) throws IOException {

        String tableName = "MoviesPartiQLBatch";
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .credentialsProvider(credentialsProvider)
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named "+tableName
+" with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the "+ tableName + "
table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("***** Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
```

```

DynamoDbWaiter dbWaiter = ddb.waiter();
ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

// Define attributes.
attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("year")
    .attributeType("N")
    .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE) // Sort
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void putRecordBatch(DynamoDbClient ddb) {
    String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        // Create three movies to add to the Amazon DynamoDB table.
        // Set data for Movie 1.
        List<AttributeValue> parameters = new ArrayList<>();

```

```
AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s("My Movie 1")
    .build();

AttributeValue att3 = AttributeValue.builder()
    .s("No Information")
    .build();

parameters.add(att1);
parameters.add(att2);
parameters.add(att3);

BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parameters)
    .build();

// Set data for Movie 2.
List<AttributeValue> parametersMovie2 = new ArrayList<>();
AttributeValue attMovie2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie2A = AttributeValue.builder()
    .s("My Movie 2")
    .build();

AttributeValue attMovie2B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie2.add(attMovie2);
parametersMovie2.add(attMovie2A);
parametersMovie2.add(attMovie2B);

BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie2)
    .build();

// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie3.add(attMovie3);
parametersMovie3.add(attMovie3A);
parametersMovie3.add(attMovie3B);

BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
```

```

        .statement(sqlStatement)
        .parameters(parametersMovie3)
        .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestMovie1);
myBatchStatementList.add(statementRequestMovie2);
myBatchStatementList.add(statementRequestMovie3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
System.out.println("ExecuteStatement successful: "+ response.toString());
System.out.println("Added new movies using a batch command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb){
    String sqlStatement = "UPDATE MoviesParti@Batch SET info = 'directors\"";
[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

// Update three records.
AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s("My Movie 1")
    .build();

parametersRec1.add(att1);
parametersRec1.add(att2);

BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec1)
    .build();

// Update record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Update record 3.

```



```
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
    System.out.println("ExecuteStatement successful: "+ response.toString());
    System.out.println("Updated three movies using a batch command.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Item was updated!");
}

public static void deleteItemBatch(DynamoDbClient ddb){
    String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Specify three records to delete.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
        .build();

    // Specify record 2.
    List<AttributeValue> parametersRec2 = new ArrayList<>();
    AttributeValue attRec2 = AttributeValue.builder()
        .n(String.valueOf("2022"))
```

```

        .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
        BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}

```

```
private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class ScenarioPartiQ {

    public static void main(String [] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "  <fileName>\n\n" +
            "Where:\n" +
            "  fileName - The path to the moviedata.json file that you can download
from the Amazon DynamoDB Developer Guide.\n" ;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .credentialsProvider(credentialsProvider)
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named MoviesPartiQ
with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Loading data into the MoviesPartiQ table.");
    }
}
```

```

loadData(ddb, fileName);

System.out.println("***** Getting data from the MoviesPartiQ table.");
getItem(ddb);

System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.

```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " +newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws IOException
{
    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s(title)
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s(info)
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        // Insert the movie into the Amazon DynamoDB table.
        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added Movie " +title);

        parameters.remove(att1);
        parameters.remove(att2);
        parameters.remove(att3);
        t++;
    }
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
```

```

List<AttributeValue> parameters = new ArrayList<>();
AttributeValue att1 = AttributeValue.builder()
    .n("2012")
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s("The Perks of Being a Wallflower")
    .build();

parameters.add(att1);
parameters.add(att2);

try {
    ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
    System.out.println("ExecuteStatement successful: "+ response.toString());

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

}

public static void updateTableItem(DynamoDbClient ddb){

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")

```

```

        .build());

parameters.add(att1);
parameters.add(att2);

try {
    executeStatementRequest(ddb, sqlStatement, parameters);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: "+ response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName +" was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse executeStatementResult)
{

```

```
        System.out.println("ExecuteStatement successful: "+
executeStatementResult.toString());
    }
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

Amazon EC2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2.

Actions are code excerpts that show you how to call individual Amazon EC2 functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon EC2 functions.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 243\)](#)

Actions

Allocate an Elastic IP address

The following code example shows how to allocate an Elastic IP address for Amazon EC2.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String getAllocateAddress( Ec2Client ec2, String instanceId ) {
    try {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        String allocationId = allocateResponse.allocationId();
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
        return "";  
    }
```

- For API details, see [AllocateAddress](#) in *AWS SDK for Java 2.x API Reference*.

Create a security group

The following code example shows how to create an Amazon EC2 security group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createEC2SecurityGroup( Ec2Client ec2,String groupName, String  
groupDesc, String vpcId) {  
    try {  
  
        CreateSecurityGroupRequest createRequest =  
CreateSecurityGroupRequest.builder()  
            .groupName(groupName)  
            .description(groupDesc)  
            .vpcId(vpcId)  
            .build();  
  
        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);  
  
        IpRange ipRange = IpRange.builder()  
            .cidrIp("0.0.0.0/0").build();  
  
        IpPermission ipPerm = IpPermission.builder()  
            .ipProtocol("tcp")  
            .toPort(80)  
            .fromPort(80)  
            .ipRanges(ipRange)  
            .build();  
  
        IpPermission ipPerm2 = IpPermission.builder()  
            .ipProtocol("tcp")  
            .toPort(22)  
            .fromPort(22)  
            .ipRanges(ipRange)  
            .build();  
  
        AuthorizeSecurityGroupIngressRequest authRequest =  
            AuthorizeSecurityGroupIngressRequest.builder()  
                .groupName(groupName)  
                .ipPermissions(ipPerm, ipPerm2)  
                .build();  
  
        AuthorizeSecurityGroupIngressResponse authResponse =  
            ec2.authorizeSecurityGroupIngress(authRequest);  
        System.out.printf("Successfully added ingress policy to Security Group %s",  
groupName);  
        return resp.groupId();  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createEC2KeyPair(Ec2Client ec2,String keyName ) {
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        ec2.createKeyPair(request);
        System.out.printf("Successfully created key pair named %s", keyName);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

Create and run an instance

The following code example shows how to create and run an Amazon EC2 instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createEC2Instance(Ec2Client ec2,String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
    }
```

```
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf( "Successfully started EC2 Instance %s based on AMI %s",
instanceId, amiId);
        return instanceId;

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- For API details, see [RunInstances](#) in *AWS SDK for Java 2.x API Reference*.

Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf("Successfully deleted Security Group with id %s",
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {

    try {
```

```
DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
    .keyName(keyPair)
    .build();

ec2.deleteKeyPair(request);
System.out.printf("Successfully deleted key pair named %s", keyPair);

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

Release an Elastic IP address

The following code example shows how to release an Elastic IP address.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void releaseEC2Address(Ec2Client ec2,String allocId) {

    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.printf("Successfully released elastic IP address %s", allocId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for Java 2.x API Reference*.

Start an instance

The following code example shows how to start an Amazon EC2 instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
}
```

```
        System.out.printf("Successfully started instance %s", instanceId);  
    }
```

- For API details, see [StartInstances](#) in *AWS SDK for Java 2.x API Reference*.

Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {  
  
    StopInstancesRequest request = StopInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    ec2.stopInstances(request);  
    System.out.printf("Successfully stopped instance %s", instanceId);  
}
```

- For API details, see [StopInstances](#) in *AWS SDK for Java 2.x API Reference*.

Terminate an instance

The following code example shows how to terminate an Amazon EC2 instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void terminateEC2( Ec2Client ec2, String instanceID) {  
  
    try{  
        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()  
            .instanceIds(instanceID)  
            .build();  
  
        TerminateInstancesResponse response = ec2.terminateInstances(ti);  
        List<InstanceStateChange> list = response.terminatingInstances();  
        for (InstanceStateChange sc : list) {  
            System.out.println("The ID of the terminated instance is " +  
sc.instanceId());  
        }  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [TerminateInstances](#) in *AWS SDK for Java 2.x API Reference*.

Amazon EC2 Auto Scaling examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2 Auto Scaling.

Actions are code excerpts that show you how to call individual Amazon EC2 Auto Scaling functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon EC2 Auto Scaling functions.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 249\)](#)
- [Scenarios \(p. 251\)](#)

Actions

Create a group

The following code example shows how to create an Auto Scaling group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                          String groupName,
                                          String launchTemplateName,
                                          String serviceLinkedRoleARN,
                                          String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(launchTemplateName)
        .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .availabilityZones("us-east-1a")
        .launchTemplate(templateSpecification)
        .maxSize(1)
        .minSize(1)
        .vpcZoneIdentifier(vpcZoneId)
        .serviceLinkedRoleARN(serviceLinkedRoleARN)
        .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
    }
```

```
        System.out.println("Auto Scaling Group created");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

Delete a group

The following code example shows how to delete an Auto Scaling group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build() ;

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
        System.out.println("You successfully deleted "+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

Get information about groups

The following code example shows how to get information about Auto Scaling groups.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String getAutoScaling( AutoScalingClient autoScalingClient, String
groupName) {
    try{
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
```

```
List<AutoScalingGroup> groups = response.autoScalingGroups();
for (AutoScalingGroup group: groups) {
    System.out.println("The group name is " +
group.autoScalingGroupName());
    System.out.println("The group ARN is " + group.autoScalingGroupARN());

    List<Instance> instances = group.instances();
    for (Instance instance : instances) {
        instanceId = instance.instanceId();
    }
    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class AutoScalingScenario {

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>\n" +
            "\n" +
            "Where:\n" +
            "    groupName - The name of the Auto Scaling group.\n" +
            "    launchTemplateName - The name of the launch template. \n" +
            "    serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
            linked role that the Auto Scaling group uses.\n" +
            "    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
            instances in the Auto Scaling group can be created.\n" ;
    }
}
```



```
if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String groupName = args[0];
String launchTemplateName = args[1];
String serviceLinkedRoleARN = args[2];
String vpcZoneId = args[3];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

System.out.println("**** Create an Auto Scaling group named "+groupName);
createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
serviceLinkedRoleARN, vpcZoneId);

System.out.println("Wait 1 min for the resources, including the instance.
Otherwise, an empty instance Id is returned");
Thread.sleep(60000);

System.out.println("**** Get Auto Scale group Id value");
String instanceId = getSpecificAutoScalingGroups(autoScalingClient, groupName);
if (instanceId.compareTo("") ==0) {
    System.out.println("Error - no instance Id value");
    System.exit(1);
} else {
    System.out.println("The instance Id value is "+instanceId);
}

System.out.println("**** Describe Auto Scaling with the Id value "+instanceId);
describeAutoScalingInstance( autoScalingClient, instanceId);

System.out.println("**** Enable metrics collection "+instanceId);
enableMetricsCollection(autoScalingClient, groupName);

System.out.println("**** Update an Auto Scaling group to update max size to
3");
updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
serviceLinkedRoleARN);

System.out.println("**** Describe all Auto Scaling groups to show the current
state of the groups");
describeAutoScalingGroups(autoScalingClient, groupName);

System.out.println("**** Describe account details");
describeAccountLimits(autoScalingClient);

System.out.println("Wait 1 min for the resources, including the instance.
Otherwise, an empty instance Id is returned");
Thread.sleep(60000);

System.out.println("**** Set desired capacity to 2");
setDesiredCapacity(autoScalingClient, groupName);

System.out.println("**** Get the two instance Id values and state");
getSpecificAutoScalingGroups(autoScalingClient, groupName);

System.out.println("**** List the scaling activities that have occurred for the
group");
describeScalingActivities(autoScalingClient, groupName);

System.out.println("**** Terminate an instance in the Auto Scaling group");
terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
```

```
        System.out.println("**** Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);

        System.out.println("**** Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient autoScalingClient,
        String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
                DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response =
                autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity: activities) {
                System.out.println("The activity Id is "+activity.activityId());
                System.out.println("The activity details are "+activity.details());
            }
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient, String
        groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
                SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
        String groupName,
        String launchTemplateName,
        String serviceLinkedRoleARN,
        String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
                LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
                CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
```

```
.launchTemplate(templateSpecification)
.maxSize(1)
.minSize(1)
.vpcZoneIdentifier(vpcZoneId)
.serviceLinkedRoleARN(serviceLinkedRoleARN)
.build();

autoScalingClient.createAutoScalingGroup(request);
DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Auto Scaling Group created");

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void describeAutoScalingInstance( AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
DescribeAutoScalingInstancesRequest.builder()
            .instanceIds(id)
            .build();

        DescribeAutoScalingInstancesResponse response =
autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance:instances ) {
            System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("*** The service to use for the health checks: "+
group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```

        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try{
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " + group.autoScalingGroupARN());
            List<Instance> instances = group.instances();

            for (Instance instance : instances) {
                instanceId = instance.instanceId();
                System.out.println("The instance id is " + instanceId);
                System.out.println("The lifecycle state is "
+instance.lifecycleState());
            }
        }

        return instanceId ;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {

        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
        .autoScalingGroupName(groupName)
        .metrics("GroupMaxSize")
        .granularity("1Minute")
        .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
        .autoScalingGroupName(groupName)
        .metrics("GroupMaxSize")

```

```

        .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
    System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is
"+response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is
"+response.numberofAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
"+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId){
    try {

```

```
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance "+instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build() ;

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
        System.out.println("You successfully deleted "+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

EventBridge examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with EventBridge.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 258\)](#)

Actions

Create a scheduled rule

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is "+
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

Delete a scheduled rule

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteEBRule(EventBridgeClient eventBrClient, String ruleName) {

    try {
        DisableRuleRequest disableRuleRequest = DisableRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .build();

        eventBrClient.disableRule(disableRuleRequest);
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
            .name(ruleName)
```

```
        .eventBusName("default")
        .build();

eventBrClient.deleteRule(ruleRequest);
System.out.println("Rule " +ruleName + " was successfully deleted!");

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteRule](#) in *AWS SDK for Java 2.x API Reference*.

Send events

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void putEEvents(EventBridgeClient eventBrClient, String resourceArn,
String resourceArn2 ) {

    try {
        // Populate a List with the resource ARN values.
        List<String> resources = new ArrayList<>();
        resources.add(resourceArn);
        resources.add(resourceArn2);

        PutEventsRequestEntry reqEntry = PutEventsRequestEntry.builder()
            .resources(resources)
            .source("com.mycompany.myapp")
            .detailType("myDetailType")
            .detail("{ \"key1\": \"value1\", \"key2\": \"value2\" }")
            .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
            .entries(reqEntry)
            .build();

        PutEventsResponse result = eventBrClient.putEvents(eventsRequest);
        for (PutEventsResultEntry resultEntry : result.entries()) {
            if (resultEntry.eventId() != null) {
                System.out.println("Event Id: " + resultEntry.eventId());
            } else {
                System.out.println("Injection failed with Error Code: " +
resultEntry.errorCode());
            }
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

AWS Glue examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Glue.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 260\)](#)
- [Scenarios \(p. 263\)](#)

Actions

Create a crawler

The following code example shows how to create an AWS Glue crawler.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createGlueCrawler(GlueClient glueClient,
                                     String iam,
                                     String s3Path,
                                     String cron,
                                     String dbName,
                                     String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");
    }
}
```

```
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Java 2.x API Reference*.

Get a crawler

The following code example shows how to get an AWS Glue crawler.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                .withLocale( Locale.US )
                .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the Crawler is " + createDate );
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetCrawler](#) in *AWS SDK for Java 2.x API Reference*.

Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getSpecificDatabase(GlueClient glueClient, String databaseName)
{
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
```

```
        .name(databaseName)
        .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for Java 2.x API Reference*.

Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName ) {

    try {
        GetTableRequest tableRequest = GetTableRequest.builder()
            .databaseName(dbName)
            .name(tableName)
            .build();

        GetTableResponse tableResponse = glueClient.getTable(tableRequest);
        Instant createDate = tableResponse.table().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
            .withLocale( Locale.US)
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the table is " + createDate );

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetTables](#) in *AWS SDK for Java 2.x API Reference*.

Start a crawler

The following code example shows how to start an AWS Glue crawler.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void startSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class GlueScenario {

    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "  <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> \n\n" +
            "Where:\n" +
```

```

        "    iam - The ARN of the IAM role that has AWS Glue and S3 permissions.
\n" +
        "    s3Path - The Amazon Simple Storage Service (Amazon S3) target that
contains data (for example, CSV data).\n" +
        "    cron - A cron expression used to specify the schedule (i.e., cron(15
12 * * ? *).\n" +
        "    dbName - The database name. \n" +
        "    crawlerName - The name of the crawler. \n" +
        "    jobName - The name you assign to this job definition."+
        "    scriptLocation - The Amazon S3 path to a script that runs a job." +
        "    locationUri - The location of the database" ;

    if (args.length != 8) {
        System.out.println(usage);
        System.exit(1);
    }

    String iam = args[0];
    String s3Path = args[1];
    String cron = args[2];
    String dbName = args[3];
    String crawlerName = args[4];
    String jobName = args[5];
    String scriptLocation = args[6];
    String locationUri = args[7];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println("About to start the AWS Glue Scenario");
    createDatabase(glueClient, dbName, locationUri);
    createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
    getSpecificCrawler(glueClient, crawlerName);
    startSpecificCrawler(glueClient, crawlerName);
    getSpecificDatabase(glueClient, dbName);
    getGlueTables(glueClient, dbName);
    createJob(glueClient, jobName, iam, scriptLocation);
    startJob(glueClient, jobName);
    getAllJobs(glueClient);
    getJobRuns(glueClient, jobName);
    deleteJob(glueClient, jobName);
    System.out.println("*** Wait 5 MIN for the "+crawlerName+" to stop");
    TimeUnit.MINUTES.sleep(5);
    deleteDatabase(glueClient, dbName);
    deleteSpecificCrawler(glueClient, crawlerName);
    System.out.println("Successfully completed the AWS Glue Scenario");
}

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri ) {

        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println("The database was successfully created");

```

```

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createGlueCrawler(GlueClient glueClient,
                                    String iam,
                                    String s3Path,
                                    String cron,
                                    String dbName,
                                    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {

    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
            .withLocale( Locale.US)
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the Crawler is " + createDate );

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

```
public static void startSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String databaseName)
{
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
            .withLocale( Locale.US)
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the database is " + createDate );
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getGlueTables(GlueClient glueClient, String dbName){
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        for (Table table: tables) {
            System.out.println("Table name is: "+table.name());
        }
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void startJob(GlueClient glueClient, String jobName) {
    try {
        StartJobRunRequest runRequest = StartJobRunRequest.builder()
```

```
        .workerType(WorkerType.G_1_X)
        .numberOfWorkers(10)
        .jobName(jobName)
        .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is "+
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {

    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("MyJob1")
            .scriptLocation(scriptLocation)
            .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName +" was successfully created.");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {

    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job: jobs) {
            System.out.println("Job name is : "+job.name());
            System.out.println("The job worker type is :
"+job.workerType().name());
        }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {
```



```
try {
    GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
        .jobName(jobName)
        .maxResults(20)
        .build();

    GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
    List<JobRun> jobRuns = response.jobRuns();
    for (JobRun jobRun: jobRuns) {
        System.out.println("Job run state is "+jobRun.jobRunState().name());
        System.out.println("Job run Id is "+jobRun.id());
        System.out.println("The Glue version is "+jobRun.glueVersion());
    }
} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void deleteJob(GlueClient glueClient, String jobName) {

    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {

    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

}

public static void deleteSpecificCrawler(GlueClient glueClient, String crawlerName)
{

    try {
        DeleteCrawlerRequest deleteCrawlerRequest = DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

IAM examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with IAM.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 269\)](#)
- [Scenarios \(p. 279\)](#)

Actions

Attach a policy to a role

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {
    try {
```

```
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
    .roleName(roleName)
    .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy: attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn)==0) {
        System.out.println(roleName + " policy is already attached to this
role.");
        return;
    }
}

        AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
" to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Java 2.x API Reference*.

Create a policy

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument)
            .build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
    }
```

```
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Java 2.x API Reference*.

Create a role

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Java 2.x API Reference*.

Create a user

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Java 2.x API Reference*.

Create an access key

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createIAMAccessKey(IamClient iam,String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

Create an alias for an account

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createIAMAccountAlias(IamClient iam, String alias) {  
  
    try {  
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()  
            .accountAlias(alias)  
            .build();  
  
        iam.createAccountAlias(request);  
        System.out.println("Successfully created account alias: " + alias);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

Delete a policy

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteIAMPolicy(IamClient iam, String policyARN) {  
  
    try {  
        DeletePolicyRequest request = DeletePolicyRequest.builder()  
            .policyArn(policyARN)  
            .build();  
  
        iam.deletePolicy(request);  
        System.out.println("Successfully deleted the policy");  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Java 2.x API Reference*.

Delete a user

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteIAMUser(IamClient iam, String userName) {  
  
    try {  
        DeleteUserRequest request = DeleteUserRequest.builder()  
            .userName(userName)  
            .build();  
  
        iam.deleteUser(request);  
        System.out.println("Successfully deleted IAM user " + userName);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Java 2.x API Reference*.

Delete an access key

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {  
  
    try {  
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()  
            .accessKeyId(accessKey)  
            .userName(username)  
            .build();  
  
        iam.deleteAccessKey(request);  
        System.out.println("Successfully deleted access key " + accessKey +  
            " from user " + username);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

Delete an account alias

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteIAMAccountAlias(IamClient iam, String alias ) {  
  
    try {  
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()  
            .accountAlias(alias)  
            .build();  
  
        iam.deleteAccountAlias(request);  
        System.out.println("Successfully deleted account alias " + alias);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

Detach a policy from a role

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )  
{  
  
    try {  
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()  
            .roleName(roleName)  
            .policyArn(policyArn)  
            .build();  
  
        iam.detachRolePolicy(request);  
        System.out.println("Successfully detached policy " + policyArn +  
            " from role " + roleName);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Java 2.x API Reference*.

List a user's access keys

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listKeys( IamClient iam,String userName ){
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Java 2.x API Reference*.

List account aliases

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAliases(IamClient iam) {  
  
    try {  
        ListAccountAliasesResponse response = iam.listAccountAliases();  
        for (String alias : response.accountAliases()) {  
            System.out.printf("Retrieved account alias %s", alias);  
        }  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Java 2.x API Reference*.

List users

The following code example shows how to list IAM users.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllUsers(IamClient iam ) {  
  
    try {  
  
        boolean done = false;  
        String newMarker = null;  
  
        while(!done) {  
            ListUsersResponse response;  
            if (newMarker == null) {  
                ListUsersRequest request = ListUsersRequest.builder().build();  
                response = iam.listUsers(request);  
            } else {  
                ListUsersRequest request = ListUsersRequest.builder()  
                    .marker(newMarker)  
                    .build();  
  
                response = iam.listUsers(request);  
            }  
  
            for(User user : response.users()) {  
                System.out.format("\n Retrieved user %s", user.userName());  
                AttachedPermissionsBoundary permissionsBoundary =  
user.permissionsBoundary();  
                if (permissionsBoundary != null)  
                    System.out.format("\n Permissions boundary details %s",  
permissionsBoundary.permissionsBoundaryTypeAsString());  
            }  
  
            if(!response.isTruncated()) {  
                done = true;  
            } else {  
                newMarker = response.marker();  
            }  
        }  
    }  
}
```

```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

Update a user

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void updateIAMUser(IamClient iam, String curName, String newName ) {
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [UpdateUser](#) in *AWS SDK for Java 2.x API Reference*.

Update an access key

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
    }
}
```

```
UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
    .accessKeyId(accessId)
    .userName(username)
    .status(statusType)
    .build();

iam.updateAccessKey(request);
System.out.printf("Successfully updated the status of access key %s to" +
    "status %s for user %s", accessId, status, username);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create functions that wrap IAM user actions.

```
public class IAMScenario {

    public static final String PolicyDocument =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [" +
                "{" +
                    "\"Effect\": \"Allow\"," +
                    "\"Action\": [" +
                        "\"s3:*\"" +
                    "]," +
                    "\"Resource\": \"*\"" +
                "}"+
            "]" +
        "};

    public static void main(String[] args) throws Exception {

        final String usage = "\n" +
            "Usage:\n" +
            "    <username> <policyName> <roleName> <roleSessionName> <fileLocation> <bucketName> \n\n" +
```

```
        "Where:\n" +
        "    username - The name of the IAM user to create. \n\n" +
        "    policyName - The name of the policy to create. \n\n" +
        "    roleName - The name of the role to create. \n\n" +
        "    roleSessionName - The name of the session required for the assumeRole
operation. \n\n" +
        "    fileLocation - The file location to the JSON required to create the
role (see Readme). \n\n" +
        "    bucketName - The name of the Amazon S3 bucket from which objects are
read. \n\n" ;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String fileLocation = args[4];
    String bucketName = args[5];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Create the IAM user.
    Boolean createUser = createIAMUser(iam, userName);

    if (createUser) {
        System.out.println(userName + " was successfully created.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        String roleArn = createIAMRole(iam, roleName, fileLocation);
        System.out.println(roleArn + " was successfully created.");
        attachIAMRolePolicy(iam, roleName, polArn);

        System.out.println("*** Wait for 1 MIN so the resource is available");
        TimeUnit.MINUTES.sleep(1);
        assumeGivenRole(roleArn, roleSessionName, bucketName);

        System.out.println("*** Getting ready to delete the AWS resources");
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println("This IAM Scenario has successfully completed");
    } else {
        System.out.println(userName + " was not successfully created.");
    }
}

public static Boolean createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
```

```
        .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return true;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return false;
}

public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return " " ;
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {
```

```

        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            String polArn;
            for (AttachedPolicy policy: attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn)==0) {
                    System.out.println(roleName + " policy is already attached to this
role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to role
" + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeGivenRole(String roleArn, String roleSessionName, String
bucketName) {

        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()

.credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
secKey, secToken)))
                .region(region)
                .build();

            System.out.println("Created a S3Client using temp credentials.");
        }
    }

```

```
        System.out.println("Listing objects in "+bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }
    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {
    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
        DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted "+polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " +roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
    }  
    public static Object readJsonSimpleDemo(String filename) throws Exception {  
        FileReader reader = new FileReader(filename);  
        JSONParser jsonParser = new JSONParser();  
        return jsonParser.parse(reader);  
    }  
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

AWS KMS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS KMS.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 284\)](#)

Actions

Create a grant for a key

The following code example shows how to create a grant for a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createGrant(KmsClient kmsClient, String keyId, String  
granteePrincipal, String operation) {
```

```
try {
    CreateGrantRequest grantRequest = CreateGrantRequest.builder()
        .keyId(keyId)
        .granteePrincipal(granteePrincipal)
        .operationsWithStrings(operation)
        .build();

    CreateGrantResponse response = kmsClient.createGrant(grantRequest);
    return response.grantId();

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
```

- For API details, see [CreateGrant](#) in *AWS SDK for Java 2.x API Reference*.

Create a key

The following code example shows how to create an AWS KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {

    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.printf("Created a customer key with id \"%s\"%n",
            result.keyMetadata().arn());
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Java 2.x API Reference*.

Create an alias for a key

The following code example shows how to create an alias for a KMS key key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {

    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAlias](#) in *AWS SDK for Java 2.x API Reference*.

Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData, String
keyId) {

    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Java 2.x API Reference*.

Describe a key

The following code example shows how to describe a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeSpecificKey(KmsClient kmsClient, String keyId ){
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        System.out.println("The key description is
"+response.keyMetadata().description());
        System.out.println("The key ARN is "+response.keyMetadata().arn());

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeKey](#) in *AWS SDK for Java 2.x API Reference*.

Disable a key

The following code example shows how to disable a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void disableKey( KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisableKey](#) in *AWS SDK for Java 2.x API Reference*.

Enable a key

The following code example shows how to enable a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
```

```
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [EnableKey](#) in *AWS SDK for Java 2.x API Reference*.

Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {
    try {
        SdkBytes myBytes = SdkBytes.fromByteArray(new byte[] {1, 2, 3, 4, 5, 6, 7,
8, 9, 0});

        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The encryption algorithm is " + algorithm);

        // Get the encrypted data.
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Java 2.x API Reference*.

List aliases for a key

The following code example shows how to list aliases for a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllAliases( KmsClient kmsClient) {  
  
    try {  
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()  
            .limit(15)  
            .build();  
  
        ListAliasesResponse aliasesResponse =  
kmsClient.listAliases(aliasesRequest) ;  
        List<AliasListEntry> aliases = aliasesResponse.aliases();  
        for (AliasListEntry alias: aliases) {  
            System.out.println("The alias name is: "+alias.aliasName());  
        }  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListAliases](#) in *AWS SDK for Java 2.x API Reference*.

List grants for a key

The following code example shows how to list grants for a KMS key.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {  
  
    try {  
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()  
            .keyId(keyId)  
            .limit(15)  
            .build();  
  
        ListGrantsResponse response = kmsClient.listGrants(grantsRequest);  
        List<GrantListEntry> grants = response.grants();  
        for ( GrantListEntry grant: grants) {  
            System.out.println("The grant Id is : "+grant.grantId());  
        }  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListGrants](#) in *AWS SDK for Java 2.x API Reference*.

List keys

The following code example shows how to list KMS keys.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllKeys(KmsClient kmsClient) {  
  
    try {  
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()  
            .limit(15)  
            .build();  
  
        ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);  
        List<KeyListEntry> keyListEntries = keysResponse.keys();  
        for (KeyListEntry key : keyListEntries) {  
            System.out.println("The key ARN is: " + key.keyArn());  
            System.out.println("The key Id is: " + key.keyId());  
        }  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Java 2.x API Reference*.

Kinesis examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Kinesis.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 290\)](#)

Actions

Create a stream

The following code example shows how to create a Kinesis stream.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createStream(KinesisClient kinesisClient, String streamName) {  
  
    try {  
        CreateStreamRequest streamReq = CreateStreamRequest.builder()  

```

```
        .streamName(streamName)
        .shardCount(1)
        .build();

        kinesisClient.createStream(streamReq);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateStream](#) in *AWS SDK for Java 2.x API Reference*.

Delete a stream

The following code example shows how to delete a Kinesis stream.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteStream(KinesisClient kinesisClient, String streamName) {
    try {
        DeleteStreamRequest delStream = DeleteStreamRequest.builder()
            .streamName(streamName)
            .build();

        kinesisClient.deleteStream(delStream);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteStream](#) in *AWS SDK for Java 2.x API Reference*.

Get data in batches from a stream

The following code example shows how to get data in batches from a Kinesis stream.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getStockTrades(KinesisClient kinesisClient, String streamName) {
    String shardIterator;
    String lastShardId = null;

    // Retrieve the Shards from a Stream
    DescribeStreamRequest describeStreamRequest = DescribeStreamRequest.builder()
        .streamName(streamName)
        .build();
```



```
List<Shard> shards = new ArrayList<>();
DescribeStreamResponse streamRes;
do {
    streamRes = kinesisClient.describeStream(describeStreamRequest);
    shards.addAll(streamRes.streamDescription().shards());

    if (shards.size() > 0) {
        lastShardId = shards.get(shards.size() - 1).shardId();
    }
} while (streamRes.streamDescription().hasMoreShards());

GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
    .streamName(streamName)
    .shardIteratorType("TRIM_HORIZON")
    .shardId(lastShardId)
    .build();

GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
shardIterator = shardIteratorResult.shardIterator();

// Continuously read data records from shard.
List<Record> records;

// Create new GetRecordsRequest with existing shardIterator.
// Set maximum records to return to 1000.
GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
    .shardIterator(shardIterator)
    .limit(1000)
    .build();

GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [GetRecords](#)
 - [GetShardIterator](#)

Put data into a stream

The following code example shows how to put data into a Kinesis stream.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setStockData( KinesisClient kinesisClient, String streamName) {
    try {
```

```
// Repeatedly send stock trades with a 100 milliseconds wait in between.
StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

// Put in 50 Records for this example.
int index = 50;
for (int x=0; x<index; x++){
    StockTrade trade = stockTradeGenerator.getRandomTrade();
    sendStockTrade(trade, kinesisClient, streamName);
    Thread.sleep(100);
}

} catch (KinesisException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as the
partition key, explained in the Supplemental Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        e.getMessage();
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    }catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
    }
}
```

```
        System.err.println(e);
        System.exit(1);
    }
}
```

- For API details, see [PutRecord](#) in *AWS SDK for Java 2.x API Reference*.

Lambda examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Lambda.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 294\)](#)
- [Scenarios \(p. 296\)](#)

Actions

Create a function

The following code example shows how to create a Lambda function.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createLambdaFunction(LambdaClient awsLambda,
                                       String functionName,
                                       String filePath,
                                       String role,
                                       String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();
    }
}
```

```
        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

        } catch(LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

Delete a function

The following code example shows how to delete a Lambda function.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Java 2.x API Reference*.

Invoke a function

The following code example shows how to invoke a Lambda function.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {
```

```
InvokeResponse res = null ;
try {
    // Need a SdkBytes instance for the payload.
    JSONObject jsonObj = new JSONObject();
    jsonObj.put("inputValue", "2000");
    String json = jsonObj.toString();
    SdkBytes payload = SdkBytes.fromUtf8String(json) ;

    // Setup an InvokeRequest.
    InvokeRequest request = InvokeRequest.builder()
        .functionName(functionName)
        .payload(payload)
        .build();

    res = awsLambda.invoke(request);
    String value = res.payload().asUtf8String() ;
    System.out.println(value);

} catch(LambdaException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [Invoke](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class LambdaScenario {

    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <functionName> <filePath> <role> <handler> <bucketName> <key> \n\n" +
            "Where:\n" +
            "    functionName - The name of the Lambda function. \n"+
```

```
        "    filePath - The path to the .zip or .jar where the code is located.
\n"+
        "    role - The AWS Identity and Access Management (IAM) service role that
has Lambda permissions. \n"+
        "    handler - The fully qualified method name (for example,
example.Handler::handleRequest). \n"+
        "    bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
that contains the .zip or .jar used to update the Lambda function's code. \n"+
        "    key - The Amazon S3 key name that represents the .zip or .jar (for
example, LambdaHello-1.0-SNAPSHOT.jar)." ;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];
    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String funArn = createLambdaFunction(awsLambda, functionName, filePath, role,
handler);
    System.out.println("The AWS Lambda ARN is "+funArn);

    // Get the Lambda function.
    System.out.println("Getting the " +functionName + " AWS Lambda function.");
    getFunction(awsLambda, functionName);

    // List the Lambda functions.
    System.out.println("Listing all functions.");
    LambdaScenario.listFunctions(awsLambda);

    System.out.println("*** Sleep for 1 min to get Lambda function ready.");
    Thread.sleep(60000);

    System.out.println("*** Invoke the Lambda function.");
    invokeFunction(awsLambda, functionName);

    System.out.println("*** Update the Lambda function code.");
    LambdaScenario.updateFunctionCode(awsLambda, functionName, bucketName, key);

    System.out.println("*** Sleep for 1 min to get Lambda function ready.");
    Thread.sleep(60000);
    System.out.println("*** Invoke the Lambda function again with the updated
code.");
    invokeFunction(awsLambda, functionName);

    System.out.println("Update a Lambda function's configuration value.");
    updateFunctionConfiguration(awsLambda, functionName, handler);

    System.out.println("Delete the AWS Lambda function.");
    LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
    awsLambda.close();
}

public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
```

```

        String role,
        String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        return functionResponse.functionArn();

    } catch(LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is "
+response.configuration().runtime());

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config: list) {
            System.out.println("The function name is "+config.functionName());
        }
    }

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

```

    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String functionName,
String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest) ;
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter.waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +response.lastModified());

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler ){
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11 )

```



```
        .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);
    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Amazon Personalize examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 300\)](#)

Actions

Create a batch interface job

The following code example shows how to create a Amazon Personalize batch interface job.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```

    public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                                                            String solutionVersionArn,
                                                            String jobName,
                                                            String
s3InputDataSourcePath,
                                                            String
s3DataDestinationPath,
                                                            String roleArn,
                                                            String explorationWeight,
                                                            String
explorationItemAgeCutOff) {

        long waitInMilliseconds = 60 * 1000;
        String status;
        String batchInferenceJobArn;

        try {

            // Set up data input and output parameters.
            S3DataConfig inputSource = S3DataConfig.builder()
                .path(s3InputDataSourcePath)
                .build();

            S3DataConfig outputDestination = S3DataConfig.builder()
                .path(s3DataDestinationPath)
                .build();

            BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
                .s3DataSource(inputSource)
                .build();

            BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                .s3DataDestination(outputDestination)
                .build();

            // Optional code to build the User-Personalization specific item
exploration config.
            HashMap<String, String> explorationConfig = new HashMap<>();

            explorationConfig.put("explorationWeight", explorationWeight);
            explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

            BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
                .itemExplorationConfig(explorationConfig)
                .build();

            // End optional User-Personalization recipe specific code.

            CreateBatchInferenceJobRequest createBatchInferenceJobRequest =
CreateBatchInferenceJobRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .jobInput(jobInput)
                .jobOutput(jobOutputLocation)
                .jobName(jobName)
                .roleArn(roleArn)
                .batchInferenceJobConfig(jobConfig) // Optional
                .build();

```

```
        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =
DescribeBatchInferenceJobRequest.builder()
                .batchInferenceJobArn(batchInferenceJobArn)
                .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

            BatchInferenceJob batchInferenceJob = personalizeClient
                .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                .batchInferenceJob();

            status = batchInferenceJob.status();
            System.out.println("Batch inference job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return batchInferenceJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for Java 2.x API Reference*.

Create a campaign

The following code example shows how to create a Amazon Personalize campaign.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn, String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                .minProvisionedTPS(1)
                .solutionVersionArn(solutionVersionArn)
                .name(name)
                .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is "+campaignResponse.campaignArn());
    }
```

```
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

Create a dataset

The following code example shows how to create a Amazon Personalize dataset.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createDataset(PersonalizeClient personalizeClient,  
    String datasetName,  
    String datasetGroupArn,  
    String datasetType,  
    String schemaArn) {  
  
    try {  
        CreateDatasetRequest request = CreateDatasetRequest.builder()  
            .name(datasetName)  
            .datasetGroupArn(datasetGroupArn)  
            .datasetType(datasetType)  
            .schemaArn(schemaArn)  
            .build();  
  
        String datasetArn = personalizeClient.createDataset(request)  
            .datasetArn();  
        System.out.println("Dataset " + datasetName + " created.");  
        return datasetArn;  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateDataset](#) in *AWS SDK for Java 2.x API Reference*.

Create a dataset export job

The following code example shows how to create a Amazon Personalize dataset export job.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,  
    String jobName,  
    String datasetArn,
```

```
        IngestionMode ingestionMode,
        String roleArn,
        String s3BucketPath,
        String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig).build();

        CreateDatasetExportJobRequest createRequest =
        CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();

        String datasetExportJobArn =
        personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
        DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetExportJob datasetExportJob =
            personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
                .datasetExportJob();

            status = datasetExportJob.status();
            System.out.println("Export job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                return status;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for Java 2.x API Reference*.

Create a dataset group

The following code example shows how to create a Amazon Personalize dataset group.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient, String
datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Create a domain dataset group.

```
public static String createDomainDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName,
String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .domain(domain)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- For API details, see [CreateDatasetGroup](#) in *AWS SDK for Java 2.x API Reference*.

Create a dataset import job

The following code example shows how to create a Amazon Personalize dataset import job.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,

String jobName,
String datasetArn,
String s3BucketPath,
String roleArn) {
```

```
long waitInMilliseconds = 60 * 1000;
String status;
String datasetImportJobArn;

try {
    DataSource importDataSource = DataSource.builder()
        .dataLocation(s3BucketPath)
        .build();

    CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
    .datasetArn(datasetArn)
    .dataSource(importDataSource)
    .jobName(jobName)
    .roleArn(roleArn)
    .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
    .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
    .datasetImportJobArn(datasetImportJobArn)
    .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
```

- For API details, see [CreateDatasetImportJob](#) in *AWS SDK for Java 2.x API Reference*.

Create a domain schema

The following code example shows how to create an Amazon Personalize domain schema.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createDomainSchema(PersonalizeClient personalizeClient, String
schemaName, String domain, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .domain(domain)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

Create a filter

The following code example shows how to create a Amazon Personalize filter.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
                                String filterName,
                                String datasetGroupArn,
                                String filterExpression) {

    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    }
    catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```



```
}
```

- For API details, see [CreateFilter](#) in *AWS SDK for Java 2.x API Reference*.

Create a recommender

The following code example shows how to create a Amazon Personalize recommender.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
                                       String name,
                                       String datasetGroupArn,
                                       String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
        CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse =
        personalizeClient.createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
        DescribeRecommenderRequest.builder()
            .recommenderArn(recommenderArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            recommenderStatus =
            personalizeClient.describeRecommender(describeRecommenderRequest).recommender().status();
            System.out.println("Recommender status: " + recommenderStatus);

            if (recommenderStatus.equals("ACTIVE") ||
            recommenderStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return recommenderArn;
    } catch (PersonalizeException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateRecommender](#) in *AWS SDK for Java 2.x API Reference*.

Create a schema

The following code example shows how to create a Amazon Personalize schema.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

Create a solution

The following code example shows how to create a Amazon Personalize solution.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                             String datasetGroupArn,
                                             String solutionName,
                                             String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSolution](#) in *AWS SDK for Java 2.x API Reference*.

Create a solution version

The following code example shows how to create a Amazon Personalize solution.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
        }
    }
}
```

```
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }

    if (solutionStatus.equals("ACTIVE")) {

        CreateSolutionVersionRequest createSolutionVersionRequest =
        CreateSolutionVersionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        CreateSolutionVersionResponse createSolutionVersionResponse =
        personalizeClient.createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn =
        createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
        DescribeSolutionVersionRequest.builder()
            .solutionVersionArn(solutionVersionArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
            personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status
            System.out.println("Solution version status: " +
            solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") ||
            solutionVersionStatus.equals("CREATE_FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return solutionVersionArn;
    }
} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateSolutionVersion](#) in *AWS SDK for Java 2.x API Reference*.

Create an event tracker

The following code example shows how to create a Amazon Personalize event tracker.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient, String
eventTrackerName, String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
        .name(eventTrackerName)
        .datasetGroupArn(datasetGroupArn)
        .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient.createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
        .eventTrackerArn(eventTrackerArn)
        .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
            System.out.println("EventTracker status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return eventTrackerId;
    }
    catch (PersonalizeException e){
        System.out.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return eventTrackerId;
}
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

Delete a campaign

The following code example shows how to delete a campaign in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn ) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCampaign](#) in *AWS SDK for Java 2.x API Reference*.

Delete a solution

The following code example shows how to delete a solution in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient, String
solutionArn ) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSolution](#) in *AWS SDK for Java 2.x API Reference*.

Delete an event tracker

The following code example shows how to delete an event tracker in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient, String
eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
        .eventTrackerArn(eventTrackerArn)
        .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);
    }
    catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

Describe a campaign

The following code example shows how to describe a campaign in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
        .campaignArn(campaignArn)
        .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is "+myCampaign.name());
        System.out.println("The Campaign status is "+myCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeCampaign](#) in *AWS SDK for Java 2.x API Reference*.

Describe a recipe

The following code example shows how to describe a recipe in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {
    try{
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
        System.out.println("The recipe name is "+recipeResponse.recipe().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeRecipe](#) in *AWS SDK for Java 2.x API Reference*.

Describe a solution

The following code example shows how to describe a solution in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {
    try {
        DescribeSolutionRequest solutionRequest = DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is "+response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSolution](#) in *AWS SDK for Java 2.x API Reference*.

List campaigns

The following code example shows how to list campaigns in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {
    try{
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign: campaigns) {
            System.out.println("Campaign name is : "+campaign.name());
            System.out.println("Campaign ARN is : "+campaign.campaignArn());
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListCampaigns](#) in *AWS SDK for Java 2.x API Reference*.

List dataset groups

The following code example shows how to list dataset groups in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listDSGroups( PersonalizeClient personalizeClient ) {
    try {
        ListDatasetGroupsRequest groupsRequest = ListDatasetGroupsRequest.builder()
            .maxResults(15)
            .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group: groups) {
            System.out.println("The DataSet name is : "+group.name());
            System.out.println("The DataSet ARN is : "+group.datasetGroupArn());
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListDatasetGroups](#) in *AWS SDK for Java 2.x API Reference*.

List recipes

The following code example shows how to list recipes in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {
    try {
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
            .maxResults(15)
            .build();

        ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
        List<RecipeSummary> recipes = response.recipes();
        for (RecipeSummary recipe: recipes) {
            System.out.println("The recipe ARN is: "+recipe.recipeArn());
            System.out.println("The recipe name is: "+recipe.name());
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListRecipes](#) in *AWS SDK for Java 2.x API Reference*.

List solutions

The following code example shows how to list solutions in Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {
    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution: solutions) {
            System.out.println("The solution ARN is: "+solution.solutionArn());
            System.out.println("The solution name is: "+solution.name());
        }
    }
}
```

```
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSolutions](#) in *AWS SDK for Java 2.x API Reference*.

Update a campaign

The following code example shows how to update a campaign Amazon Personalize.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,  
    String campaignArn,  
    String solutionVersionArn,  
    Integer minProvisionedTPS) {  
  
    try {  
        // build the updateCampaignRequest  
        UpdateCampaignRequest updateCampaignRequest =  
UpdateCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .solutionVersionArn(solutionVersionArn)  
            .minProvisionedTPS(minProvisionedTPS)  
            .build();  
  
        // update the campaign  
        personalizeClient.updateCampaign(updateCampaignRequest);  
  
        DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .build();  
  
        DescribeCampaignResponse campaignResponse =  
personalizeClient.describeCampaign(campaignRequest);  
        Campaign updatedCampaign = campaignResponse.campaign();  
  
        System.out.println("The Campaign status is " + updatedCampaign.status());  
        return updatedCampaign.status();  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [UpdateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Events.

Actions are code excerpts that show you how to call individual Amazon Personalize Events functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 319\)](#)

Actions

Import real-time interaction event data

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                          String datasetArn,
                          String item1Id,
                          String item1PropertyName,
                          String item1PropertyValue,
                          String item2Id,
                          String item2PropertyName,
                          String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s\"}",
                item1PropertyName, item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{ \"%1$s\": \"%2$s\"}",
                item2PropertyName, item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    }
}
```

```
    } catch (PersonalizeEventsException e) {  
        System.out.println(e.awsErrorDetails().errorMessage());  
    }  
    return responseCode;  
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

Incrementally import a user

The following code example shows how to incrementally import a user into Amazon Personalize Events.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,  
    String datasetArn,  
    String user1Id,  
    String user1PropertyName,  
    String user1PropertyValue,  
    String user2Id,  
    String user2PropertyName,  
    String user2PropertyValue) {  
  
    int responseCode = 0;  
    ArrayList<User> users = new ArrayList<>();  
  
    try {  
        User user1 = User.builder()  
            .userId(user1Id)  
            .properties(String.format("{\\\"%1$s\\\": \\\"%2$s\\\"}",  
                user1PropertyName, user1PropertyValue))  
            .build();  
  
        users.add(user1);  
  
        User user2 = User.builder()  
            .userId(user2Id)  
            .properties(String.format("{\\\"%1$s\\\": \\\"%2$s\\\"}",  
                user2PropertyName, user2PropertyValue))  
            .build();  
  
        users.add(user2);  
  
        PutUsersRequest putUsersRequest = PutUsersRequest.builder()  
            .datasetArn(datasetArn)  
            .users(users)  
            .build();  
  
        responseCode =  
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();  
        System.out.println("Response code: " + responseCode);  
        return responseCode;  
  
    } catch (PersonalizeEventsException e) {  
        System.out.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        return responseCode;
    }
```

- For API details, see [PutUsers](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Runtime examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Runtime.

Actions are code excerpts that show you how to call individual Amazon Personalize Runtime functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 321\)](#)

Actions

Get recommendations (custom dataset group)

The following code example shows how to get Amazon Personalize Runtime ranked recommendations.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                                String campaignArn,
                                                String userId,
                                                ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
            .build();

        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
        }
    }
}
```

```
        System.out.println("Item score is : " + item.score());
        System.out.println("-----");
        rank++;
    }
    return rankedItems;
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [GetPersonalizedRanking](#) in *AWS SDK for Java 2.x API Reference*.

Get recommendations from a recommender (domain dataset group)

The following code example shows how to get Amazon Personalize Runtime Runtime recommendations.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Get a list of recommended items.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
        .campaignArn(campaignArn)
        .numResults(20)
        .userId(userId)
        .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Get a list of recommended items from a recommender created in a domain dataset group.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
        .recommenderArn(recommenderArn)
```

```
        .numResults(20)
        .userId(userId)
        .build();

    GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
    List<PredictedItem> items = recommendationsResponse.itemList();

    for (PredictedItem item: items) {
        System.out.println("Item Id is : "+item.itemId());
        System.out.println("Item score is : "+item.score());
    }
} catch (AwsServiceException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Use a filter when requesting recommendations.

```
    public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                     String campaignArn,
                                     String userId,
                                     String filterArn,
                                     String parameter1Name,
                                     String parameter1Value1,
                                     String parameter1Value2,
                                     String parameter2Name,
                                     String parameter2Value){

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("%1$s\", \"%2$s\"",
parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("%1$s\"",
parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .filterArn(filterArn)
            .filterValues(filterValues)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


- For API details, see [GetRecommendations](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#) (p. 324)

Actions

Create a campaign

The following code example shows how to create a campaign.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create a campaign.

```
public static void createPinCampaign(PinpointClient pinpoint, String appId, String
segmentId) {
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());
}

public static CampaignResponse createCampaign(PinpointClient client, String appId,
String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration = MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
```

```
        .name("MyCampaign")
        .segmentId(segmentID)
        .messageConfiguration(messageConfiguration)
        .build();

    CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
        .applicationId(appID)
        .writeCampaignRequest(request).build()
    );

    System.out.println("Campaign ID: " + result.campaignResponse().id());
    return result.campaignResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return null;
}
```

- For API details, see [CreateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

Create a segment

The following code example shows how to create a segment.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static SegmentResponse createSegment(PinpointClient client, String appId) {

    try {
        Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
        segmentAttributes.put("Team", AttributeDimension.builder()
            .attributeType(AttributeType.INCLUSIVE)
            .values("Lakers")
            .build());

        RecencyDimension recencyDimension = RecencyDimension.builder()
            .duration("DAY_30")
            .recencyType("ACTIVE")
            .build();

        SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
            .recency(recencyDimension)
            .build();

        SegmentDemographics segmentDemographics = SegmentDemographics
            .builder()
            .build();

        SegmentLocation segmentLocation = SegmentLocation
            .builder()
            .build();

        SegmentDimensions dimensions = SegmentDimensions
            .builder()
    }
```

```
        .attributes(segmentAttributes)
        .behavior(segmentBehaviors)
        .demographic(segmentDemographics)
        .location(segmentLocation)
        .build();

    WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
        .name("MySegment")
        .dimensions(dimensions)
        .build();

    CreateSegmentRequest createSegmentRequest = CreateSegmentRequest.builder()
        .applicationId(appId)
        .writeSegmentRequest(writeSegmentRequest)
        .build();

    CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
    System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
    System.out.println("Done");
    return createSegmentResult.segmentResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [CreateSegment](#) in *AWS SDK for Java 2.x API Reference*.

Create an application

The following code example shows how to create an application.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createApplication(PinpointClient pinpoint, String appName) {

    try {
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateApp](#) in *AWS SDK for Java 2.x API Reference*.

Delete an application

The following code example shows how to delete an application.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Delete an application.

```
public static void deletePinApp(PinpointClient pinpoint, String appId ) {  
  
    try {  
        DeleteAppRequest appRequest = DeleteAppRequest.builder()  
            .applicationId(appId)  
            .build();  
  
        DeleteAppResponse result = pinpoint.deleteApp(appRequest);  
        String appName = result.applicationResponse().name();  
        System.out.println("Application " + appName + " has been deleted.");  
  
    } catch (PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Java 2.x API Reference*.

Delete an endpoint

The following code example shows how to delete an endpoint.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Delete an endpoint.

```
public static void deletePinEndpoint(PinpointClient pinpoint, String appId, String  
endpointId ) {  
  
    try {  
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()  
            .applicationId(appId)  
            .endpointId(endpointId)  
            .build();  
  
        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);  
        String id = result.endpointResponse().id();  
        System.out.println("The deleted endpoint id " + id);  
  
    } catch (PinpointException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

Export an endpoint

The following code example shows how to export an endpoint.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Export an endpoint.

```
public static void exportAllEndpoints(PinpointClient pinpoint,
                                     S3Client s3Client,
                                     String applicationId,
                                     String s3BucketName,
                                     String path,
                                     String iamExportRoleArn) {

    try {
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn, applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz")).collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName, String iamExportRoleArn, String applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix + "/";
    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest = CreateExportJobRequest.builder()
            .applicationId(applicationId)
            .exportJobRequest(jobRequest)
            .build();
    }
}
```

```
        System.out.format("Exporting endpoints from Amazon Pinpoint application %s
to Amazon S3 " +
        "bucket %s . . .\n", applicationId, s3BucketName);

        CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
        String jobId = exportResult.exportJobResponse().id();
        System.out.println(jobId);
        printExportJobStatus(pinpoint, applicationId, jobId);

        ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
            .bucket(s3BucketName)
            .prefix(endpointsKeyPrefix)
            .build();

        // Create a list of object keys.
        ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
        List<S3Object> objects = v2Response.contents();
        for (S3Object object: objects) {
            key = object.key();
            objectKeys.add(key);
        }

        return objectKeys;

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void printExportJobStatus(PinpointClient pinpointClient,
        String applicationId,
        String jobId) {

    GetExportJobResponse getExportJobResult;
    String status;

    try {
        // Checks the job status until the job completes or fails.
        GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
            .jobId(jobId)
            .applicationId(applicationId)
            .build();

        do {
            getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
            status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
            System.out.format("Export job %s . . .\n", status);
            TimeUnit.SECONDS.sleep(3);

        } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

        if (status.equals("COMPLETED")) {
            System.out.println("Finished exporting endpoints.");
        } else {
            System.err.println("Failed to export endpoints.");
            System.exit(1);
        }
    }

    } catch (PinpointException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(s3BucketName)
                    .key(key)
                    .build();

                ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
                newPath = path + fileSuffix+".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");
        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [CreateExportJob](#) in *AWS SDK for Java 2.x API Reference*.

Get endpoints

The following code example shows how to get endpoints.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint ) {

    try {
        GetEndpointRequest appRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpoint)
            .build();

        GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
        EndpointResponse endResponse = result.endpointResponse();

        // Uses the Google Gson library to pretty print the endpoint JSON.
        Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create();
    }
}
```

```
String endpointJson = gson.toJson(endResponse);
System.out.println(endpointJson);

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- For API details, see [GetEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

Import a segment

The following code example shows how to import a segment.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Import a segment.

```
public static ImportJobResponse createImportSegment(PinpointClient client,
                                                    String appId,
                                                    String bucket,
                                                    String key,
                                                    String roleArn) {

    try {
        ImportJobRequest importRequest = ImportJobRequest.builder()
            .defineSegment(true)
            .registerEndpoints(true)
            .roleArn(roleArn)
            .format(Format.JSON)
            .s3Url("s3://" + bucket + "/" + key)
            .build();

        CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
            .importJobRequest(importRequest)
            .applicationId(appId)
            .build();

        CreateImportJobResponse jobResponse = client.createImportJob(jobRequest);
        return jobResponse.importJobResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [CreateImportJob](#) in *AWS SDK for Java 2.x API Reference*.

List endpoints

The following code example shows how to list endpoints.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllEndpoints(PinpointClient pinpoint,
                                   String applicationId,
                                   String userId) {

    try {
        GetUserEndpointsRequest endpointsRequest =
        GetUserEndpointsRequest.builder()
            .userId(userId)
            .applicationId(applicationId)
            .build();

        GetUserEndpointsResponse response =
        pinpoint.getUserEndpoints(endpointsRequest);
        List<EndpointResponse> endpoints = response.endpointsResponse().item();

        // Display the results.
        for (EndpointResponse endpoint: endpoints) {
            System.out.println("The channel type is: "+endpoint.channelType());
            System.out.println("The address is "+endpoint.address());
        }

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetUserEndpoints](#) in *AWS SDK for Java 2.x API Reference*.

List segments

The following code example shows how to list segments.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

List segments.

```
public static void listSegs( PinpointClient pinpoint, String appId) {

    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for(SegmentResponse segment: segments) {
            System.out.println("Segement " + segment.id() + " " + segment.name() +
            " " + segment.lastModifiedDate());
        }

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- For API details, see [GetSegments](#) in *AWS SDK for Java 2.x API Reference*.

Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Send an email message.

```
public static void sendEmail(PinpointClient pinpoint,
                             String subject,
                             String appId,
                             String senderAddress,
                             String toAddress) {

    try {
        Map<String,AddressConfiguration> addressMap = new HashMap<>();
        AddressConfiguration configuration = AddressConfiguration.builder()
            .channelType(ChannelType.EMAIL)
            .build();

        addressMap.put(toAddress, configuration);
        SimpleEmailPart emailPart = SimpleEmailPart.builder()
            .data(htmlBody)
            .charset(charset)
            .build();

        SimpleEmailPart subjectPart = SimpleEmailPart.builder()
            .data(subject)
            .charset(charset)
            .build();

        SimpleEmail simpleEmail = SimpleEmail.builder()
            .htmlPart(emailPart)
            .subject(subjectPart)
            .build();

        EmailMessage emailMessage = EmailMessage.builder()
            .body(htmlBody)
            .fromAddress(senderAddress)
            .simpleEmail(simpleEmail)
            .build();

        DirectMessageConfiguration directMessageConfiguration =
        DirectMessageConfiguration.builder()
            .emailMessage(emailMessage)
            .build();

        MessageRequest messageRequest = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(directMessageConfiguration)
            .build();

        SendMessagesRequest messagesRequest = SendMessagesRequest.builder()
            .applicationId(appId)
```

```
        .messageRequest(messageRequest)
        .build();

    pinpoint.sendMessage(messagesRequest);
} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Send an SMS message.

```
public static void sendSMSMessage(PinpointClient pinpoint, String message, String
appId, String originationNumber, String destinationNumber) {

    try {
        Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
        AddressConfiguration addConfig = AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response= pinpoint.sendMessage(request);
        MessageResponse msg1 = response.getMessageResponse();
        Map map1 = msg1.getResult();

        //Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Java 2.x API Reference*.

Update an endpoint

The following code example shows how to update an endpoint.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```

public static EndpointResponse createEndpoint(PinpointClient client, String appId)
{
    String endpointId = UUID.randomUUID().toString();
    System.out.println("Endpoint ID: " + endpointId);

    try {
        EndpointRequest endpointRequest = createEndpointRequestData();
        UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .endpointRequest(endpointRequest)
            .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
        System.out.println(getEndpointResponse.endpointResponse().address());
        System.out.println(getEndpointResponse.endpointResponse().channelType());
        System.out.println(getEndpointResponse.endpointResponse().applicationId());

        System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
        System.out.println(getEndpointResponse.endpointResponse().requestId());
        System.out.println(getEndpointResponse.endpointResponse().user());

        return getEndpointResponse.endpointResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static EndpointRequest createEndpointRequestData() {
    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
            .appVersion("1.0")
            .make("apple")
    
```

```
.model("iPhone")
.modelVersion("7")
.platform("ios")
.platformVersion("10.1.1")
.timezone("America/Los_Angeles")
.build();

EndpointLocation location = EndpointLocation.builder()
    .city("Los Angeles")
    .country("US")
    .latitude(34.0)
    .longitude(-118.2)
    .postalCode("90068")
    .region("CA")
    .build();

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = EndpointUser.builder()
    .userId(UUID.randomUUID().toString())
    .build();

DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
"Z" to indicate UTC, no timezone offset
String nowAsISO = df.format(new Date());

return EndpointRequest.builder()
    .address(UUID.randomUUID().toString())
    .attributes(customAttributes)
    .channelType("APNS")
    .demographic(demographic)
    .effectiveDate(nowAsISO)
    .location(location)
    .metrics(metrics)
    .optOut("NONE")
    .requestId(UUID.randomUUID().toString())
    .user(user)
    .build();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [UpdateEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

Update channels

The following code example shows how to update channels.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
private static SMSChannelResponse getSMSChannel(PinpointClient client, String
appId) {
```

```
    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
            .build();

        SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();

    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest = UpdateSmsChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSmsChannelResponse result = client.updateSmsChannel(updateRequest);
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSmsChannel](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint SMS and Voice API.

Actions are code excerpts that show you how to call individual Amazon Pinpoint SMS and Voice API functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon Pinpoint SMS and Voice API functions.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 338\)](#)

Actions

Send a voice message with Amazon Pinpoint SMS and Voice API

The following code example shows how to send a voice message with Amazon Pinpoint SMS and Voice API.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber, String destinationNumber ) {

    try {
        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
        SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for Java 2.x API Reference*.

Amazon RDS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon RDS.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 339\)](#)

Actions

Create an instance

The following code example shows how to create an Amazon RDS instance and wait for it to be in an available state.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createDatabaseInstance(RdsClient rdsClient,
                                         String dbInstanceIdentifier,
                                         String dbName,
                                         String masterUsername,
                                         String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest = CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0.15")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
            rdsClient.createDBInstance(instanceRequest);
        System.out.println("The status is " +
            response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {

    Boolean instanceReady = false;
    String instanceReadyStr = "";
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbInstancesRequest instanceRequest =
            DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

        // Loop until the cluster is ready
        while (!instanceReady) {

            DescribeDbInstancesResponse response =
                rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
            }
        }
    }
}
```



```
        if (instanceReadyStr.contains("available"))
            instanceReady = true;
        else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database instance is available!");
} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

Delete an instance

The following code example shows how to delete an Amazon RDS instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

Describe an instance

The following code example shows how to describe an Amazon RDS instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeInstances(RdsClient rdsClient) {  
    try {  
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();  
        List<DBInstance> instanceList = response.dbInstances();  
        for (DBInstance instance: instanceList) {  
            System.out.println("Instance ARN is: "+instance.dbInstanceArn());  
            System.out.println("The Engine is " +instance.engine());  
            System.out.println("Connection endpoint is "  
+instance.endpoint().address());  
        }  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Java 2.x API Reference*.

Modify an instance

The following code example shows how to modify an Amazon RDS instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeInstances(RdsClient rdsClient) {  
    try {  
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();  
        List<DBInstance> instanceList = response.dbInstances();  
        for (DBInstance instance: instanceList) {  
            System.out.println("Instance ARN is: "+instance.dbInstanceArn());  
            System.out.println("The Engine is " +instance.engine());  
            System.out.println("Connection endpoint is "  
+instance.endpoint().address());  
        }  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ModifyDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

Reboot an instance

The following code example shows how to reboot an Amazon RDS instance.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeInstances(RdsClient rdsClient) {  
    try {  
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();  
        List<DBInstance> instanceList = response.dbInstances();  
        for (DBInstance instance: instanceList) {  
            System.out.println("Instance ARN is: "+instance.dbInstanceArn());  
            System.out.println("The Engine is " +instance.engine());  
            System.out.println("Connection endpoint is "  
+instance.endpoint().address());  
        }  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [RebootDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

Retrieve attributes

The following code example shows how to retrieve attributes that belong to an Amazon RDS account.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getAccountAttributes(RdsClient rdsClient) {  
    try {  
        DescribeAccountAttributesResponse response =  
rdsClient.describeAccountAttributes();  
        List<AccountQuota> quotasList = response.accountQuotas();  
        for (AccountQuota quotas: quotasList) {  
            System.out.println("Name is: "+quotas.accountQuotaName());  
            System.out.println("Max value is " +quotas.max());  
        }  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeAccountAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Redshift examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Redshift.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 343\)](#)

Actions

Create a cluster

The following code example shows how to create an Amazon Redshift cluster.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create the cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String clusterId,
String masterUsername, String masterUserPassword ) {

    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
            .masterUserPassword(masterUserPassword) // set the user password here
            .nodeType("ds2.xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
            redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
            clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Java 2.x API Reference*.

Delete a cluster

The following code example shows how to delete an Amazon Redshift cluster.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Delete the cluster.

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
```

```
try {
    DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .skipFinalClusterSnapshot(true)
        .build();

    DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
    System.out.println("The status is "+response.cluster().clusterStatus());

} catch (RedshiftException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Java 2.x API Reference*.

Describe your clusters

The following code example shows how to describe your Amazon Redshift clusters.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Describe the cluster.

```
public static void describeRedshiftClusters(RedshiftClient redshiftClient) {

    try {
        DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters();
        List<Cluster> clusterList = clusterResponse.clusters();
        for (Cluster cluster: clusterList) {
            System.out.println("Cluster database name is: "+cluster.dbName());
            System.out.println("Cluster status is: "+cluster.clusterStatus());
        }

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeClusters](#) in *AWS SDK for Java 2.x API Reference*.

Modify a cluster

The following code example shows how to modify an Amazon Redshift cluster.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Modify a cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String clusterId)
{
    try {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "+ clusterResponse.cluster().preferredMaintenanceWindow() +" as the maintenance
window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyCluster](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Rekognition examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Rekognition.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 345\)](#)
- [Scenarios \(p. 356\)](#)

Actions

Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage, String targetImage) {
    try {
```

```
InputStream sourceStream = new FileInputStream(sourceImage);
InputStream tarStream = new FileInputStream(targetImage);
SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

// Create an Image object for the source image.
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

Image tarImage = Image.builder()
    .bytes(targetBytes)
    .build();

CompareFacesRequest facesRequest = CompareFacesRequest.builder()
    .sourceImage(souImage)
    .targetImage(tarImage)
    .similarityThreshold(similarityThreshold)
    .build();

// Compare the two images.
CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
for (CompareFacesMatch match: faceDetails){
    ComparedFace face= match.face();
    BoundingBox position = face.boundingBox();
    System.out.println("Face at " + position.left().toString()
        + " " + position.top()
        + " matches with " + face.confidence().toString()
        + "% confidence.");
}
List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
System.out.println("There was " + uncompered.size() + " face(s) that did
not match");
System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println("Failed to load source image " + sourceImage);
    System.exit(1);
}
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for Java 2.x API Reference*.

Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createMyCollection(RekognitionClient rekClient, String
collectionId ) {
```

```
    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
        .collectionId(collectionId)
        .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " + collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for Java 2.x API Reference*.

Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteMyCollection(RekognitionClient rekClient,String
collectionId ) {

    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
        .collectionId(collectionId)
        .build();

        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCollection](#) in *AWS SDK for Java 2.x API Reference*.

Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteFacesCollection(RecognitionClient rekClient,
                                         String collectionId,
                                         String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RecognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFaces](#) in *AWS SDK for Java 2.x API Reference*.

Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeColl(RecognitionClient rekClient, String collectionName)
{
    try {
        DescribeCollectionRequest describeCollectionRequest =
        DescribeCollectionRequest.builder()
            .collectionId(collectionName)
            .build();

        DescribeCollectionResponse describeCollectionResponse =
        rekClient.describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
        describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
        describeCollectionResponse.creationTimestamp().toString());

    } catch (RecognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Java 2.x API Reference*.

Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectFacesinImage(RekognitionClient rekClient,String
sourceImage ) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectFacesRequest facesRequest = DetectFacesRequest.builder()
            .attributes(Attribute.ALL)
            .image(souImage)
            .build();

        DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
        List<FaceDetail> faceDetails = facesResponse.faceDetails();
        for (FaceDetail face : faceDetails) {
            AgeRange ageRange = face.ageRange();
            System.out.println("The detected face is estimated to be between "
                + ageRange.low().toString() + " and " +
ageRange.high().toString()
                + " years old.");

            System.out.println("There is a smile :
"+face.smile().value().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectFaces](#) in *AWS SDK for Java 2.x API Reference*.

Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label: labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for Java 2.x API Reference*.

Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectModLabels(RekognitionClient rekClient, String sourceImage)
{

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
            .image(souImage)
            .minConfidence(60F)
    }
```

```
        .build();

        DetectModerationLabelsResponse moderationLabelsResponse =
rekClient.detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels = moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");

        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Java 2.x API Reference*.

Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text: textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- For API details, see [DetectText](#) in *AWS SDK for Java 2.x API Reference*.

Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void addToCollection(RekognitionClient rekClient, String  
collectionId, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(sourceImage);  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
        Image souImage = Image.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        IndexFacesRequest facesRequest = IndexFacesRequest.builder()  
            .collectionId(collectionId)  
            .image(souImage)  
            .maxFaces(1)  
            .qualityFilter(QualityFilter.AUTO)  
            .detectionAttributes(Attribute.DEFAULT)  
            .build();  
  
        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);  
        System.out.println("Results for the image");  
        System.out.println("\n Faces indexed:");  
        List<FaceRecord> faceRecords = facesResponse.faceRecords();  
        for (FaceRecord faceRecord : faceRecords) {  
            System.out.println(" Face ID: " + faceRecord.face().faceId());  
            System.out.println(" Location: " +  
faceRecord.faceDetail().boundingBox().toString());  
        }  
  
        List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();  
        System.out.println("Faces not indexed:");  
        for (UnindexedFace unindexedFace : unindexedFaces) {  
            System.out.println(" Location: " +  
unindexedFace.faceDetail().boundingBox().toString());  
            System.out.println(" Reasons:");  
            for (Reason reason : unindexedFace.reasons()) {  
                System.out.println("Reason: " + reason);  
            }  
        }  
    }  
  
    } catch (RekognitionException | FileNotFoundException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for Java 2.x API Reference*.

List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllCollections(RekognitionClient rekClient) {
    try {
        ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
            .maxResults(10)
            .build();

        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }
    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListCollections](#) in *AWS SDK for Java 2.x API Reference*.

List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listFacesCollection(RekognitionClient rekClient, String
collectionId ) {
    try {
        ListFacesRequest facesRequest = ListFacesRequest.builder()
            .collectionId(collectionId)
            .maxResults(10)
            .build();

        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face: faces) {
```

```
        System.out.println("Confidence level there is a face:
"+face.confidence());
        System.out.println("The face Id value is "+face.faceId());
    }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Java 2.x API Reference*.

Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request = RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs=result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity: celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url: celebrity.urls()){
                System.out.println(url);
            }
            System.out.println();
        }
        System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Java 2.x API Reference*.

Search for faces in a collection

The following code example shows how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see [Searching for a face \(face ID\)](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is "+face.similarity());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [SearchFaces](#) in *AWS SDK for Java 2.x API Reference*.

Search for faces in a collection compared to a reference image

The following code example shows how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see [Searching for a face \(image\)](#).

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).


```
public static void searchFaceById(RekognitionClient rekClient, String collectionId,
String faceId) {

    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is "+face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Detect information in videos

The following code example shows how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Get celebrity results from a video located in an Amazon S3 bucket.

```
public static void StartCelebrityDetection(RekognitionClient rekClient,
                                           NotificationChannel channel,
                                           String bucket,
                                           String video){

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();
```

```

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
    .jobTag("Celebrities")
    .notificationChannel(channel)
    .video(vidOb)
    .build();

        StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient.startCelebrityRecognition(recognitionRequest);
        startJobId = startCelebrityRecognitionResult.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{

            if (recognitionResponse !=null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
    .maxResults(10)
    .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
            for (CelebrityRecognition celeb: celebs) {

```

```
        long seconds=celeb.timestamp()/1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details=celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

Detect labels in a video by a label detection operation.

```
public static void startLabels(RekognitionClient rekClient,
                              NotificationChannel channel,
                              String bucket,
                              String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
        rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
            GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
            rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: "+status);
        }
    }
}
```

```

        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId + " status is: "+status);

} catch(RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs, String
queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId );
                    System.out.println("Status : " + operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }

                else{
                    System.out.println("Job received was not job " + startJobId);
                    sqs.deleteMessage(deleteMessageRequest);
                }
            }
        }

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

```

```

    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("  Label:" + label.name());
                System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("  Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("          " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("          Confidence: " +
instance.confidence().toString());
                        System.out.println("          Bounding box: " +
instance.boundingBox().toString());
                    }
                }
                System.out.println("  Parent labels for " + label.name() + ":");
                List<Parent> parents = label.parents();

                if (parents.isEmpty()) {
                    System.out.println("          None");
                } else {
                    for (Parent parent : parents) {

```

```
                System.out.println("    " + parent.name());
            }
        }
        System.out.println();
    }
} while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=
null);

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}
```

Detect faces in a video stored in an Amazon S3 bucket.

```
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
        rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
            GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
            rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: "+status);

            Thread.sleep(1000);
            yy++;
        }
    }
}
```

```

    }

    System.out.println(startJobId + " status is: "+status);

} catch(RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs, String
queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId);
                    System.out.println("Status : " + operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }

                else{
                    System.out.println("Job received was not job " + startJobId);
                    sqs.deleteMessage(deleteMessageRequest);
                }
            }
        }

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    }
}

```

```

    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("  Label:" + label.name());
                System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("  Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("          " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("          Confidence: " +
instance.confidence().toString());
                        System.out.println("          Bounding box: " +
instance.boundingBox().toString());
                    }
                }
                System.out.println("  Parent labels for " + label.name() + ":");
                List<Parent> parents = label.parents();

                if (parents.isEmpty()) {
                    System.out.println("          None");
                } else {
                    for (Parent parent : parents) {
                        System.out.println("          " + parent.name());
                    }
                }
            }
        }
    }
}

```



```
        }
        System.out.println();
    }
    } while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=
null);

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```
public static void startModerationDetection(RekognitionClient rekClient,
                                           NotificationChannel channel,
                                           String bucket,
                                           String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartContentModerationResponse startModDetectionResult =
rekClient.startContentModeration(modDetectionRequest);
        startJobId=startModDetectionResult.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetModResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetContentModerationResponse modDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (modDetectionResponse !=null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
```

```
        .build();

// Wait until the job succeeds
while (!finished) {
    modDetectionResponse = rekClient.getContentModeration(modRequest);
    status = modDetectionResponse.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + status);
        Thread.sleep(1000);
    }
    yy++;
}

finished = false;

// Proceed when the job is done - otherwise VideoMetadata is null
VideoMetadata videoMetaData=modDetectionResponse.videoMetadata();
System.out.println("Format: " + videoMetaData.format());
System.out.println("Codec: " + videoMetaData.codec());
System.out.println("Duration: " + videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());
System.out.println("Job");

List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
for (ContentModerationDetection mod: mods) {
    long seconds=mod.timestamp()/1000;
    System.out.print("Mod label: " + seconds + " ");
    System.out.println(mod.moderationLabel().toString());
    System.out.println();
}

} while (modDetectionResponse !=null && modDetectionResponse.nextToken() !=
null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

Detect technical cue segments and shot detection segments in a video stored in an Amazon S3 bucket.

```
public static void StartSegmentDetection (RekognitionClient rekClient,
                                         NotificationChannel channel,
                                         String bucket,
                                         String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
        StartShotDetectionFilter.builder()
```

```
        .minSegmentConfidence(60F)
        .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
        .minSegmentConfidence(60F)
        .build();

        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
        .shotFilter(cueDetectionFilter)
        .technicalCueFilter(technicalCueDetectionFilter)
        .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .segmentTypes(SegmentType.TECHNICAL_CUE , SegmentType.SHOT)
        .video(vidOb)
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                status = segDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
        }
    }
}
```

```

    }
    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
    for (VideoMetadata metaData : videoMetaData) {
        System.out.println("Format: " + metaData.format());
        System.out.println("Codec: " + metaData.codec());
        System.out.println("Duration: " + metaData.durationMillis());
        System.out.println("FrameRate: " + metaData.frameRate());
        System.out.println("Job");
    }

    List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
    for (SegmentDetection detectedSegment : detectedSegments) {
        String type = detectedSegment.type().toString();
        if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
            System.out.println("Technical Cue");
            TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
            System.out.println("\tType: " + segmentCue.type());
            System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
        }

        if (type.contains(SegmentType.SHOT.toString())) {
            System.out.println("Shot");
            ShotSegment segmentShot = detectedSegment.shotSegment();
            System.out.println("\tIndex " + segmentShot.index());
            System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
        }

        long seconds = detectedSegment.durationMillis();
        System.out.println("\tDuration : " + seconds + " milliseconds");
        System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
        System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
        System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
        System.out.println();
    }

    } while (segDetectionResponse !=null && segDetectionResponse.nextToken() !=
null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

```

Detect text in a video stored in a video stored in an Amazon S3 bucket.

```

public static void startTextLabels(RekognitionClient rekClient,
                                NotificationChannel channel,
                                String bucket,
                                String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)

```

```

        .name(video)
        .build();

    Video vidOb = Video.builder()
        .s3Object(s3Obj)
        .build();

    StartTextDetectionRequest labelDetectionRequest =
    StartTextDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vidOb)
        .build();

    StartTextDetectionResponse labelDetectionResponse =
    rekClient.startTextDetection(labelDetectionRequest);
    startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
            GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
            rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());

```

```
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText: labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " + detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " + detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

        } while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

Detect people in a video stored in a video stored in an Amazon S3 bucket.

```
public static void startPersonLabels(RekognitionClient rekClient,
                                    NotificationChannel channel,
                                    String bucket,
                                    String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vidOb)
            .notificationChannel(channel)
            .build();

        StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetPersonDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetPersonTrackingResponse personTrackingResult=null;
```

```

boolean finished = false;
String status;
int yy=0 ;

do{
    if (personTrackingResult !=null)
        paginationToken = personTrackingResult.nextToken();

        GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

        // Wait until the job succeeds
        while (!finished) {

            personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
            status = personTrackingResult.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + "status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null
        VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<PersonDetection> detectedPersons= personTrackingResult.persons();
        for (PersonDetection detectedPerson: detectedPersons) {

            long seconds=detectedPerson.timestamp()/1000;
            System.out.print("Sec: " + seconds + " ");
            System.out.println("Person Identifier: " +
detectedPerson.person().index());
            System.out.println();
        }

    } while (personTrackingResult !=null && personTrackingResult.nextToken() !=
null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [GetCelebrityRecognition](#)

- [GetContentModeration](#)
- [GetLabelDetection](#)
- [GetPersonTracking](#)
- [GetSegmentDetection](#)
- [GetTextDetection](#)
- [StartCelebrityRecognition](#)
- [StartContentModeration](#)
- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

Amazon S3 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon S3.

Actions are code excerpts that show you how to call individual Amazon S3 functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 371\)](#)
- [Scenarios \(p. 390\)](#)

Actions

Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketCors(bucketCorsRequest) ;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```

    }
}

public static void getBucketCorsInformation(S3Client s3, String bucketName, String
accountId) {

    try {
        GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        GetBucketCorsResponse corsResponse = s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule: corsRules) {
            System.out.println("allowOrigins: "+rule.allowedOrigins());
            System.out.println("AllowedMethod: "+rule.allowedMethods());
        }

    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {

    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest = PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- For API details, see [PutBucketCors](#) in *AWS SDK for Java 2.x API Reference*.

Add a lifecycle configuration to a bucket

The following code example shows how to add a lifecycle configuration to an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        // Create a rule to archive objects with the "glacierobjects/" prefix to
Amazon S3 Glacier.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("Archive immediately rule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Create a second rule.
        Transition transition2 = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        List<Transition> transitionList = new ArrayList<>();
        transitionList.add(transition2);

        LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        LifecycleRule rule2 = LifecycleRule.builder()
            .id("Archive and then delete rule")
            .filter(ruleFilter2)
            .transitions(transitionList)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(ruleList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
```

```
        .bucket(bucketName)
        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

    s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

// Retrieve the configuration and add a new rule.
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId){

    try {
        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest =
GetBucketLifecycleConfigurationRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        GetBucketLifecycleConfigurationResponse response =
s3.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
        List<LifecycleRule> newList = new ArrayList<>();
        List<LifecycleRule> rules = response.rules();
        for (LifecycleRule rule: rules) {
            newList.add(rule);
        }

        // Add a new rule with both a prefix predicate and a tag predicate.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("YearlyDocuments/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(newList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
            .bucket(bucketName)
            .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);
```

```
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the configuration from the Amazon S3 bucket.
public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {

    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutBucketLifecycleConfiguration](#) in *AWS SDK for Java 2.x API Reference*.

Add a policy to a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setPolicy(S3Client s3, String bucketName, String policyText) {

    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
        .bucket(bucketName)
        .policy(policyText)
        .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
```

```
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

Copy an object from one bucket to another

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

    String encodedUrl = "";
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());

    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }

    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySourceIfMatch(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Java 2.x API Reference*.

Create a bucket

The following code example shows how to create an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createBucket( S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Java 2.x API Reference*.

Delete a policy from a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {

    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();
```

```
try {
    s3.deleteBucketPolicy(delReq);
    System.out.println("Done!");
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

Delete an empty bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Java 2.x API Reference*.

Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteBucketObjects(S3Client s3, String bucketName) {

    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
    }
}
```

```
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(del)
        .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Java 2.x API Reference*.

Delete the website configuration from a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteBucketWebsiteConfig(S3Client s3,String bucketName ) {

    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

Determine the existence and content type of an object

The following code example shows how to determine the existence and content type of an object in an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getContentType (S3Client s3, String bucketName, String keyName)
{
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is "+type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [HeadObject](#) in *AWS SDK for Java 2.x API Reference*.

Get an object from a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Read data as a byte array.

```
public static void getObjectBytes (S3Client s3, String bucketName, String keyName,
String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path );
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

Read tags that belong to an object.

```
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet= tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Get a URL for an object.

```
public static void getURL(S3Client s3, String bucketName, String keyName ) {
    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for "+keyName+" is "+ url);
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Get an object by using the S3Presigner client object.

```
public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName ) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
        GetObjectPresignRequest.builder()
```

```
        .signatureDuration(Duration.ofMinutes(60))
        .getObjectRequest(getObjectRequest)
        .build();

    PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
    String theUrl = presignedGetObjectRequest.url().toString();
    System.out.println("Presigned URL: " + theUrl);
    HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
    presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
        values.forEach(value -> {
            connection.setRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream = connection.getOutputStream()) {
            IOUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IOUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
```

- For API details, see [GetObject](#) in *AWS SDK for Java 2.x API Reference*.

Get the ACL of a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String getBucketACL(S3Client s3, String objectKey, String bucketName)
{
    try {
        GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
        List<Grant> grants = aclRes.grants();
    }
}
```

```
String grantee = "";
for (Grant grant : grants) {
    System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
    grantee = grant.grantee().id();
}

return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- For API details, see [GetBucketAcl](#) in *AWS SDK for Java 2.x API Reference*.

Get the policy for a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String getPolicy(S3Client s3, String bucketName) {

    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

List in-progress multipart uploads

The following code example shows how to list in-progress multipart uploads to an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listUploads( S3Client s3, String bucketName) {  
    try {  
        ListMultipartUploadsRequest listMultipartUploadsRequest =  
ListMultipartUploadsRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
        ListMultipartUploadsResponse response =  
s3.listMultipartUploads(listMultipartUploadsRequest);  
        List<MultipartUpload> uploads = response.uploads();  
        for (MultipartUpload upload: uploads) {  
            System.out.println("Upload in progress: Key = \"" + upload.key() + "\",  
id = " + upload.uploadId());  
        }  
  
    } catch (S3Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListMultipartUploads](#) in *AWS SDK for Java 2.x API Reference*.

List objects in a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listBucketObjects(S3Client s3, String bucketName ) {  
  
    try {  
        ListObjectsRequest listObjects = ListObjectsRequest  
        .builder()  
        .bucket(bucketName)  
        .build();  
  
        ListObjectsResponse res = s3.listObjects(listObjects);  
        List<S3Object> objects = res.contents();  
        for (S3Object myValue : objects) {  
            System.out.print("\n The name of the key is " + myValue.key());  
            System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");  
            System.out.print("\n The owner is " + myValue.owner());  
        }  
  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
//convert bytes to kbs.  
private static long calKb(Long val) {  
    return val/1024;  
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Java 2.x API Reference*.

Restore an archived copy of an object

The following code example shows how to restore an archived copy of an object back into an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void restoreS3Object(S3Client s3, String bucketName, String keyName,
String expectedBucketOwner) {

    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

        .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [RestoreObject](#) in *AWS SDK for Java 2.x API Reference*.

Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setBucketAcl(S3Client s3, String bucketName, String id) {

    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id)
            .type(Type.CANONICAL_USER))
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
```

```
        .owner(builder -> builder.id(id))
        .grants(grantList2)
        .build();

    PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
        .bucket(bucketName)
        .accessControlPolicy(acl)
        .build();

    s3.putBucketAcl(putAclReq);

} catch (S3Exception e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Java 2.x API Reference*.

Set the website configuration for a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setWebsiteConfig( S3Client s3, String bucketName, String
indexDoc) {

    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq = PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

Upload an object to a bucket

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Upload an object to a bucket.

```
public static String putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {

    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
        return response.eTag();

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return "";
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {

    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
```

Upload an object to a bucket and set tags.

```
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {

    try {

        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
```



```
        .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {

    try {
        GetObjectTaggingRequest taggingRequest = GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag: obTags) {
            System.out.println("The tag key is: "+sinTag.key());
            System.out.println("The tag value is: "+sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 = PutObjectTaggingRequest.builder()
```

```
        .bucket(bucketName)
        .key(objectKey)
        .tagging(updatedTags)
        .build();

    s3.putObjectTagging(taggingRequest1);
    GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
    List<Tag> modTags = getTaggingRes2.getTagSet();
    for (Tag sinTag: modTags) {
        System.out.println("The tag key is: "+sinTag.key());
        System.out.println("The tag value is: "+sinTag.value());
    }

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

Upload an object to a bucket and set metadata.

```
    public static String putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {

        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
            return response.eTag();

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        return "";
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {

        FileInputStream fileInputStream = null;
        byte[] byteArray = null;

        try {
            File file = new File(filePath);
            byteArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(byteArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
```

```
        try {
            fileInputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

return byteArray;
}
```

Upload an object to a bucket and set an object retention value.

```
public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try{
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
        PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully placed
        on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutObject](#) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void signBucket(S3Presigner presigner, String bucketName, String
keyName) {
```

```
try {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(keyName)
        .contentType("text/plain")
        .build();

    PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10))
        .putObjectRequest(objectRequest)
        .build();

    PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
    String myURL = presignedRequest.url().toString();
    System.out.println("Presigned URL to upload a file to: " +myURL);
    System.out.println("Which HTTP method needs to be used when uploading a
file: " + presignedRequest.httpRequest().method());

    // Upload content to the Amazon S3 bucket by using this URL.
    URL url = presignedRequest.url();

    // Create the connection and use it to upload the new object by using the
presigned URL.
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setDoOutput(true);
    connection.setRequestProperty("Content-Type", "text/plain");
    connection.setRequestMethod("PUT");
    OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
    out.write("This text was uploaded as an object by using a presigned URL.");
    out.close();

    connection.getResponseCode();
    System.out.println("HTTP response code is " +
connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
```

Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3Scenario {

    public static void main(String[] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "  <bucketName> <key> <objectPath> <savePath> <toBucket>\n\n" +
            "Where:\n" +
            "  bucketName - The Amazon S3 bucket to create.\n\n" +
            "  key - The key to use.\n\n" +
            "  objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf). "+
            "  savePath - The path where the file is saved after it's downloaded (for
example, C:/AWS/book2.pdf). " +
            "  toBucket - An Amazon S3 bucket to where an object is copied to (for
example, C:/AWS/book2.pdf). ";

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4] ;

        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();

        // Create an Amazon S3 bucket.
        createBucket(s3, bucketName);

        // Update a local file to the Amazon S3 bucket.
        uploadLocalFile(s3, bucketName, key, objectPath);

        // Download the object to another local file.
        getObjectBytes (s3, bucketName, key, savePath);

        // Perform a multipart upload.
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);

        // List all objects located in the Amazon S3 bucket.
        // Show 2 ways
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName) ;

        // Copy the object to another Amazon S3 bucket
```

```

copyBucketObject (s3, bucketName, key, toBucket);

// Delete the object from the Amazon S3 bucket.
deleteObjectFromBucket(s3, bucketName, key);

// Delete the Amazon S3 bucket
deleteBucket(s3, bucketName);
System.out.println("All Amazon S3 operations were successfully performed");
s3.close();
}

// Create a bucket by using a S3Waiter object
public static void createBucket( S3Client s3Client, String bucketName) {

    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts
 */
private static void multipartUpload(S3Client s3, String bucketName, String key) {

    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()

```

```

        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * MB))).eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
        .uploadId(uploadId)
        .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * MB))).eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
// parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
        .parts(part1, part2)
        .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    // Return a byte array
    private static byte[] getObjectFile(String filePath) {

        FileInputStream fileInputStream = null;
        byte[] byteArray = null;

        try {
            File file = new File(filePath);
            byteArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(byteArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        return byteArray;
    }

    public static void getObjectBytes (S3Client s3, String bucketName, String keyName,
String path ) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path );
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        s3.putObject(objectRequest, RequestBody.fromBytes(getObjectFile(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {

        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }

```



```

    }
}

public static void anotherListExample(S3Client s3, String bucketName) {

    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));

    // Helper method to work with paginated collection of items directly
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + " size =
" + content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " + content.size());
    }
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {

    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The "+ objectKey + " was copied to "+toBucket);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
        return "";  
    }  
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

S3 Glacier examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with S3 Glacier.

Actions are code excerpts that show you how to call individual S3 Glacier functions.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 397\)](#)

Actions

Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void createGlacierVault(GlacierClient glacier, String vaultName ) {  
  
    try {  
        CreateVaultRequest vaultRequest = CreateVaultRequest.builder()  
            .vaultName(vaultName)  
            .build();  
  
        CreateVaultResponse createVaultResult = glacier.createVault(vaultRequest);  
        System.out.println("The URI of the new vault is " +  
            createVaultResult.location());  
  
    } catch(GlacierException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateVault](#) in *AWS SDK for Java 2.x API Reference*.

Delete a vault

The following code example shows how to delete an Amazon S3 Glacier vault.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
    try {
        DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        glacier.deleteVault(delVaultRequest);
        System.out.println("The vault was deleted!");
    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteVault](#) in *AWS SDK for Java 2.x API Reference*.

Delete an archive

The following code example shows how to delete an Amazon S3 Glacier archive.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
    try {
        DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        glacier.deleteVault(delVaultRequest);
        System.out.println("The vault was deleted!");
    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [DeleteArchive](#) in *AWS SDK for Java 2.x API Reference*.

List vaults

The following code example shows how to list Amazon S3 Glacier vaults.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllVault(GlacierClient glacier) {

    boolean listComplete = false;
    String newMarker = null;
    int totalVaults = 0;
    System.out.println("Your Amazon Glacier vaults:");

    try {

        while (!listComplete) {
            ListVaultsResponse response = null;
            if (newMarker != null) {
                ListVaultsRequest request = ListVaultsRequest.builder()
                    .marker(newMarker)
                    .build();

                response = glacier.listVaults(request);
            } else {
                ListVaultsRequest request = ListVaultsRequest.builder()
                    .build();
                response = glacier.listVaults(request);
            }

            List<DescribeVaultOutput> vaultList = response.vaultList();
            for (DescribeVaultOutput v: vaultList) {
                totalVaults += 1;
                System.out.println("* " + v.vaultName());
            }

            // Check for further results.
            newMarker = response.marker();
            if (newMarker == null) {
                listComplete = true;
            }
        }

        if (totalVaults == 0) {
            System.out.println("No vaults found.");
        }

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListVaults](#) in *AWS SDK for Java 2.x API Reference*.

Retrieve a vault inventory

The following code example shows how to retrieve an Amazon S3 Glacier vault inventory.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```

    public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {

        try {

            JobParameters job = JobParameters.builder()
                .type("inventory-retrieval")
                .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                .jobParameters(job)
                .accountId(accountId)
                .vaultName(vaultName)
                .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " +response.jobId());
            System.out.println("The relative URI path of the job is: "
+response.location());
            return response.jobId();

        } catch(GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {

        try{
            boolean finished = false;
            String jobStatus;
            int yy=0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                    .jobId(jobId)
                    .accountId(account)
                    .vaultName(name)
                    .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
            }
        }
    }

```

```
        yy++;
    }

    System.out.println("Job has Succeeded");
    GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
        .jobId(jobId)
        .vaultName(name)
        .accountId(account)
        .build();

    ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
    // Write the data to a local file.
    byte[] data = objectBytes.asByteArray();
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from a Glacier vault");
    os.close();

} catch(GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [InitiateJob](#) in *AWS SDK for Java 2.x API Reference*.

Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {

    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String computeSHA256(File inputFile) {

    try {
```

```
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256: %s",
nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;

    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
```

```

        ioe.getMessage());
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
    return sb.toString().toLowerCase();
}

```

- For API details, see [UploadArchive](#) in *AWS SDK for Java 2.x API Reference*.

Amazon SES examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 404\)](#)

Actions

List email templates

The following code example shows how to list Amazon SES email templates.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllTemplates(SesV2Client sesv2Client) {  
  
    try {  
        ListEmailTemplatesRequest templatesRequest =  
ListEmailTemplatesRequest.builder()  
            .pageSize(1)  
            .build();  
  
        ListEmailTemplatesResponse response =  
sesv2Client.listEmailTemplates(templatesRequest);  
        response.templatesMetadata().forEach(template ->  
            System.out.println("Template name: " + template.templateName()));  
  
    } catch (SesV2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListTemplates](#) in *AWS SDK for Java 2.x API Reference*.

List identities

The following code example shows how to list Amazon SES identities.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listSESIIdentities(SesClient client) {
```

```
try {
    ListIdentitiesResponse identitiesResponse = client.listIdentities();
    List<String> identities = identitiesResponse.getIdentities();
    for (String identity: identities) {
        System.out.println("The identity is "+identity);
    }
} catch (SesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListIdentities](#) in *AWS SDK for Java 2.x API Reference*.

Send email

The following code example shows how to send email with Amazon SES.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void send(SesClient client,
                      String sender,
                      String recipient,
                      String subject,
                      String bodyHTML
) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
            "using the AWS SDK for Java...");
    }
```

```
        client.sendEmail(emailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendemailAttachment(SesClient client,
                                       String sender,
                                       String recipient,
                                       String subject,
                                       String bodyText,
                                       String bodyHTML,
                                       String fileLocation) throws AddressException,
MessagingException, IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

    // Create a multipart/alternative child container.
    MimeMultipart msgBody = new MimeMultipart("alternative");

    // Create a wrapper for the HTML and text parts.
    MimeBodyPart wrap = new MimeBodyPart();

    // Define the text part.
    MimeBodyPart textPart = new MimeBodyPart();
    textPart.setContent(bodyText, "text/plain; charset=UTF-8");

    // Define the HTML part.
    MimeBodyPart htmlPart = new MimeBodyPart();
    htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

    // Add the text and HTML parts to the child container.
    msgBody.addBodyPart(textPart);
    msgBody.addBodyPart(htmlPart);

    // Add the child container to the wrapper object.
    wrap.setContent(msgBody);

    // Create a multipart/mixed parent container.
    MimeMultipart msg = new MimeMultipart("mixed");

    // Add the parent container to the message.
    message.setContent(msg);
    msg.addBodyPart(wrap);

    // Define the attachment.
    MimeBodyPart att = new MimeBodyPart();
    DataSource fds = new ByteArrayDataSource(fileContent, "application/
vnd.openxmlformats-officedocument.spreadsheetml.sheet");
    att.setDataHandler(new DataHandler(fds));

    String reportName = "WorkReport.xls";
```

```
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
    RawMessage rawMessage = RawMessage.builder()
        .data(data)
        .build();

    SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
        .rawMessage(rawMessage)
        .build();

    client.sendRawEmail(rawEmailRequest);

} catch (SesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Email sent using SesClient with attachment");
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

Send templated email

The following code example shows how to send templated email with Amazon SES.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void send(SesV2Client client, String sender, String recipient, String
templateName){

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
    Specify both name and favorite animal (favoriteanimal) in your code when
defining the Template object.
    If you don't specify all the variables in the template, Amazon SES doesn't
send the email.
    */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
```

```
        "  \"name\": \"Jason\"\n", " +  
        "  \"favoriteanimal\": \"Cat\"\n" +  
        "}")  
    .build();  
  
    EmailContent emailContent = EmailContent.builder()  
        .template(myTemplate)  
        .build();  
  
    SendEmailRequest emailRequest = SendEmailRequest.builder()  
        .destination(destination)  
        .content(emailContent)  
        .fromEmailAddress(sender)  
        .build();  
  
    try {  
        System.out.println("Attempting to send an email based on a template using  
the AWS SDK for Java (v2)...");  
        client.sendEmail(emailRequest);  
        System.out.println("email based on a template was sent");  
    } catch (SesV2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for Java 2.x API Reference*.

Amazon SES API v2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES API v2.

Actions are code excerpts that show you how to call individual Amazon SES API v2 functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon SES API v2 functions.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 408\)](#)

Actions

Send an email

The following code example shows how to send an Amazon SES API v2 email.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Sends a message.

```
public static void send(SesV2Client client,  
                      String sender,
```

```
        String recipient,
        String subject,
        String bodyHTML
    ){

        Destination destination = Destination.builder()
            .toAddresses(recipient)
            .build();

        Content content = Content.builder()
            .data(bodyHTML)
            .build();

        Content sub = Content.builder()
            .data(subject)
            .build();

        Body body = Body.builder()
            .html(content)
            .build();

        Message msg = Message.builder()
            .subject(sub)
            .body(body)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(msg)
            .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
            .destination(destination)
            .content(emailContent)
            .fromEmailAddress(sender)
            .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

Amazon SNS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SNS.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 410\)](#)
- [Scenarios \(p. 420\)](#)

Actions

Add tags to a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void addTopicTags(SnsClient snsClient, String topicArn) {  
  
    try {  
        Tag tag = Tag.builder()  
            .key("Team")  
            .value("Development")  
            .build();  
  
        Tag tag2 = Tag.builder()  
            .key("Environment")  
            .value("Gamma")  
            .build();  
  
        List<Tag> tagList = new ArrayList<>();  
        tagList.add(tag);  
        tagList.add(tag2);  
  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(topicArn)  
            .tags(tagList)  
            .build();  
  
        snsClient.tagResource(tagResourceRequest);  
        System.out.println("Tags have been added to "+topicArn);  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [TagResource](#) in *AWS SDK for Java 2.x API Reference*.

Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void checkPhone(SnsClient snsClient, String phoneNumber) {  
    try {  
        CheckIfPhoneNumberIsOptedOutRequest request =  
CheckIfPhoneNumberIsOptedOutRequest.builder()  
            .phoneNumber(phoneNumber)  
            .build();  
  
        CheckIfPhoneNumberIsOptedOutResponse result =  
snsClient.checkIfPhoneNumberIsOptedOut(request);  
        System.out.println(result.isOptedOut() + "Phone Number " + phoneNumber + "  
has Opted Out of receiving sns messages." +  
            "\n\nStatus was " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

Confirm an endpoint owner wants to receive messages

The following code example shows how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void confirmSub(SnsClient snsClient, String subscriptionToken, String  
topicArn ) {  
    try {  
        ConfirmSubscriptionRequest request = ConfirmSubscriptionRequest.builder()  
            .token(subscriptionToken)  
            .topicArn(topicArn)  
            .build();  
  
        ConfirmSubscriptionResponse result =  
snsClient.confirmSubscription(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n" +  
result.subscriptionArn());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ConfirmSubscription](#) in *AWS SDK for Java 2.x API Reference*.

Create a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {
    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for Java 2.x API Reference*.

Delete a subscription

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
            result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " + request.subscriptionArn());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [Unsubscribe](#) in *AWS SDK for Java 2.x API Reference*.

Delete a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {  
  
    try {  
        DeleteTopicRequest request = DeleteTopicRequest.builder()  
            .topicArn(topicArn)  
            .build();  
  
        DeleteTopicResponse result = snsClient.deleteTopic(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for Java 2.x API Reference*.

Get the properties of a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn ) {  
  
    try {  
        GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()  
            .topicArn(topicArn)  
            .build();  
  
        GetTopicAttributesResponse result = snsClient.getTopicAttributes(request);  
        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()  
+ "\n\nAttributes: \n\n" + result.attributes());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Get the settings for sending SMS messages

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getSNSAttrutes(SnsClient snsClient,String topicArn ) {  
  
    try {  
        GetSubscriptionAttributesRequest request =  
GetSubscriptionAttributesRequest.builder()  
        .subscriptionArn(topicArn)  
        .build();  
  
        // Get the Subscription attributes  
        GetSubscriptionAttributesResponse res =  
snsClient.getSubscriptionAttributes(request);  
        Map<String, String> map = res.attributes();  
  
        // Iterate through the map  
        Iterator iter = map.entrySet().iterator();  
        while (iter.hasNext()) {  
            Map.Entry entry = (Map.Entry) iter.next();  
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +  
entry.getValue());  
        }  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    System.out.println("\n\nStatus was good");  
}
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for Java 2.x API Reference*.

List opted out phone numbers

The following code example shows how to list phone numbers that are opted out of receiving Amazon SNS messages.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listOpts( SnsClient snsClient) {  
  
    try {  
        ListPhoneNumbersOptedOutRequest request =  
ListPhoneNumbersOptedOutRequest.builder().build();  
        ListPhoneNumbersOptedOutResponse result =  
snsClient.listPhoneNumbersOptedOut(request);  
        System.out.println("Status is " + result.sdkHttpResponse().statusCode() +  
"\n\nPhone Numbers: \n\n" + result.phoneNumbers());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListPhoneNumbersOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

List the subscribers of a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listSNSSubscriptions( SnsClient snsClient) {  
  
    try {  
        ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()  
            .build();  
  
        ListSubscriptionsResponse result = snsClient.listSubscriptions(request);  
        System.out.println(result.subscriptions());  
  
    } catch (SnsException e) {  
  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Java 2.x API Reference*.

List topics

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listSNSTopics(SnsClient snsClient) {  
  
    try {  
        ListTopicsRequest request = ListTopicsRequest.builder()  
            .build();  
  
        ListTopicsResponse result = snsClient.listTopics(request);  
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +  
"\n\nTopics\n\n" + result.topics());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Java 2.x API Reference*.

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

Publish to a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

Set a filter policy

The following code example shows how to set an Amazon SNS filter policy.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetSubscriptionAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Set the default settings for sending SMS messages

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public class SetSMSAttributes {
    public static void main(String[] args) {
```

```
HashMap<String, String> attributes = new HashMap<>(1);
attributes.put("DefaultSMSType", "Transactional");
attributes.put("UsageReportsS3Bucket", "janbucket" );

SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
setSNSAttributes(snsClient, attributes);
snsClient.close();
}

public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String>
attributes) {

    try {
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ". Status
was " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetSmsAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {

    try {
        SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
+ " updated " + request.attributeName() + " to " +
request.attributeValue());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Subscribe a Lambda function to a topic

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("lambda")
            .endpoint(lambdaArn)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        return result.subscriptionArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

Subscribe an HTTP endpoint to a topic

The following code example shows how to subscribe an HTTP or HTTPS endpoint so it receives notifications from an Amazon SNS topic.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void subHTTPS(SnsClient snsClient, String topicArn, String url ) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("http")
            .endpoint(url)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
    }
```



```
final String usage = "\n" +
    "Usage: " +
    "  <token>\n\n" +
    "Where:\n" +
    "  token - The name of the FIFO topic. \n\n" +
    "  platformApplicationArn - The ARN value of platform application. You can
get this value from the AWS Management Console. \n\n";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String token = args[0];
String platformApplicationArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

createEndpoint(snsClient, token, platformApplicationArn);
}

public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn){

    System.out.println("Creating platform endpoint with token " + token);

    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
            .token(token)
            .platformApplicationArn(platformApplicationArn)
            .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " + response.endpointArn());
    } catch ( SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create a FIFO topic and FIFO queues. Subscribe the queues to the topic.

```
public static void main(String[] args) {

    final String usage = "\n" +
        "Usage: " +
        "  <topicArn>\n\n" +
```

```

        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic. \n\n" +
        "    fifoQueueARN - The ARN value of a SQS FIFO queue. You can get this
value from the AWS Management Console. \n\n";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String fifoTopicName = "PriceUpdatesTopic3.fifo";
    String fifoQueueARN = "arn:aws:sqs:us-east-1:814548047983:MyPriceSQS.fifo";
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    createFIFO(snsClient, fifoTopicName, fifoQueueARN);
}

public static void createFIFO(SnsClient snsClient, String topicName, String
queueARN) {

    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("FifoTopic", "true");
        topicAttributes.put("ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is"+topicArn);

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS FIFO queues can receive notifications from an Amazon SNS
FIFO topic.
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicArn)
            .endpoint(queueARN)
            .protocol("sqs")
            .build();

        snsClient.subscribe(subscribeRequest);
        System.out.println("The topic is subscribed to the queue.");

        // Compose and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String payload = "{\"product\": 214, \"price\": 79.99}";
        String groupId = "PID-214";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()

```

```
        .topicArn(topicArn)
        .subject(subject)
        .message(payload)
        .messageGroupId(groupId)
        .messageDeduplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

        snsClient.publish(pubRequest);
        System.out.println("Message was published to "+topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Publish SMS messages to a topic

The following code example shows how to:

- Create an Amazon SNS topic.
- Subscribe phone numbers to the topic.
- Publish SMS messages to the topic so that all subscribed phone numbers receive the message at once.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Create a topic and return its ARN.

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Subscribe an endpoint to a topic.

```
public static void subTextSNS( SnsClient snsClient, String topicArn, String
phoneNumber) {

    try {
        SubscribeRequest request = SubscribeRequest.builder()
```

```
        .protocol("sms")
        .endpoint(phoneNumber)
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
        Status is " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Set attributes on the message, such as the ID of the sender, the maximum price, and its type. Message attributes are optional.

```
public class SetSMSAttributes {
    public static void main(String[] args) {

        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket" );

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String>
attributes) {

        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was " + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publish a message to a topic. The message is sent to every subscriber.

```
    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
                .build();
```

```
        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Amazon SQS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SQS.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 425\)](#)

Actions

Create a queue

The following code example shows how to create an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createQueue(SqsClient sqsClient, String queueName ) {

    try {
        System.out.println("\nCreate Queue");

        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);

        System.out.println("\nGet queue url");

        GetQueueUrlResponse getQueueUrlResponse =
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

    }
    return "";
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl ) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }

    System.out.println("\nSend message");
    try {
        sqsClient.sendMessage(SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello world!")
            .delaySeconds(10)
            .build());

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
            .build();
    }
}

```

```
        sqsClient.sendMessageBatch(sendMessageBatchRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nReceive messages");
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .maxNumberOfMessages(5)
            .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Java 2.x API Reference*.

Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
        .queueUrl(queueUrl)
        .receiptHandle(message.receiptHandle())
        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Java 2.x API Reference*.

Delete a queue

The following code example shows how to delete an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
        .queueName(queueName)
        .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
        .queueUrl(queueUrl)
        .build();

        sqsClient.deleteQueue(deleteQueueRequest);
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Java 2.x API Reference*.

Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
GetQueueUrlResponse getQueueUrlResponse =
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
return getQueueUrlResponse.queueUrl();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Java 2.x API Reference*.

List queues

The following code example shows how to list Amazon SQS queues.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [ListQueues](#) in *AWS SDK for Java 2.x API Reference*.

Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Java 2.x API Reference*.

Send a batch of messages to a queue

The following code example shows how to send a batch of messages to an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
    sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- For API details, see [SendMessageBatch](#) in *AWS SDK for Java 2.x API Reference*.

Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {

    try {
        CreateQueueRequest request = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();
        sqsClient.createQueue(request);

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
```

```
        .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(message)
            .delaySeconds(5)
            .build();

        sqsClient.sendMessage(sendMsgRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Java 2.x API Reference*.

Secrets Manager examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Secrets Manager.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 431\)](#)

Actions

Create a secret

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createNewSecret( SecretsManagerClient secretsClient, String
secretName, String secretValue) {

    try {
        CreateSecretRequest secretRequest = CreateSecretRequest.builder()
            .name(secretName)
            .description("This secret was created by the AWS Secret Manager Java
API")
            .secretString(secretValue)
            .build();

        CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
```

```
        return secretResponse.arn();
    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Java 2.x API Reference*.

Delete a secret

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void deleteSpecificSecret(SecretsManagerClient secretsClient, String
secretName) {
    try {
        DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
            .secretId(secretName)
            .build();

        secretsClient.deleteSecret(secretRequest);
        System.out.println(secretName + " is deleted.");
    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSecret](#) in *AWS SDK for Java 2.x API Reference*.

Describe a secret

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void describeGivenSecret(SecretsManagerClient secretsClient, String
secretName) {
    try {
        DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
            .secretId(secretName)
            .build();

        DescribeSecretResponse secretResponse =
            secretsClient.describeSecret(secretRequest);
    }
```

```
Instant lastChangedDate = secretResponse.lastChangedDate();

// Convert the Instant to readable date.
DateTimeFormatter formatter =
    DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
        .withLocale( Locale.US )
        .withZone( ZoneId.systemDefault() );

formatter.format( lastChangedDate );
System.out.println("The date of the last change to " + secretResponse.name()
+" is " + lastChangedDate );

} catch (SecretsManagerException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Java 2.x API Reference*.

Get a secret value

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getValue(SecretsManagerClient secretsClient,String secretName) {

    try {
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
        String secret = valueResponse.secretString();
        System.out.println(secret);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

List secrets

The following code example shows how to list Secrets Manager secrets.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listAllSecrets(SecretsManagerClient secretsClient) {
```

```
try {
    ListSecretsResponse secretsResponse = secretsClient.listSecrets();
    List<SecretListEntry> secrets = secretsResponse.secretList();
    for (SecretListEntry secret: secrets) {
        System.out.println("The secret name is "+secret.name());
        System.out.println("The secret description is "+secret.description());
    }
} catch (SecretsManagerException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Java 2.x API Reference*.

Put a value in a secret

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

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void updateMySecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {

    try {
        UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()
            .secretId(secretName)
            .secretString(secretValue)
            .build();

        secretsClient.updateSecret(secretRequest);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

Step Functions examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Step Functions.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 435\)](#)

Actions

Create a state machine

The following code example shows how to create a Step Functions state machine.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String createMachine( SfnClient sfnClient, String roleARN, String
stateMachineName, String jsonFile) {

    String json = getJSONString(jsonFile);
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String getJSONString(String path) {
    try {
        JSONParser parser = new JSONParser();
        JSONObject data = (JSONObject) parser.parse(new FileReader(path)); //path to
the JSON file.
        return data.toJSONString();

    } catch (IOException | org.json.simple.parser.ParseException e) {
        e.printStackTrace();
    }
    return "";
}
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

Delete a state machine

The following code example shows how to delete a Step Functions state machine.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).


```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        System.out.println(stateMachineArn + " was successfully deleted.");
    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

List state machine runs

The following code example shows how to list Step Functions state machine runs.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
        .executionArn(exeARN)
        .maxResults(10)
        .build();

        GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event: events) {
            System.out.println("The event type is "+event.type().toString());
        }
    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListExecutions](#) in *AWS SDK for Java 2.x API Reference*.

List state machines

The following code example shows how to list Step Functions state machines.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void listMachines(SfnClient sfnClient) {  
    try {  
        ListStateMachinesResponse response = sfnClient.listStateMachines();  
        List<StateMachineListItem> machines = response.stateMachines();  
        for (StateMachineListItem machine : machines) {  
            System.out.println("The name of the state machine is:  
"+machine.name());  
            System.out.println("The ARN value is : "+machine.stateMachineArn());  
        }  
    } catch (SfnException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Java 2.x API Reference*.

Start a state machine run

The following code example shows how to start a Step Functions state machine run.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,  
String jsonFile) {  
    String json = getJSONString(jsonFile);  
    UUID uuid = UUID.randomUUID();  
    String uuidValue = uuid.toString();  
    try {  
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()  
            .input(json)  
            .stateMachineArn(stateMachineArn)  
            .name(uuidValue)  
            .build();  
        StartExecutionResponse response =  
sfnClient.startExecution(executionRequest);  
        return response.executionArn();  
    } catch (SfnException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}  
  
private static String getJSONString(String path) {  
    try {  
        JSONParser parser = new JSONParser();  
        JSONObject data = (JSONObject) parser.parse(new FileReader(path));  
        //path to  
the JSON file.  
        String json = data.toJSONString();  
    } catch (Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

```
        return json;

    } catch (IOException | org.json.simple.parser.ParseException e) {
        e.printStackTrace();
    }
    return "";
}
```

- For API details, see [StartExecution](#) in *AWS SDK for Java 2.x API Reference*.

Amazon Textract examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Textract.

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

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions \(p. 438\)](#)

Actions

Analyze a document

The following code example shows how to analyze a document using Amazon Textract.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {

    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        List<FeatureType> featureTypes = new ArrayList<FeatureType>();
        featureTypes.add(FeatureType.FORMS);
        featureTypes.add(FeatureType.TABLES);

        AnalyzeDocumentRequest analyzeDocumentRequest =
        AnalyzeDocumentRequest.builder()
            .featureTypes(featureTypes)
            .document(myDoc)
            .build();
```

```
        AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
        List<Block> docInfo = analyzeDocument.blocks();
        Iterator<Block> blockIterator = docInfo.iterator();

        while(blockIterator.hasNext()) {
            Block block = blockIterator.next();
            System.out.println("The block type is " +block.blockType().toString());
        }

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for Java 2.x API Reference*.

Detect text in a document

The following code example shows how to detect text in a document using Amazon Textract.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

Detect text from an input document.

```
public static void detectDocText(TextractClient textractClient,String sourceDoc) {

    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation
        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is "
+documentMetadata.pages());

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

Detect text from a document located in an Amazon S3 bucket.

```
public static void detectDocTextS3 (TextractClient textractClient, String  
bucketName, String docName) {  
  
    try {  
        S3Object s3Object = S3Object.builder()  
            .bucket(bucketName)  
            .name(docName)  
            .build();  
  
        // Create a Document object and reference the s3Object instance  
        Document myDoc = Document.builder()  
            .s3Object(s3Object)  
            .build();  
  
        DetectDocumentTextRequest detectDocumentTextRequest =  
DetectDocumentTextRequest.builder()  
            .document(myDoc)  
            .build();  
  
        DetectDocumentTextResponse textResponse =  
textractClient.detectDocumentText(detectDocumentTextRequest);  
        for (Block block: textResponse.blocks()) {  
            System.out.println("The block type is " +block.blockType().toString());  
        }  
  
        DocumentMetadata documentMetadata = textResponse.documentMetadata();  
        System.out.println("The number of pages in the document is "  
+documentMetadata.pages());  
  
    } catch (TextractException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectDocumentText](#) in *AWS SDK for Java 2.x API Reference*.

Start asynchronous analysis of a document

The following code example shows how to start asynchronous analysis of a document using Amazon Textract.

SDK for Java 2.x

Tip

To learn how to set up and run this example, see [GitHub](#).

```
public static String startDocAnalysisS3 (TextractClient textractClient, String  
bucketName, String docName) {  
  
    try {  
        List<FeatureType> myList = new ArrayList<>();  
        myList.add(FeatureType.TABLES);  
    }  
}
```

```
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
            .featureTypes(myList)
            .build();

        StartDocumentAnalysisResponse response =
textextractClient.startDocumentAnalysis(documentAnalysisRequest);

        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TextextractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "" ;
}

private static String getJobResults(TextextractClient textextractClient, String jobId) {

    boolean finished = false;
    int index = 0 ;
    String status = "" ;

    try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                .jobId(jobId)
                .maxResults(1000)
                .build();

            GetDocumentAnalysisResponse response =
textextractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }

        return status;

    } catch( InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for Java 2.x API Reference*.

Cross-service examples using SDK for Java 2.x

The following sample applications use the AWS SDK for Java 2.x to work across multiple AWS services.

Examples

- [Build an application to submit data to a DynamoDB table \(p. 442\)](#)
- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 442\)](#)
- [Build a publish and subscription application that translates messages \(p. 443\)](#)
- [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 443\)](#)
- [Create a dynamic web application to track DynamoDB data \(p. 443\)](#)
- [Create an Amazon Relational Database Service item tracker \(p. 443\)](#)
- [Create an Amazon Redshift item tracker \(p. 444\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 444\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 444\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 445\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 445\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 445\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 446\)](#)

Build an application to submit data to a DynamoDB table

SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- Amazon SNS

Create an Amazon Lex Chatbot within a web application to engage your web site visitors

SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Comprehend
- Amazon Lex

- Amazon Translate

Build a publish and subscription application that translates messages

SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on [GitHub](#).

Services used in this example

- Amazon SNS
- Amazon Translate

Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API

SDK for Java 2.x

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Comprehend
- Amazon SQS

Create a dynamic web application to track DynamoDB data

SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- Amazon SES

Create an Amazon Relational Database Service item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

Services used in this example

- Amazon RDS
- Amazon SES

Create an Amazon Redshift item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

Services used in this example

- Amazon Redshift
- Amazon SES

Detect PPE in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detect objects in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detect people and objects in a video with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Use API Gateway to invoke a Lambda function

SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Use Step Functions to invoke Lambda functions

SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB

- Lambda
- Amazon SES
- Step Functions

Use scheduled events to invoke a Lambda function

SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Security for the AWS SDK for Java

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.

Topics

- [Data protection in AWS SDK for Java 2.x \(p. 447\)](#)
- [AWS SDK for Java support for TLS \(p. 448\)](#)
- [Identity and Access Management for this AWS Product or Service \(p. 448\)](#)
- [Compliance validation for the AWS SDK for Java \(p. 449\)](#)
- [Resilience for this AWS Product or Service \(p. 449\)](#)
- [Infrastructure Security for this AWS Product or Service \(p. 450\)](#)

Data protection in AWS SDK for Java 2.x

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.
- 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 this

AWS product or service or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into this AWS product or service 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.

AWS SDK for Java support for TLS

The following information applies only to Java SSL implementation (the default SSL implementation in the AWS SDK for Java). If you're using a different SSL implementation, see your specific SSL implementation to learn how to enforce TLS versions.

How to check the TLS version

Consult your Java virtual machine (JVM) provider's documentation to determine which TLS versions are supported on your platform. For some JVMs, the following code will print which SSL versions are supported.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()))
```

To see the SSL handshake in action and what version of TLS is used, you can use the system property **javax.net.debug**.

```
java app.jar -Djavax.net.debug=ssl
```

Enforcing a minimum TLS version

The SDK always prefers the latest TLS version supported by the platform and service. If you wish to enforce a specific minimum TLS version, consult your JVM's documentation.

For OpenJDK-based JVMs, you can use the system property `jdk.tls.client.protocols`.

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Consult your JVM's documentation for the supported values of PROTOCOLS.

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](#)".

Compliance validation for the AWS SDK for Java

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](#)".

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](#)".

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 describe show 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](#)."

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](#)."

Document history

This topic describes important changes to the AWS SDK for Java Developer Guide over the course of its history.

This documentation was last built on: 2022-09-13

| Change | Description | Date |
|---|--|-------------------|
| the section called "AWS database services" (p. 73) | Added overview section of AWS database services and the SDK for Java 2.x | 13 September 2022 |
| EC2-Classic Networking is Retiring | EC2-Classic is retiring on August 15, 2022 | 28 July 2022 |
| the section called "Additional setup information" (p. 19) | Update to dependency required for single sign-on authentication | 18 July 2022 |
| the section called "Enforcing a minimum TLS version" (p. 448) | Update TLS security information | 8 April 2022 |
| the section called "Additional setup information" (p. 19) | Added more information about setting up and using credentials | 22 February 2021 |
| the section called "Setting up a GraalVM Native Image project" (p. 18) | New topic for setting up a GraalVM Native Image project | 18 February 2021 |
| the section called "Waiters" (p. 69) | Waiters released; added topic for the new feature | 30 September 2020 |
| the section called "SDK Metrics" (p. 56) | Metrics released; added topic for the new feature | 17 August 2020 |
| the section called " Amazon Pinpoint " (p. 161), the section called " Amazon Cognito " (p. 146), the section called " Amazon Simple Notification Service " (p. 169) | Added example topics for Amazon Pinpoint, Amazon Cognito, and Amazon SNS | 30 May 2020 |
| the section called "Optimizing cold start performance for AWS Lambda" (p. 40) | Added AWS Lambda function performance topic | 29 May 2020 |
| the section called "Setting the JVM TTL for DNS name lookups" (p. 49) | Added JVM TTL DNS caching topic | 27 April 2020 |
| the section called "Setting up an Apache Maven project" (p. 13), the section called "Setting up a Gradle project" (p. 17) | New Maven and Gradle set up topics | 21 April 2020 |

| Change | Description | Date |
|--|---|------------------|
| the section called "Mapping items in DynamoDB tables" (p. 99) | Added DynamoDB enhanced client topic | 20 April 2020 |
| the section called "Enforcing a minimum TLS version" (p. 448) | Added TLS 1.2 to security section | 19 March 2020 |
| the section called "Subscribing to Amazon Kinesis Data Streams" (p. 152) | Added Kinesis stream examples | 2 August 2018 |
| the section called "Pagination" (p. 61) | Added auto pagination topic | 5 April 2018 |
| Working with AWS services (p. 72) | Added example topics for IAM, Amazon EC2, CloudWatch and DynamoDB | 29 December 2017 |
| the section called " Amazon Simple Storage Service (S3)" (p. 75) | Added getObject example for Amazon S3 | 7 August 2017 |
| the section called "Asynchronous programming" (p. 50) | Added async topic | 4 August 2017 |
| GA release of the AWS SDK for Java 2.x | AWS SDK for Java version 2 (v2) released | 28 June 2017 |