AWS SDK for Java

Migration Guide

AWS SDK for Java: Migration Guide

Table of Contents

What Is the AWS SDK for Java 2.x Migration Guide?

The AWS SDK for Java 2.x is a major rewrite of the 1.11.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 guide 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 1.11.x.

For more details about the new features and to see specific code examples, see the AWS SDK for Java 2.x Developer Guide.

Topics:

What's New in the AWS SDK for Java 2.x

This topic briefly describes the major new features in AWS SDK for Java 2.x. For details about each feature and to see examples of how to use them, see the links provided or the AWS SDK for Java 2.x Developer Guide.

- You can configure your own HTTP clients. See HTTP Transport Configuration in the *AWS SDK for Java 2.x Developer Guide* for an example.
- Async clients are now truly nonblocking and return `CompletableFuture` objects. See Basic Async in the *AWS SDK for Java 2.x Developer Guide*.
- 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 example in the *AWS SDK for Java 2.x Developer Guide*.
- SDK start time performance for AWS Lambda functions is improved. See SDK Start Time Performance Improvements in the *AWS SDK for Java 2.x Developer Guide* for details.
- Version 2.x supports a new shorthand method for creating requests.

Example

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

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 1.11.x and DynamoDB from version 2.1.0.

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>1.11.428</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.1.0</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>
```

What's Different between the SDK for Java 1.11.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 1.11.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 complete list of libraries.

If your application depends on these libraries, see Side by Side to learn how to configure your pom.xml to use both 1.11.x and 2.x. Refer to the SDK for Java 2.x changelog for updates about these libraries.

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.x.0</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 1.11.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 1.11.x and 2.x, see the AWS SDK for Java 2.x changelog.

Setter Methods

In the 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 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 Changes.

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 1.11.x exceptions, see Exception Changes Details.

Service-Specific Changes

Amazon S3 Operation Name Changes

Many of the operation names for the Amazon S3 client have changed in the 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 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.

1.11.x	2.x
AWS_CBOR_DISABLED	CBOR_ENABLED
AWS_ION_BINARY_DISABLE	BINARY_ION_ENABLED

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

1.11.x	2.x
com.amazonaws.sdk.disableEc2Metadata	aws.disableEc2Metadata
com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	aws.ec2MetadataServiceEndpoint
com.amazonaws.sdk.disableCbor	aws.cborEnabled
com.amazonaws.sdk.disableIonBinary	aws.binaryIonEnabled

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

1.11.x
com.amazonaws.sdk.disableCertChecking
com.amazonaws.sdk.enableDefaultMetrics
com.amazonaws.sdk.enableThrottledRetry
com.amazonaws.regions.RegionUtils.fileOverride
com.amazonaws.regions.RegionUtils.disableRemote
com.amazonaws.services.s3.disableImplicitGlobalClients
com.amazonaws.sdk.enableInRegionOptimizedMode

- 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 1.11.x and 2.x of the 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 Working with AWS Credentials in the *AWS SDK for Java 2.x Developer Guide*
- 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 `ProfileCredentialsProvier.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 1.11.x and 2.x

Class name changes

1.11.x: com.amazonaws.auth.AWSCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.AwsCredentialsProvider
1.11.x: com.amazonaws.auth.DefaultAWSCredentialsProviderChain **2.x:** software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
1.11.x: com.amazonaws.auth.AWSStaticCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.StaticCredentialsProvider
1.11.x: com.amazonaws.auth.EnvironmentVariableCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider
1.11.x: com.amazonaws.auth.SystemPropertiesCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider
1.11.x: com.amazonaws.auth.ProfileCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider
1.11.x: com.amazonaws.auth.ContainerCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider
1.11.x: com.amazonaws.auth.InstanceProfileCredentialsProvider **2.x:** software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider
1.11.x: com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider

2.x: software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider
1.11.x: com.amazonaws.auth.STSSessionCredentialsProvider **2.x:** software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider
1.11.x: com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider **2.x:** software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider
1.11.x: com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper **2.x:** software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider or software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider
1.11.x: com.amazonaws.services.s3.S3CredentialsProviderChain **2.x:** software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider or software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider
1.11.x: com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider **2.x:** Not Supported
1.11.x: com.amazonaws.auth.PropertiesFileCredentialsProvider **2.x:** Not Supported

Method name changes

1.11.x	2.x
AWSCredentialsProvider.getCredentials	AwsCredentialsProvider.resolveCredentials
DefaultAWSCredentialsProviderChain.getInstance	Not Supported
AWSCredentialsProvider.getInstance	Not Supported
AWSCredentialsProvider.refresh	Not Supported

Environment variable name changes

1.11.x	2.x
AWS_ACCESS_KEY	AWS_ACCESS_KEY_ID
AWS_SECRET_KEY	AWS_SECRET_ACCESS_KEY
AWS_CREDENTIAL_PROFILES_FILE	AWS_SHARED_CREDENTIALS_FILE

System property name changes

1.11.x	2.x
aws.secretKey	aws.secretAccessKey
com.amazonaws.sdk.disableEc2Metadata	aws.disableEc2Metadata
com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	aws.ec2MetadataServiceEndpoint

Region Class Name Changes

This section describes the changes implemented in the 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 1.11.x and 2.x of the 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

1.11.x	2.x
Regions.fromName	Region.of
Regions.getName	Region.id
Regions.getDescription	Not Supported
Regions.getCurrentRegion	Not Supported
Regions.DEFAULT_REGION	Not Supported
Regions.name	Not Supported

Region class method changes

1.11.x	2.x
Region.getName	Region.id
Region.hasHttpsEndpoint	Not Supported
Region.hasHttpEndpoint	Not Supported
Region.getAvailableEndpoints	Not Supported
Region.createClient	Not Supported

RegionMetadata class method changes

1.11.x	2.x
RegionMetadata.getName	RegionMetadata.name
RegionMetadata.getDomain	RegionMetadata.domain
RegionMetadata.getPartition	RegionMetadata.partition

ServiceMetadata class method changes

1.11.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 1.11.x and 2.x.

This table maps the exception class name changes.

1.11.x: com.amazonaws.SdkBaseException and com.amazonaws.AmazonClientException **2.x:** software.amazon.awssdk.core.exception.SdkException
1.11.x: com.amazonaws.SdkClientException **2.x:** software.amazon.awssdk.core.exception.SdkClientException
1.11.x: com.amazonaws.AmazonServiceException **2.x:** software.amazon.awssdk.awscore.exception.AwsServiceException

The following table maps the methods on exception classes between version 1.11.x and 2.x.

1.11.x: AmazonServiceException.getRequestId **2.x:** SdkServiceException.requestId
1.11.x: AmazonServiceException.getServiceName **2.x:** AwsServiceException.awsErrorDetails().serviceName
1.11.x: AmazonServiceException.getErrorCode **2.x:** AwsServiceException.awsErrorDetails().errorCode
1.11.x: AmazonServiceException.getErrorMessage **2.x:** AwsServiceException.awsErrorDetails().errorMessage

1.11.x: AmazonServiceException.getStatusCode **2.x:** AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode
1.11.x: AmazonServiceException.getHttpHeaders **2.x:** AwsServiceException.awsErrorDetails().sdkHttpResponse().headers
1.11.x: AmazonServiceException.rawResponse **2.x:** AwsServiceException.awsErrorDetails().rawResponse

Document History for AWS SDK for Java 2.x Migration Guide

The following table describes major updates for this migration guide.

- **Latest documentation update:** November 19, 2018

update-history-change	update-history-description	update-history-date