



Developer Guide

AWS SDK for SAP ABAP



AWS SDK for SAP ABAP: Developer Guide

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

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

Table of Contents

What is AWS SDK for SAP ABAP?	1
Features of AWS SDK for SAP ABAP	1
Maintenance	1
API reference	2
Pricing	2
Resources	2
Getting started	3
Step 1: Prepare your AWS account	3
IAM role for SAP users	3
Authentication	4
Step 2: Install the SDK	6
Step 3: Configure the SDK	6
Step 4: Functional setup	8
Step 5: Authorize SAP Users	10
Step 6: Write the code	12
Step 7: Run the application	14
Setting up	17
SAP prerequisites	17
SDK for SAP ABAP	17
SDK for SAP ABAP - BTP edition	21
Installing AWS SDK for SAP ABAP	22
Download the SDK	22
Verify the file	22
AWS SDK Transports	23
Installing SDK - BTP edition	26
Install SDK for SAP ABAP - BTP edition	27
Modules	27
Patching SDK for SAP ABAP - BTP edition	28
Configuring	29
Global settings	30
Technical settings	31
Configure scenarios	31
Application configuration	32
SDK profile	32

Logical resource resolver	33
Example	34
Runtime settings	34
Log and trace	35
OPT-IN: enhanced telemetry	35
Active scenario	35
Advanced connectivity scenarios	35
Connection through a proxy server	36
Connection through a packet inspecting firewall	36
Gateway endpoints	36
Custom interface endpoints	37
Accessing endpoints in multiple Regions	38
Service provider settings	39
Refresh, trace, and telemetry	39
SAP system refresh	39
Trace	40
Telemetry	41
Using the SDK	43
Data representation	43
Data types	43
AWS data types	45
Example program	46
Prerequisites	46
Code	47
Code sections	48
Concepts	50
API classes	51
Additional objects	51
Structure classes	51
Arrays	53
Maps	54
Higher level functions	55
Features	1
Programmatic configuration	55
Waiters	56
Paginators	57

Retry behavior	58
Building products	59
Setting a product ID	59
Limitations	60
Code examples	61
Actions and scenarios	61
Amazon Bedrock Runtime	62
CloudWatch	67
DynamoDB	74
Amazon EC2	89
Kinesis	105
Lambda	115
Amazon S3	129
SageMaker	138
Amazon SNS	157
Amazon SQS	165
Amazon Textract	173
Amazon Translate	183
Security	193
System authentication	193
Metadata authentication	194
Secret access key authentication	194
Certificate-based authentication using IAM Roles Anywhere	195
Next step	195
Best practices for IAM Security	196
Best practice for Amazon EC2 instance profile	196
IAM roles for SAP users	197
SAP authorizations	200
Authorizations for configuration	200
SAP authorizations for end users	201
Secure operations	202
Encryption Of Data At Rest	202
Encryption Of Data In Transit	202
API Usage	2
Using certificates	203
Prerequisites	203

Procedure	204
Credential Store	21
Configuration steps	207
Using SAP Credential Store with the SDK	208
Troubleshoot	212
Import failure	212
Unspecified location constraint	212
SSL error	213
Profile configuration	214
IAM authorization	215
Authorization for actions	215
Active scenario	35
Special characters	216
Connectivity	216
Additional topics	217
Releases	217
Release strategy	217
Best practices	196
Patching SDK for SAP ABAP	218
Installing an additional module	218
Uninstalling SDK for SAP ABAP	218
SAP licensing	219
Document history	220

What is AWS SDK for SAP ABAP?

AWS SDK for SAP ABAP provides an interface to the services offered by AWS in the ABAP language. Using the SDK, you can implement ABAP BADIs, reports, transactions, OData services, and other ABAP artifacts on AWS services, such as Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Translate, and more. You can also develop for ABAP-based systems, starting from SAP NetWeaver 7.4 and in an SAP Business Technology Platform environment. For more information, see [Installing AWS SDK for SAP ABAP - BTP edition](#).

Topics

- [Features of AWS SDK for SAP ABAP](#)
- [Maintenance and support for SDK major versions](#)
- [API reference](#)
- [Pricing](#)
- [Additional resources](#)

Features of AWS SDK for SAP ABAP

AWS SDK for SAP ABAP has been designed to feel familiar and natural to SAP developers. For example, while all AWS services use the `true` and `false` strings to represent Boolean data in XML and JSON structures, SDK for SAP ABAP converts these to ABAP-native `'X'` and `' '` single-character values. SDK for SAP ABAP uses native ABAP constructs as much as possible, including in data types and timestamp formats. As a result, the ABAP programmer does not need to be concerned about the underlying JSON and XML serialization or with the wire format of the API protocol.

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 and version support matrix](#)

API reference

To see a complete list of AWS SDK for SAP ABAP APIs, see [AWS SDK for SAP ABAP - API Reference Guide](#).

To see a complete module list of AWS SDK for SAP ABAP TLAs, see [AWS SDK for SAP ABAP - Module List](#).

To see a complete module list of SDK for SAP ABAP - BTP edition developer preview TLAs, see [AWS SDK for SAP ABAP - BTP edition - Module List](#).

Pricing

AWS SDK for SAP ABAP is available to you at no additional cost. You only pay for AWS resources and services that you consume with the SDK.

Additional resources

In addition to this guide, the following online resources are available for SDK for SAP ABAP.

- [SAP on AWS documentation](#)
- [AWS developer blog](#)
- [AWS developer forums](#)
- [AWS SDK Code Example Library](#)
- [@awsdevelopers](#)(Twitter)

Getting started with AWS SDK for SAP ABAP

This section describes how to get started with the SDK. It includes information about installing the SDK, performing basic configuration, and creating a Hello World code example that translates a phrase from one language to another. If you are new to AWS SDK, we recommend performing these steps in a sandbox environment.

Steps

- [Step 1: Prepare your AWS account](#)
- [Step 2: Install the SDK](#)
- [Step 3: Configure the SDK](#)
- [Step 4: Functional setup](#)
- [Step 5: Authorize SAP Users](#)
- [Step 6: Write the code](#)
- [Step 7: Run the application](#)

Step 1: Prepare your AWS account

To get started with SDK for SAP ABAP, you must have an active AWS account . You need an AWS account even if your SAP system is hosted on-premises, on SAP Business Technology Platform (BTP) or with another cloud provider.

If your SAP system is running on AWS Cloud, then you will be making calls to AWS services in your AWS account.

Topics

- [IAM role for SAP users](#)
- [Authentication](#)

IAM role for SAP users

- Create an IAM role with the instructions provided in the *AWS Identity and Access Management User Guide*. For more information, see [Creating a role to delegate permissions to an AWS service](#). Note the Amazon Resource Name (ARN) of the IAM role for later use.

- Select Amazon EC2 as the use case.
- Use SapDemoTranslate as the name of the role.
- Attach TranslateReadOnly profile to the role.
- The role must have the following entities to enable the SAP system to assume the role. Replace **"111122223333"** with your AWS account number.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": { "AWS": "111122223333" }
    }
  ]
}
```

This example shows that any principal from the AWS account **"111122223333"** can assume the role. This is a broad permission that is suitable for proof-of-concept. You can use a narrower principal for production, such as the following examples.

- A specific user – when the SAP system is using either one of the following:
 - SSF-encrypted credentials from an on-premises SAP system
 - Credentials from SAP Credential Store service on SAP BTP, ABAP environment
- A specific role – when the SAP system is on Amazon EC2 and there is an instance profile.
- Amazon EC2 – when the SAP system is on Amazon EC2 and there is *no* instance profile.

For more information, see [Best practices for IAM Security](#).

Authentication

Authentication depends on where your SAP system is hosted.

Locations

- [On AWS Cloud](#)
- [On-premises, SAP BTP or other cloud](#)

On AWS Cloud

Ensure that the EC2 instance on which your SAP system is running has an instance profile with the following permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/SapDemoTranslate"
    }
  ]
}
```

Add the ARN that you noted in the previous step.

This permission enables your SAP system to assume the `SapDemoTranslate` role on behalf of the ABAP user.

On-premises, SAP BTP or other cloud

If your SAP system is located on-premises, on SAP BTP or on other cloud, use the following steps to establish a connection for authentication using secret access key.

1. Create an IAM user. For more information, see [Creating IAM users \(console\)](#).
2. Use `SapDemoSID` as the name of the IAM user. SID is the system ID of your SAP system.
3. Assign `SapDemoTranslate` role to this user.

Retain the `access_key` and `secret_access_key`. You must configure these credentials in your SAP system.

Note

If your SAP system is located on-premises, on SAP BTP or on other cloud, you can authenticate using one of the following options.

- [Secret access key authentication](#) using SSF or SAP Credential Store

- [Using certificates with IAM Roles Anywhere](#)

Step 2: Install the SDK

See the following tabs for installation instructions.

SDK for SAP ABAP

Import SDK for SAP ABAP transports in your SAP system. You can import the transports into any client. For more information, see [Installing SDK for SAP ABAP](#).

SDK for SAP ABAP - BTP edition

Install SDK for SAP ABAP - BTP edition using the **Deploy Product** application. For more information, see [Installing SDK for SAP ABAP - BTP edition](#).

Step 3: Configure the SDK

Before configuring the SDK, ensure that you have the required authorizations. For more information, see [SAP authorizations](#).

See the following tabs for configuration instructions.

SDK for SAP ABAP

Run the /AWS1/IMG transaction to open the Implementation Guide for SDK for SAP ABAP. To run this transaction, enter /n/AWS1/IMG in the command bar of your SAP system, and then choose **Enter**.

Complete the following configurations.

- Go to **Technical Prerequisites**.
 - Review the recommended [Parameters](#) and [HTTPS connectivity](#).
- Go to **Global Settings** → **Configure Scenarios**.
 - Change the settings, according to the recommendations in [Global settings](#).
- Go to **Global Settings** → **Technical Settings**.
 - Change the settings, according to the recommendations in [Global settings](#).

- Go to **Runtime Settings** → **Log And Trace**.
 - Select **New Entries**.
 - **Trace level**: No Trace.
 - **Maximum Dump Lines**: 100.
 - **OPT-IN: enh telemetry**: Keep this blank.
 - Select **Save**.
- Go to **Runtime Settings** → **Active Scenario**.
 - Under **New Scenario**, select DEFAULT.
 - Select **Commit Scenario Change**.
 - Accept the prompt.

Prerequisites for On-Premises Systems

If your SAP system is running on-premises or in another cloud, then the credentials must be stored in your SAP database. The credentials are encrypted using SAP SSF and require a configured cryptographic library, such as SAP's CommonCryptoLib.

The steps for configuring SSF for SDK for SAP ABAP are described in the /AWS1/IMG transaction.

Note

The preceding prerequisite does not apply if your SAP system is running on Amazon EC2. SAP systems running on Amazon EC2 retrieve short-lived, automatically rotating credentials from the Amazon EC2 instance metadata.

SDK for SAP ABAP - BTP edition

Open your ABAP environment in a web browser, and navigate to the Custom Business Configurations application.

Complete the following configurations.

- Go to **Configure Scenarios**.
 - Change the settings, according to the recommendations in [Global settings](#).

- Go to **Technical Settings**.
 - Change the settings, according to the recommendations in [Global settings](#).

Step 4: Functional setup

See the following tabs for setup instructions.

SDK for SAP ABAP

Run transaction `/AWS1/IMG` (enter `/n/AWS1/IMG` in the command bar, and choose **Enter**) to open the implementation guide for AWS SDK.

- Go to **Application Configuration** → **SDK Profile**.
 - Select **New Entries**.
 - **Profile**: DEMO.
 - **Description**: Demo profile.
 - Select **Save**.
 - Highlight the entry that you created and click on the **Authentication And Settings** tree branch.
 - Select **New Entries**.
 - **SID**: The system ID of the SAP system that you are currently in.
 - **Client**: The client of the SAP system that you are currently in.
 - **Scenario ID**: The dropdown list where you'll find the DEFAULT scenario created by your Basis administrator.
 - **AWS Region**: enter the AWS Region that you want to make calls to. If your SAP system is running in AWS, enter the AWS Region that it is running in.
 - Authentication Method:
 - Select **Instance Role via Metadata** if your SAP system is running on Amazon EC2.
 - Select **Credentials from SSF Storage** if your SAP system is running on-premises or in another cloud.
 - Select **Set Credentials**.
 - Enter the Access Key ID and Secret Access Key that you created in the previous step.
 - Keep **Disable IAM roles** blank.
 - Select **Save**.

- Click on the **IAM Role Mapping** tree branch.
- Select **New Entries**.
 - Enter **Sequence number**: 010.
 - Enter **Logical IAM role**: TESTUSER.
 - Enter **IAM Role ARN**: enter the arn:aws: of the IAM role containing the TranslateReadOnly policy created in the previous step.

SDK for SAP ABAP - BTP edition

Set up authentication using SAP Credential Store. For more information, see [Using SAP Credential Store](#).

Open your ABAP environment in a web browser, and navigate to the Custom Business Configurations application.

- Go to **SDK Profile**.
 - Select **Edit** to create a new profile.
 - **Profile**: DEMO.
 - **Description**: Demo profile.
- Select the right arrow key next to the created entry to navigate to **Authentication and Settings** tab.

Select **New Entries**.

- **SID**: The system ID of the SAP system that you are currently in.
- **Client**: The client of the SAP system that you are currently in.
- **Scenario ID**: The dropdown list where you'll find the DEFAULT scenario created by your Basis administrator.
- **AWS Region**: enter the AWS Region that you want to make calls to. If your SAP system is running in AWS, enter the AWS Region that it is running in.
- Authentication Method: Select **Credentials from SAP Credential Store**.
- Enter the **Namespace** and **Key name** of the credentials stored in SAP Credentials Store.
- Enter the name of the **Communication Arrangement** created to establish communication between SDK for SAP ABAP - BTP edition and SAP Credential Store.

- Right-click on the right arrow key next to the created entry to navigate to **IAM Role Mapping** tab.

Select **New Entries**.

- Enter **Sequence number**: 010.
- Enter **Logical IAM role**: TESTUSER.
- Enter **IAM Role ARN**: enter the arn:aws: of the IAM role containing the TranslateReadOnly policy created in the previous step.

Step 5: Authorize SAP Users

SAP users are not authorized to use AWS functionality by default. The users must be explicitly authorized using SAP authorizations. See the following tabs for more details.

SDK for SAP ABAP

Create a PFCG role

- Go to transaction PFCG
- Enter the role name ZAWS_SDK_DEMO_TESTUSER and select **Create Single Role**.
 - **Description**: Role for demo AWS SDK functionality.
 - Go to the **Authorizations** tab.
 - Select **Change Authorization Data** and accept the informational pop-up.
 - At the *Choose Template* pop-up, select **Do not select templates**.
 - Select **Add Manually** from the toolbar.
 - Add the following authorization objects:
 - /AWS1/LROL
 - /AWS1/SESS
 - In the authorization tree, enter:
 - Profile for accessing AWS APIs: DEMO
 - Logical IAM Role: TESTUSER
 - Select **Save**.
 - Select **Generate**.
 - Select **Back**.

- Select **Save** to save the role.

Assign the PFCG role to SAP users

Any user who has the ZAWS_SDK_DEMO_TESTUSER role assigned will be authorized to use AWS SDK functions with the settings configured in DEMO SDK profile. The authorized user will also assume the IAM role mapped to the TESTUSER logical IAM role in that profile.

- Run transaction SU01.
 - Enter the user ID of an SAP user who will be testing AWS SDK functionality.
 - Select **Change**.
 - Go to the **Roles** tab and assign ZAWS_SDK_DEMO_TESTUSER role to the user.
 - Select **Save**.

SDK for SAP ABAP - BTP edition

Create a Business role

- Open your ABAP environment in a web browser, and navigate to the **Maintain Business Roles** application.
- Select **Create from Template**, and enter the following details.
 - **Template** – Choose **/AWS1/RT_BTP_ENDUSER**.
 - **New Business Role ID** – Enter an ID.
 - **New Business Role Description** – Enter a description.
- Select **OK** to see the page for the business role.
- Under **General Role Details** tab, go to **Access Categories**, and set the **Write, Read, Value Help** field as **Restricted**.
- Select **Maintain Restrictions**, and expand **Assigned Restriction Types** from the left navigation pane. Update the following field in the **Restrictions and Values** section.
 - Under **Choose SDK Session**, select the pencil icon next to **SDK Profile**, and navigate to the **Ranges** tab. Enter **DEMO**, and select **Add**.
 - Under **Choose Logical IAM Role**, select the pencil icon next to **Logical IAM Role**, and navigate to the **Ranges** tab. Enter **TESTUSER**, and select **Add**.

Select the pencil icon next to **SDK Profile**, and navigate to the **Ranges** tab. Enter **DEMO**, and select **Add**

- Navigate back to the Business Role template, and open the **Business Users** tab. Select **Add** to assign the newly created Business Role to an SAP business user who will test the SDK functionality. Select **Save**.

Any business user assigned to the created Business Role will be authorized to use AWS SDK functions with the settings configured in DEMO SDK profile. The authorized user will also assume the IAM role mapped to the TESTUSER logical IAM role in that profile.

Step 6: Write the code

See the following tabs for more details.

SDK for SAP ABAP

1. Open transaction SE38.
 - Enter ZDEMO_TRANSLATE_HELLO_WORLD as the program name.
 - Select Create.
 - Enter AWS SDK Hello World In Any Language as the title.
 - Type: choose **Executable Program**.
 - Status: choose **Test Program**.
 - Select **Save**.
 - Save the program as a **Local Object**.

Add the following code.

```
*&-----*
*& Report  ZAWS1_DEMO_XL8_SIMPLE
*&
*&-----*
*& A simple demo of language translation with AWS Translate
*&
*&-----*
```

```

REPORT zaws1_demo_xl8_simple.

START-OF-SELECTION.
  PARAMETERS pv_text TYPE /aws1/xl8boundedlengthstring DEFAULT 'Hello, World'
  OBLIGATORY.

  PARAMETERS pv_lang1 TYPE languageiso DEFAULT 'EN' OBLIGATORY.
  PARAMETERS pv_lang2 TYPE languageiso DEFAULT 'ES' OBLIGATORY.

TRY.
  DATA(go_session) = /aws1/cl_rt_session_aws=>create( 'DEMO' ).
  DATA(go_xl8)      = /aws1/cl_xl8_factory=>create( go_session ).
  DATA(lo_output) = go_xl8->translatetext(
    iv_text          = pv_text
    iv_sourcelanguagecode = CONV /aws1/xl8languagecodestring( pv_lang1 )
    iv_targetlanguagecode = CONV /aws1/xl8languagecodestring( pv_lang2 )
  ).

  WRITE: / 'Source Phrase: ', pv_text.
  WRITE: / 'Target Phrase: ', lo_output->get_translatedtext( ).
  CATCH /aws1/cx_xl8unsuppedlanguage00 INTO DATA(lo_lang).
  WRITE: / 'ERROR' COLOR COL_NEGATIVE,
    'Cannot translate from',
    lo_lang->sourcelanguagecode,
    'to',
    lo_lang->targetlanguagecode.
  CATCH cx_root INTO DATA(lo_root).
  WRITE: / 'ERROR' COLOR COL_NEGATIVE, lo_root->get_text( ).
ENDTRY.

```

SDK for SAP ABAP - BTP edition

1. Right-click on the package where the ABAP class needs to be created, then select **New > ABAP class**.
2. Enter **ZCL_DEMO_XL8_SIMPLE** for Class name, and add a Class description. Select **Next**.
3. Create or choose a transport request. Select **Finish**.

Add the following code.

```

CLASS zcl_demo_xl8_simple DEFINITION
  PUBLIC
  FINAL

```

```
CREATE PUBLIC .

PUBLIC SECTION.
  INTERFACES if_oo_adt_classrun.
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.

CLASS zcl_demo_xl8_simple IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.

    TRY.
      " input parameters
      DATA(pv_text) = |Hello, World|.
      DATA(pv_lang1) = |EN|.
      DATA(pv_lang2) = |ES|.

      DATA(go_session) = /aws1/cl_rt_session_aws=>create( 'DEMO' ).
      DATA(go_xl8)      = /aws1/cl_xl8_factory=>create( go_session ).
      DATA(lo_output) = go_xl8->translatetext(
        iv_text          = pv_text
        iv_sourcelanguagecode = pv_lang1
        iv_targetlanguagecode = pv_lang2
      ).

      out->write( |Source Phrase: { pv_text }| ).
      out->write( |Target Phrase: { lo_output->get_translatedtext( ) }| ).
      CATCH /aws1/cx_xl8unsuppdedlanguage00 INTO DATA(lo_lang).
        out->write( |ERROR - Cannot translate from { lo_lang->sourcelanguagecode }
to { lo_lang->targetlanguagecode }| ).
      CATCH cx_root INTO DATA(lo_root).
        out->write( |ERROR - { lo_root->get_text( ) }| ).
    ENDTRY.
  ENDMETHOD.
ENDCLASS.
```

For details on how to write ABAP code that uses the SDK, see [Using AWS SDK for SAP ABAP](#).

Step 7: Run the application

See the following tabs for more details.

SDK for SAP ABAP

Run the application in SE38. If successful, the following will be your output.

```
Source Phrase: Hello, World  
Target Phrase: Hola, mundo
```

If you are missing authorizations, configuration, or Basis prerequisites, you might get an error message. See the following example.

```
ERROR Could not find configuration under profile DEMO with  
scenario DEFAULT for SBX:001
```

If your SAP role authorizes you to use an SDK profile and map it to a logical IAM role while your IAM permissions are not configured for the SAP system to assume the IAM role, the following will be your output.

```
ERROR Could not assume role arn:aws:iam::111122223333:role/SapDemoTranslate
```

In this case, review your IAM permissions and trust configuration on the IAM roles, users, or both defined in [the section called “Step 1: Prepare your AWS account”](#).

SDK for SAP ABAP - BTP edition

Run the application on **Eclipse > Run As > ABAP Application (Console)**. If successful, the following will be your output.

```
Source Phrase: Hello, World  
Target Phrase: Hola, mundo
```

If you are missing authorizations, configuration, or Basis prerequisites, you might get an error message. See the following example.

```
ERROR Could not find configuration under profile DEMO with  
scenario DEFAULT for SBX:001
```

If your SAP role authorizes you to use an SDK profile and map it to a logical IAM role while your IAM permissions are not configured for the SAP system to assume the IAM role, the following will be your output.

ERROR Could not assume role *arn:aws:iam::111122223333:role/SapDemoTranslate*

In this case, review your IAM permissions and trust configuration on the IAM roles, users, or both defined in [the section called “Step 1: Prepare your AWS account”](#).

Setting up

This section provides information about how to set up your development environment to use AWS SDK for SAP ABAP.

Topics

- [SAP prerequisites](#)
- [Installing AWS SDK for SAP ABAP](#)
- [Installing AWS SDK for SAP ABAP - BTP edition](#)

SAP prerequisites

The following prerequisites for installing the SDK are applicable when your SAP systems are hosted on AWS.

Topics

- [Prerequisites for AWS SDK for SAP ABAP](#)
- [Prerequisites for AWS SDK for SAP ABAP - BTP edition](#)

Prerequisites for AWS SDK for SAP ABAP

The following are the prerequisites for AWS SDK for SAP ABAP.

Topics


- [Basis release](#)
- [Kernel release](#)
- [Parameters](#)
- [Notes](#)
- [Outbound connectivity](#)
- [HTTPS connectivity](#)
- [Access to Amazon EC2 instance metadata](#)

Basis release

SDK for SAP ABAP is compatible with SAP NetWeaver 7.4 and higher. SDK for SAP ABAP doesn't touch any SAP application tables. It is completely agnostic about the applications, such as SAP Enterprise Resource Planning and SAP Landscape Transformation Replication Server.

The minimum supported SP-Level for SAP_BASIS 740 is SP 0008. For more information, see [SAP Note 1856171 - Supporting form fields of the same name in CL_HTTP_ENTITY](#) (requires SAP portal access). Based on your business requirements, you can choose a higher SP-Level, as shown in the following image.

Installed Software Component Versions Installed Product Versions



Component	Release	SP-Level	Support Package	Short Description of Component
SAP_BASIS	740	0026	SAPKB74026	SAP Basis Component
SAP_ABA	740	0026	SAPKA74026	Cross-Application Component
SAP_GWFND	740	0027	SAPK-74027INSAPGWFND	SAP Gateway Foundation
SAP_UI	754	0008	SAPK-75408INSAPUI	User Interface Technology
PL_BASIS	740	0008	SAPK-74008INSAPBASIS	Basis Plus for PLM

There is no minimum SP-Level requirement for SAP_BASIS 750 and higher releases.

Kernel release

SDK for SAP ABAP and tools that use the Internet Communication Manager (ICM) for HTTP connectivity, rely on the SAP kernel for its cryptographic, HTTP, XML, and JSON capabilities. We recommend using the latest kernel release that is compatible with your SAP NetWeaver platform. The minimum requirement is kernel release **741**. For more information, see [SAP Note 2083594 - SAP Kernel Versions and SAP Kernel Patch Levels](#) (requires SAP portal access).

If you are using kernel release 741 or 742, the following patch levels are required:

- 741 patchno 212
- 742 patchno 111

Parameters

Your SAP system must support Server Name Indication (SNI) as described in the following SAP Notes (requires SAP portal access).

- [SAP Note 2124480 - ICM/Web Dispatcher: TLS Extension Server Name Indication \(SNI\) as client](#)
- [SAP Note 2582368 - SapSSL update for client-side sending of TLS extension SNI by saphttp, sapkprotp, sldreg](#)

Configure the following parameter in the DEFAULT.PFL file.

```
icm/HTTPS/client_sni_enabled = TRUE
```

Notes

Apply the following SAP Note to your system.

- <https://launchpad.support.sap.com/#/notes/0001856171>
- <https://launchpad.support.sap.com/#/notes/0002619546>

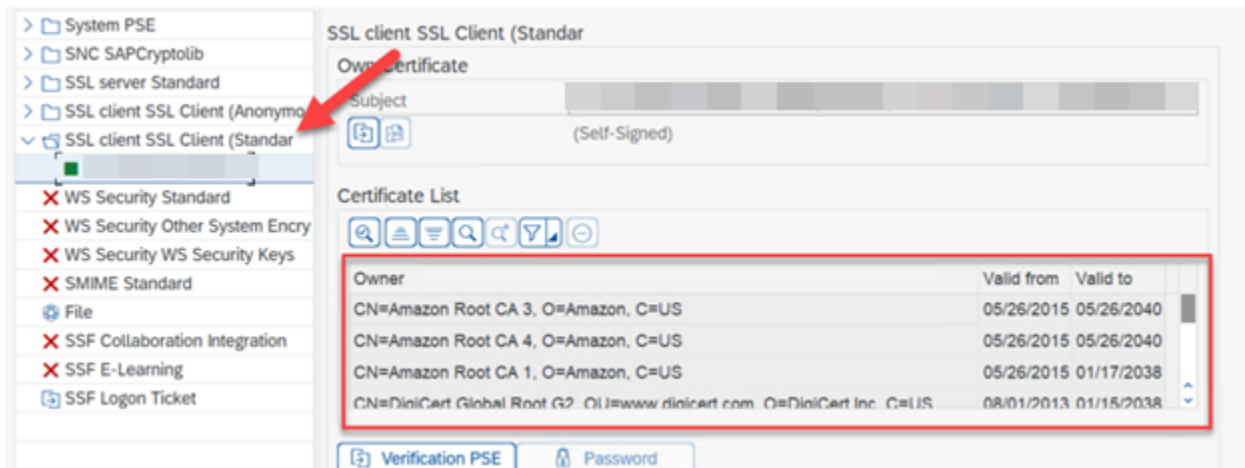
Outbound connectivity

SDK for SAP ABAP is an HTTPS client. The SAP system sends HTTPS messages outbound. There is no requirement of an inbound connectivity.

HTTPS connectivity

All AWS API calls are made with encrypted HTTPS channels. The SAP system must be set up to trust AWS certificates to establish an outbound HTTPS connection.

1. Go to <https://www.amazontrust.com/repository/>.
2. Under **Root CAs**, download all the certificates using the *PEM* link.
3. Import these certificates in STRUST of your SSL Client (Standard) PSE on each of your SAP systems, as shown in the following image.



Access to Amazon EC2 instance metadata

ABAP system makes unencrypted HTTP connections to localhost (<http://169.254.169.254>) to enable Amazon EC2 instance metadata. The HTTP channel is used only to retrieve AWS credentials from the local server. The HTTP traffic stays within the host.

The metadata allows an SAP system in AWS to securely authenticate itself without storing a secret key in the SAP Secure Store. This feature is applicable only to SAP systems hosted on Amazon EC2.

Configure the DEFAULT.PFL file with the following parameter to enable your SAP system to make an unencrypted outbound HTTP connection.

```
icm/server_port_<xx> = PROT=HTTP,PORT=8000,TIMEOUT=60,PROCTIMEOUT=600
```

Use the following parameter to enable the outbound HTTP connection without enabling the inbound connection.

```
icm/server_port_<xx> = PROT=HTTP,PORT=0,TIMEOUT=60,PROCTIMEOUT=600
```

Verify that your SAP system is configured for outbound HTTP connections with the following steps:

1. Run **SMICM** transaction.
2. Go to **Active Services**.
3. Verify that you see a **green check mark** in the HTTP row, under *Active* column, as shown in the following image.

Active Services						
No.	Protocol	Service Name/Port	Host Name	Keep Alive	Proc.Timeo	Actv E:
1	HTTPS	50001		60	600	✓
2	HTTP	0		60	600	✓

Prerequisites for AWS SDK for SAP ABAP - BTP edition

The following are the only prerequisites for AWS SDK for SAP ABAP - BTP edition.

Topics

- [SAP Landscape Portal – BTP edition](#)
- [SAP Credential Store – BTP edition](#)

SAP Landscape Portal – BTP edition

This prerequisite is only applicable for AWS SDK for SAP ABAP - BTP edition.

SAP Landscape Portal is the only supported mechanism to install add-ons in an SAP BTP environment. Ensure that you are subscribed to use this service. For more information, see [Landscape Portal](#).

SAP Credential Store – BTP edition

This prerequisite is only applicable for AWS SDK for SAP ABAP - BTP edition.

In the developer preview, secret access key authentication is the only supported mechanism for authenticating AWS SDK for SAP ABAP - BTP edition. The SDK reads the credentials from the Credential Store, and stores the secret access key securely.

You must meet the following prerequisites.

- Subscription to Credential Store.
- Credential Store assigned as an entitlement to your BTP sub-account. See [Initial setup](#) for more details.
- A service instance with standard plan for Credential Store. See [Create a service instance](#) for more details.

For more information, see [Using SAP Credential Store](#).

The SAP Credential Store service runs in SAP BTP outside of the ABAP BTP system. See [SAP Credential Store](#) for more details.

Installing AWS SDK for SAP ABAP

Topics

- [Download SDK for SAP ABAP](#)
- [Verify SDK for SAP ABAP file – optional](#)
- [AWS SDK Transports](#)

Download SDK for SAP ABAP

Download the SDK from <https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/abapsdk-LATEST.zip>.

```
curl "https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/abapsdk-LATEST.zip" -o "abapsdk-LATEST.zip"
```

When the download is complete, we recommend that you unzip the downloaded file into a directory, such as `/tmp/awssdk`.

Verify SDK for SAP ABAP file – *optional*

This optional step of validating the signature of your SDK file helps you confirm that your SDK has not been tampered with. Use the following steps to verify your SDK file.

1. Download the SDK SIGNATURE file with the following command.

```
curl "https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/abapsdk-LATEST.sig" -o "abapsdk-LATEST.sig"
```

2. Copy the following public key, and save it to a file named `abapsdk-signing-key.pem`.

```
-----BEGIN PUBLIC KEY-----  
MIICIjANBgkqhkiG9w0BAQEFAAACg8AMIICCgKCAgEAmS3oN3wKBh4HJ0Ga0tye  
15RR5909nuw0Jx0vEDCT709wUrxS3mjgEw6b6hvr2dLdoFr+eH4ewT5bV16U3gDv  
051sTdEJJpfLEWJJZZNK3v9fGWKyXgYe+ifmsPmf4lhNd2auzpvIy2Ur1SYijCRB
```

```
BWZFW+Ux00kILz+8vCFSXMZ6Z0qtLI1ZFbGrn6A5adbwwzf0qkg9BUEZK0wB6TAi
ZTnkMdBZGCBM9K2MRKKMxtrixUn+TFcAYyh5pM9tUAb2q4XE5m7092UnZG7ur/QY1
1FSZwAhQmk8hUPgUaq00QRC6z3TRzIGK0A/DI0cUPJMzFR4LCxEJkgh4rkRaU9V2
07DthUpj8b7QcQai0pnMpBf3zWLgbjNmX0hB0Eprg8/nVRHspf3zuiscJ21MPkz0
cHOR31MNsMLzm+d/gVklT31R/JwAcFCkXTwvR8/V0WNGZZXdVUbefrI/k7fP60B
bzUrI1N4poq16rc4Tk5Derg+wQ7r0WjXkXop2kiCMjbYo0o10kS/At64PLj pz8dH
Zg25o79U9EJ1n+1pqZ297Ks+Hoct0v2GPbeeh0s7+N0fRTy0r81EZIURLPKLVQUw
otVRzNDgLOA7eA667NimegZfHCmqEwK9tXakZUHAcMzRPyhALc/HtmovxdStN9h1
JC4ex0GqstAv1fX5QaTbMSECAwEAAQ==
-----END PUBLIC KEY-----
```

3. Verify the downloaded SDK ZIP file with the following command. The command requires `openssl` that is part of many Linux distributions.

```
openssl dgst -sha256 -verify abapsdk-signing-key.pem -keyform PEM -signature
abapsdk-LATEST.sig abapsdk-LATEST.zip
```

4. Verify that the output of the preceding command is `Verified OK`.
5. If the output is `Verification Failure`, repeat the preceding steps. If you continue to receive a failed output, don't install the SDK and contact AWS Support.

AWS SDK Transports

Topics

- [Contents](#)
- [Importing](#)
- [Namespace](#)

Contents

The installation of SDK for SAP ABAP is completed through ABAP Transports. You must import these transports into your development or sandbox environment.

Each SDK for SAP ABAP release completely replaces the previous one. There is no need to apply incremental transports. The transports are bundled in a ZIP file. The following is the structure of the ZIP file.

```
transports/
transports/core/
```

```
transports/core/Knnnnnn.AWS
transports/core/Rnnnnnn.AWS
transports/tla1/
transports/tla1/Knnnnnn.AWS
transports/tla1/Rnnnnnn.AWS
transports/tla2/
transports/tla2/Knnnnnn.AWS
transports/tla2/Rnnnnnn.AWS
.
.
.
```

The `transports` folder contains a `core` subfolder. The `core` subfolder contains the core runtime transports and a subfolder for each module, named by the three letter abbreviation of the module. For a complete module list of the TLAs, see [AWS SDK for SAP ABAP - Module List](#).

AWS SDK transports are workbench requests. Depending on the configuration of your TMS routes, the SDK may not automatically forward to your quality assurance and production queues after importing into the previous system. You must manually add them to the queue of each system.

When your project is ready for the next phase, AWS SDK can be imported along with separate transports containing your own Z code with business functionality. If you are using a change control system, such as SAP Change Request Management (ChaRM), consult your ChaRM administrator for correct handling of third-party transports.

Importing

Topics

- [Key pointers](#)
- [Time to import](#)

AWS SDK transports are client-independent. The core transport is mandatory and contains the SDK runtime code, the API for AWS Security Token Service, and the API for Amazon Simple Storage Service. The remaining SDK modules are each delivered in a separate transport. To keep the size of the SDK small in your system, each SDK module is optional. You can install additional modules later, if required for your business logic.

For example, if you want to use the APIs for Amazon S3 and Amazon Translate, import the core transport (containing core runtime, Amazon S3, and AWS STS modules) and the `x18` transport (containing the module for Amazon Translate) transports.

To see a complete list of SDK for SAP ABAP APIs, see [SDK for SAP ABAP - API Reference Guide](#).

The following are key pointers when importing AWS SDK transports.

- Each transport is delivered as Knnnnnn.AWS and Rnnnnnn.AWS
 - Knnnnnn.AWS must be copied to /usr/sap/trans/cofiles
 - Rnnnnnn.AWS must be copied to /usr/sap/trans/data.
- When importing transports, you must select the **Ignore Invalid Component Version** found at *Import Transport Request > Options > Import Options*.
- All desired transports can be imported simultaneously.
- If importing the transports separately, the core transport must be imported first.
- The release level of all the transports must be identical.

Time to import

AWS SDK transports may take many minutes to import. The transports are successful if STMS shows a green (RC=0) or yellow (RC=4) light.

- A red light (RC=8) indicates that the import had a syntax error.
 - Select **Request** → **Display** → **Logs to examine the import error**.
 - During the import, if an error is thrown because of a missing interface IF_SYSTEM_UUID_RFC4122_STATIC, then ensure that SAP Note 2619546 is applied to the system. For more information, see [Notes](#).
 - If the cause of the error is unknown, contact AWS Support.
- A red lightning bolt (RC=12) indicates that the transport files have not been correctly loaded into /usr/sap/trans or do not have the necessary permissions.

Key pointers

The following are key pointers when importing AWS SDK transports.

- Each transport is delivered as Knnnnnn.AWS and Rnnnnnn.AWS
 - Knnnnnn.AWS must be copied to /usr/sap/trans/cofiles
 - Rnnnnnn.AWS must be copied to /usr/sap/trans/data.

- When importing transports, you must select the **Ignore Invalid Component Version** found at *Import Transport Request > Options > Import Options*.
- All desired transports can be imported simultaneously.
- If importing the transports separately, the core transport must be imported first.
- The release level of all the transports must be identical.

Time to import

AWS SDK transports may take many minutes to import. The transports are successful if STMS shows a green (RC=0) or yellow (RC=4) light.

- A red light (RC=8) indicates the import had a syntax error.
 - Select **Request** → **Display** → **Logs to examine the import error**.
 - During the import, if an error is thrown due to a missing interface `IF_SYSTEM_UUID_RFC4122_STATIC`, then ensure that SAP Note 2619546 is applied to the system. For more information, see [Notes](#).
 - If the cause of the error is unknown, contact AWS Support.
- A red lightning bolt (RC=12) indicates the transport files have not been correctly loaded into `/usr/sap/trans` or do not have the necessary permissions.

Namespace

SDK for SAP ABAP uses the `/AWS1/` namespace and does not modify SAP objects or any other objects in your system with the following exception.

- AWS auth objects are in an **Auth Object Class**. Auth Object Classes are limited to four characters and do not support namespaces. SDK for SAP ABAP uses Auth Object Class is YAW1. If you already have an auth object class YAW1 in transaction SU21, contact AWS Support before installation.

Installing AWS SDK for SAP ABAP - BTP edition

The BTP edition is in developer preview, and can be installed by joining the preview. To install the SDK, fill the participation form at [AWS SDK for SAP ABAP - BTP edition developer preview](#).

Before installing SDK for SAP ABAP - BTP edition, ensure that you are meeting the required prerequisites. For more information, see [SAP Landscape Portal](#) and [SAP Credential Store](#).

Topics

- [Install SDK for SAP ABAP - BTP edition](#)
- [Modules](#)
- [Patching SDK for SAP ABAP - BTP edition](#)

Install SDK for SAP ABAP - BTP edition

1. Go to your SAP Landscape Portal instance, and launch the **Deploy Product** fiori application.
2. In **Products**, select **/AWS1/SDK_OMNI** under **Partner Products**.

Contact AWS Support if you do not see /AWS1/SDK_OMNI after being accepted in the developer preview.

3. In **Target Version**, choose the version of SDK for SAP ABAP - BTP edition you want to install on your system.
4. In **Available Systems**, check the checkboxes for all the SIDs on which you want to install the SDK.
5. Select **Deploy**, enter the scheduling details, and select **Schedule**. You can monitor the progress in **Product Version Deployment Status**.

The installation may take 30-45 minutes, and includes system downtime. For more details, see [Deploy Product](#).

Modules

The following modules are included in the developer preview of AWS SDK for SAP ABAP - BTP edition.

- [Amazon API Gateway \[agw\]](#)
- [Amazon Athena \[ath\]](#)
- [Amazon Bedrock Runtime \[bdr\]](#)
- [Amazon Comprehend \[cpd\]](#)
- [Amazon EventBridge \[evb\]](#)

- [Amazon Forecast \[fcs\]](#)
- [Amazon Kinesis \[kns\]](#)
- [Amazon Data Firehose \[frh\]](#)
- [Amazon SageMaker \[sgm\]](#)
- [Amazon Simple Notification Service \[sns\]](#)
- [Amazon Simple Queue Service \[sqs\]](#)
- [Amazon Simple Storage Service \[s3\]](#)
- [AWS Systems Manager \[ssm\]](#)
- [Amazon Textract \[tex\]](#)
- [Amazon Transcribe \[tnb\]](#)
- [Amazon Translate \[x18\]](#)
- [AWS CloudTrail \[trl\]](#)
- [AWS IoT \[iot\]](#)
- [AWS KMS \[kms\]](#)
- [AWS Lambda \[lmd\]](#)
- [AWS Secrets Manager \[smr\]](#)
- [AWS Security Token Service \[sts\]](#)
- [AWS Transfer Family \[trn\]](#)
- [IAM Roles Anywhere \[r1a\]](#)
- [Amazon Redshift Data API \[rsd\]](#)

Patching SDK for SAP ABAP - BTP edition

The patching process for SDK for SAP ABAP - BTP edition is similar to the installation process. If you install the SDK on a system that has an already installed older version, then the SDK is patched to your choice of new version.

Configuring AWS SDK for SAP ABAP

Before using AWS SDK for SAP ABAP, you must configure the SDK with technical and functional settings that are necessary for the SDK operations. Some settings are transportable and some are runtime settings. Many of the settings are directly analogous to the settings defined in .INI files for other SDKs.

The SDK configurations, except for Runtime settings, must be completed in your development environment. You can transport configurations to QA and production following the usual transport and change control rules. Transportable configuration is not recommended for production environments.

If you do not have permissions to configure AWS SDK, see [SAP authorizations](#).

Configuring AWS SDK for SAP ABAP

To run the configuration transaction, enter /n/AWS1/IMG in the SAPGUI command bar.

Configuring AWS SDK for SAP ABAP - BTP edition

Use the following steps to configure SDK for SAP ABAP - BTP edition.

1. Open your ABAP environment in a web browser.
2. Navigate to Custom Business Configurations application.

To create a customizing request using the Export Customizing Transports application, see [Working in the Export Customizing Transports App - Create Request](#).

In the Custom Business Configuration application, you can group configurations based on the type of SDK settings. Use the following steps to group configurations.

1. Open your ABAP environment in a web browser, and navigate to the Custom Business Configurations application.
2. Select **Settings** > **Group**, and choose **Configuration Group** from the drop-down list. Select **OK**.
3. The configurations are now available in a hierarchical structure as displayed in the image. To save the view, see [Views \(Variant Management\) - Components](#).

Custom Business Configurations (4)

Name	Description	
Application Configuration		
SDK Profile	Maintain AWS SDK Profile	>
Logical Resource Resolver	Maintain Logical Resource Resolution	>
Global Settings		
Technical Settings	Maintain Technical Settings	>
Configure Scenarios	Configure Scenarios	>

This section covers the following topics.

Topics

- [Global settings](#)
- [Application configuration](#)
- [Runtime settings](#)
- [Advanced connectivity scenarios](#)
- [Service provider settings](#)
- [Refresh, trace, and telemetry topics for AWS SDK for SAP ABAP](#)

Global settings

Use /n/AWS1/IMG IMG transaction for AWS SDK for SAP ABAP, and Custom Business Configuration application for AWS SDK for SAP ABAP - BTP edition to configure the global settings. This topic uses IMG and Custom Business Configuration interchangeably.

This section covers the following topics.

Topics

- [Technical settings](#)

- [Configure scenarios](#)

Technical settings

The Global settings of /AWS1/IMG transaction affect the behavior of the entire SDK. These settings are generally configured by a Basis administrator. You can set these values to the following recommended settings.

- Select **New Entries**.
 - **S3 regionalization**: Access us-east-1 buckets by using s3.amazonaws.com.
 - **STS regionalization**: Access STS by using global endpoint.
 - **Disable EC2 metadata**: Keep this field blank. This field is read-only in the BTP edition, and is set to 'Yes' by default.
 - **Metadata Endpt Mode**: Use IPv4 metadata endpoint. This field is read-only in the BTP edition, and is auto-updated.
 - **Metadata Endpt URL**: Keep this field blank. This field is read-only in the BTP edition.
- Select **Save**.

Configure scenarios

Scenarios enable AWS SDK to more efficiently switch settings during a multi-Region disaster testing or disaster recovery testing scenario. You may not need this feature, and instead need only to configure the following DEFAULT scenario.

- Select **New Entries**.
 - Scenario ID:DEFAULT
 - Scenario Description: Default Scenario
- Select **Save**.

If you have a multi-Region disaster recovery setup or other unique cases that require a quick change of settings, then you can configure multiple scenarios.

- DEFAULT - Standard operation.
- DR - Special configuration if a disaster requires moving the entire system to another Region.

- DR_TEST - Special configuration for simulating a disaster, for example, in a temporary clone of production.

Application configuration

Configuring SDK for SAP ABAP is similar to configuring other ABAP-based applications. It is organized into different *profiles* to group the settings of various scenarios. An ABAP SDK profile defines the settings required for a specific application scenario. For example, if transactions ZVA01, ZVA02, and ZVA03 are invoice-related transactions enhanced and runs on AWS services, such as Amazon S3, AWS Lambda, and Amazon SageMaker, then an SDK profile called ZINVOICE can be made. This profile can group the technical settings, SAP authorizations, and IAM role mappings for the invoice-related functionality.

Use /n/AWS1/IMG transaction for AWS SDK for SAP ABAP, and Custom Business Configuration application for AWS SDK for SAP ABAP - BTP edition to configure the global settings. This topic uses IMG and Custom Business Configuration interchangeably.

Topics

- [SDK profile](#)
- [Logical resource resolver](#)
- [Example](#)

SDK profile

An ABAP SDK profile defines the following for each SID and client.

Note

The client is always 100 in SAP BTP, ABAP environment.

- The default AWS Region for all API calls. For example, if your SAP systems are running in the us-east-1 Region, it is likely that your other AWS resources are also in the same Region, and this should be your default Region. Your ABAP code can override the default Region.
- Authentication method

- For SAP systems running on Amazon EC2, we strongly recommend choosing instance role metadata to benefit from the short-lived, automatically rotating credentials.
- For SAP systems running on-premises or in another cloud, you must choose credentials from SSF storage.
- For ABAP systems running on SAP BTP, you must choose credentials from SAP Credential Store. For more information, see [Using SAP Credential Store for authentication](#).
- A mapping of logical IAM roles to IAM roles.
 - This mapping is sorted in the order of descending priority.
 - An IAM role of highest priority for which a user is authorized in a PFCG role will automatically be selected for the user.

Note

PFCG roles are called Business Roles in SAP BTP, ABAP environment.

When an ABAP program wants to connect to an AWS service, it will specify an ABAP SDK profile that pulls the necessary settings. An AUTHORIZATION-CHECK will be performed to confirm that the user has permissions to access the SDK profile. Your SAP Security Administrator can define a PFCG role granting you access to the appropriate users.

Logical resource resolver

Logical resource resolver enables you with a standard place to store resource names. It ships with SDK for SAP ABAP. Its action is similar to the way that FILE transaction maps logical file names to physical file names.

A logical resource defines the concept of an AWS resource, such as the Amazon S3 bucket that holds our invoices. This logical resource, for example, can be named ZINVOICES_OUTBOUND and it can map to a different physical bucket name, depending on whether the SAP system is development, QA, or production.

SDK for SAP ABAP is set up such that a QA system resolves logical resources to the QA physical resources, even after a system refresh from production. The resource mappings for ALL systems is defined in your development SAP system and transported forward. This approach is different from the usual setup in SAP systems where the mapping is handled as master data and set in each

system. The advantage of logical resource resolver offered by SDK for SAP ABAP is that the chances of a mistaken transport after system refreshes are almost none.

Example

There are four separate Amazon S3 buckets - one each for development, production, and QA, as well as a second QA bucket for regression testing.

When the SDK resolves a logical resource like ZINVOICE_OUTBOUND to a physical resource, it checks SY-SYSID and SY-MANDT to ask *Which SID and client am I running in?*, and automatically selects the correct physical resource.

If the mapping of a resource in production needs to change, you must change the mapping in the IMG of the development system and transport it forward. This ensures that reassigning AWS resources to an SAP system is subject to change control as with any other transport.

Note

As the SDK configuration is client-dependent, reassignment of resources is transported in a customizing request, and the transport must be imported into each client.

Runtime settings

This section covers the following topics.

Note

These settings are not transportable and are local to each SAP system.

Topics

- [Log and trace](#)
- [OPT-IN: enhanced telemetry](#)
- [Active scenario](#)

Log and trace

You can activate a trace for debugging purposes. It is recommended to keep the trace level at **No Trace**, unless diagnosing a technical issue. For more information, see [secure operation](#).

These settings are not applicable to SDK for SAP ABAP - BTP edition.

OPT-IN: enhanced telemetry

All SDKs send telemetry information to AWS for support purposes. You can opt in for enhanced telemetry. This is particularly useful when you contact AWS Support to identify the source of a particular API call. For more information, see [Trace](#) and [Telemetry](#).

These settings are not applicable to SDK for SAP ABAP - BTP edition.

Active scenario

Activate your DEFAULT scenario in this transaction. This activation is required only once for each system and should not be changed unless the system is undergoing a multi-Region disaster recovery. In a multi-Region setup, you can use this setting to switch your SAP system to a disaster recovery environment or disaster recovery test scenarios.

Advanced connectivity scenarios

AWS SDK for SAP ABAP consumes AWS services by making HTTPS calls to AWS endpoints. In general, AWS endpoints are accessible over the internet. An SAP system must be able to reach out to the internet to establish these outbound connections. SDK for SAP ABAP never requires an inbound connection from the internet into the SAP system.

The following scenarios offer different ways to establish the outbound connection.

Scenarios

- [Connection through a proxy server](#)
- [Connection through a packet inspecting firewall](#)
- [Gateway endpoints](#)
- [Custom interface endpoints](#)
- [Accessing endpoints in multiple Regions](#)

Connection through a proxy server

To establish a connection through a proxy server, use the following steps.

1. In the SDK, go to Transaction **SICF**.
2. Choose **Execute**.
3. In the menu, choose **Client > Proxy server**.
4. Set **Proxy setting** as **Active**.
5. In the field for **No Proxy for the Following Addresses**, list any exceptions separated by semicolons.
6. In the **HTTP Protocol** and **HTTPs Protocol** fields, specify the connection details for your proxy server.

The SDK is unaware of the proxy server, and does not require any settings to use the SAP system's proxy server configuration.

Note

If you use [Amazon EC2 instance metadata authentication](#), then the SAP system cannot use the proxy server to access the local instance metadata at `http://169.254.169.254`. You must include `169.254.169.254` in the field for *No Proxy for the Following Addresses*.

Connection through a packet inspecting firewall

You can configure a packet inspecting firewall for outbound connection. These firewalls decrypt the SSL traffic, and then re-encrypt it before passing it on to the endpoint. This configuration usually requires the firewall to issue its own certificates to the SAP system that is consuming an AWS service. You must install your firewall's CA certificate in STRUST. For more information, see [HTTPS connectivity](#).

Gateway endpoints

Some AWS services offer gateway endpoints to provide a VPC with high-performance access without internet. These endpoints are transparent to SDK for SAP ABAP, and do not require any configuration.

For more information, see [Gateway endpoints](#).

Custom interface endpoints

If you need to override the default endpoint resolution with a custom endpoint, you can use an interface endpoint to provide your VPC with high-performance access without internet. For more information, see [Configure an interface endpoint](#).

When not using private DNS, these endpoints have their own DNS addresses, and an ABAP program must explicitly override the usual endpoint resolution logic. For more information, see AWS re:Post – [Why can't I resolve service domain names for an interface VPC endpoint?](#)

In the following example, an interface endpoint is created for AWS STS and Amazon Translate. The SAP system is not using private DNS, and calls the services with custom endpoint. The logical resources defined in `/AWS1/IMG` represent the physical interface endpoint addresses, such as `vpce-0123456789abcdef-hd52vxz.translate.us-west-2.vpce.amazonaws.com`. This avoids hard coding the DNS in code.

In the following code, the logical resources in `/AWS1/IMG` are first resolved to physical endpoint names. They are then provided to the factory methods of AWS session class (that uses AWS STS to assume an IAM role) and translate API class.

```
" This example assumes we have defined our logical endpoints in /AWS1/IMG
" as logical resources so that we don't hardcode our endpoints in code.
" The endpoints may be different in Dev, QA and Prod environments.
DATA(lo_config) = /aws1/cl_rt_config=>create( 'DEMO' ).
DATA(lo_resolver) = /aws1/cl_rt_lresource_resolver=>create( lo_config ).

" logical resource STS_ENDPOINT should resolve to the interface endpoint
" for example vpce-0123456789-abcdefg.sts.us-west-2.vpce.amazonaws.com
DATA(lv_sts_endpoint) = lo_resolver->resolve_lresource( 'STS_ENDPOINT' ).

" logical resource XL8_ENDPOINT should resolve to the interface endpoint
" e.g. vpce-0123456789abcdefg-12345567.translate.us-west-2.vpce.amazonaws.com
DATA(lv_xl8_endpoint) = lo_resolver->resolve_lresource( 'XL8_ENDPOINT' ).

" the session itself uses the sts service to assume a role, so the
" session creation process requires a custom endpoint, specified here
DATA(lo_session) = /aws1/cl_rt_session_aws=>create(
  iv_profile_id = 'DEMO'
  iv_custom_sts_endpoint = |https://{ lv_sts_endpoint }|
).
```

```

" now we create an API object, and override the default endpoint with
" the custom endpoint
DATA(lo_xl8)      = /aws1/cl_xl8_factory=>create(
  io_session = lo_session
  iv_custom_endpoint = |https://{ lv_xl8_endpoint }| " provide custom endpoint
).
" now calls to lo_xl8 go to custom endpoint...

```

As shown in the example, any method calls on `go_xl8` go to the endpoint `https://vpce-0123456789abcdefg-12345567.translate.us-west-2.vpce.amazonaws.com`.

Accessing endpoints in multiple Regions

AWS endpoint is automatically determined from your default AWS Region that is defined in the SDK profile. You can also specify a region programmatically, overriding the default region. This can be overridden in the factory `CREATE()` method, or later with the SDK's configuration object. For more information, see [Programmatic configuration](#).

In the following example, the factory `CREATE()` method is used to set the region and list the Amazon SQS queues in both `us-east-1` and `us-west-2` Regions.

```

REPORT zdemo_sqs_queue_list.
parameters: profile type /AWS1/RT_PROFILE_ID OBLIGATORY.

START-OF-SELECTION.
DATA(go_session) = /aws1/cl_rt_session_aws=>create( profile ).
data(lt_region) = VALUE stringtab(
  ( |us-east-1| )
  ( |us-west-2| )
).

LOOP AT lt_region INTO DATA(lv_region).
  DATA(go_sqs) = /aws1/cl_sqs_factory=>create(
    io_session = go_session
    iv_region = conv /AWS1/RT_REGION_ID( lv_region )
  ).
  WRITE: / lv_region COLOR COL_HEADING.
  LOOP AT go_sqs->listqueues( )->get_queueurls( ) INTO DATA(lo_url).
    WRITE: / lo_url->get_value( ).
  ENDLOOP.
ENDLOOP.

```

Service provider settings

Basis administrators sometimes need to control certain features of the SDK across the entire system, from client 000. This is a common scenario for hosting and service providers that operate systems in their own AWS account on behalf of their customers. AWS SDK for SAP ABAP supports Service Provider settings. These settings are configured in client 000, and affect the SDK across all clients. Service Provider settings are not supported in SDK for SAP ABAP - BTP edition.

Service Provider settings are configured in transaction /AWS1/IMG, and must be configured in client 000. Service Provider settings in other clients are ignored. The settings in client 000 take effect across all clients, and supercede other IMG settings in case of conflict.

Use the following steps to configure the Service Provider settings in client 000.

1. Expand the **Service Provider Settings** branch in transaction /AWS1/IMG.
2. Choose **Service Provider Guardrails**
3. Select **New Entries**, and adjust the settings based on your business requirements.
 - *Disable EC2 Metadata* – prevents the SDK from accessing EC2 instance metadata in all clients, even if an SDK Profile is configured to authenticate using EC2 instance metadata. The SDK raises an exception if an ABAP program attempts to access instance metadata using the SDK.
4. Select **Save**.

Refresh, trace, and telemetry topics for AWS SDK for SAP ABAP

This section covers the following topics.

Topics

- [SAP system refresh](#)
- [Trace](#)
- [Telemetry](#)

SAP system refresh

After a system refresh, the primary challenge for a Basis administrator is to ensure that the separate systems are not accessing each other's resources. For example, you may want to ensure

that your QA SAP system is not accessing the resources, such as an S3 bucket, of your Production landscape.

SDK for SAP ABAP provides a safety-conscious approach of *logical resources* to this challenge. A business analyst can take the following steps.

1. Define a logical resource, such as ZINVOICE_OUTBOUND.
2. Map all systems and clients in the development system.
3. Transport the configuration of ALL systems forward until the production landscape.

Basis steps after a refresh

1. Check the authentication
 - If the system is using Secret Access Key authentication, the SSF-encrypted credentials will be invalid because they are stored in master data. The credentials must be re-entered, which may require regenerating a new Secret Access Key in <https://console.aws.amazon.com/iam/>.
 - If the system is authenticating with EC2 instance metadata, no steps are required.

Check the trace settings

- In /AWS1/IMG, ensure that the trace settings are what you want. These settings are not transportable.

Trace

Trace output is controlled in the **IMG runtime settings**.

The trace levels that you can use are:

- **No Trace**
- **Trace API calls**
- **Trace API calls and payload**

This option contains unencrypted payload information.

- **Trace API calls, payload, and internal XML transformation**

This option contains unencrypted payload information.

If API trace is activated, traces are written to DIR_WORK in aws1_trace-*YYYY-MM-DD*.log file.

If payload trace is additionally activated, additional files with the title aws1_payload_* are created for each call and payload component. The payload trace length can be limited with the length limit applying to each individual payload trace fail.

Payload traces are primarily intended to collect information to be provided to AWS Support in the event of a serialization error. We recommend that you choose **No Trace** unless you're attempting to diagnose an SDK error.

Note

Payload traces can contain unencrypted business information. We recommend turning these traces on only for a request by AWS Support to help you troubleshoot. You can turn these traces off after resolution. Traces are not automatically deleted, and need to be removed by the system administrator when no longer needed.

These settings are not applicable to SDK for SAP ABAP - BTP edition.

Telemetry

SDKs send telemetry information to AWS Support. SDK for SAP ABAP collects the following information:

- OS release and patch level
- SAP_BASIS release and patch level
- SAP Kernel release and patch level

You can opt in to send the following information to AWS Support.

- SAP SID and instance name (host_sid_nn)
- SAP Client (SY-MANDT)
- Transaction code (SY-TCODE) and report (SY-REPID)

The additional information enables AWS Support to help you better. AWS Support can detect why a certain API call was made and can further find the relevant transaction in a SAP system.

Telemetry is limited to the SDK and API versions for SDK for SAP ABAP - BTP edition.

Using AWS SDK for SAP ABAP

SDK for SAP ABAP has two major components.

- SDK Runtime (package /AWS1/RT) – a set of objects that underpin the security, authentication, tracing, configuration, data conversion, and other cross-API functions. The API modules for Amazon S3, AWS STS, IAM Roles Anywhere, and Secrets Manager are mandatory.
- APIs (package /AWS1/API and its sub-packages) – a sub-package for each API where the objects of each API are completely independent of each other, ensuring that a change in one API does not break another API. To see a complete list of AWS SDK for SAP ABAP APIs, see [AWS SDK for SAP ABAP - API Reference Guide](#).

This section covers the following topics.

Topics

- [Representation of data in ABAP](#)
- [Amazon S3 example program](#)
- [SDK for SAP ABAP concepts](#)
- [AWS SDK for SAP ABAP features](#)
- [Building products with SDK](#)
- [Limitations](#)

Representation of data in ABAP

This section covers the following topics.

Topics

- [Data types](#)
- [AWS data types](#)

Data types

AWS services have a standard set of data types that must be mapped to ABAP data types. See the following table for more details.

AWS data type	ABAP data type	Comments
boolean	C	Single character "X" and " "
String	STRING	
Byte	INT2	INT2 has a larger range than 0-255. Most AWS services will truncate overflows but this behavior is not formally defined.
Short	INT2	
Integer	INT4	
Long	DEC19	INT8 is not available until ABAP 750. DEC19 is used for compatibility and consistency across all supported ABAP platforms.
Blob	XSTRING	Represents binary data
Float	STRING	While ABAP supports DECFLOATs, it cannot represent values such as NaN, Infinity and -Infinity. AWS SDK represents these internally as STRINGs, and converts them to DECFLOAT16 at runtime. If NaN, Infinity or +Infinity are represented, the developer may process these in response to a special set of exceptions or mappings.
Double	STRING	
bigInteger	STRING	These values represent infinite-length numbers that

AWS data type	ABAP data type	Comments
bigDecimal	STRING	cannot be represented in ABAP, and STRINGS are used instead of bigInteger.
Timestamp	TZNTSTMP	TZNTSTMP enables processing with native ABAP timestamp functions.

AWS services also return the following aggregate data types.

AWS data type	ABAP data type	Comments
Structure	Class	
Union	Class	A union is the same as a structure, except that a union will never have more than one field set. All other fields will be set to <i>No Value</i> .
Array	STANDARD TABLE	
Hash	HASHED TABLE	The hashed table will only have two columns: a KEY (string) and a VALUE (class).

AWS data types

The following approaches have been integrated to support AWS services in ABAP.

- Certain AWS data types cannot be represented in ABAP. For examples, the float data type in ABAP does not support the NaN, Infinity, or -Infinity values. Therefore, the float data type is represented as STRING and is translated to DECFLOAT16 at runtime.

- AWS data is represented on the wire as JSON or XML, and the values are optional. For example, see the following examples returned by an AWS service in JSON.

```
Fullname: {
  Firstname: "Ana",
  Middlename: "Carolina",
  Lastname: "Silva"
}
```

If Ana doesn't have a middle name, the service returns the following output.

```
Fullname: {
  Firstname: "Ana",
  Lastname: "Silva"
}
```

ABAP does not distinguish between *a string of length 0* and *a string that has no value*. Other languages might assign a NULL value to the string or wrap the string in a construct (such as, Java's `Optional<>` wrapper). These are not supported in ABAP. Therefore, SDK for SAP ABAP facilitates the distinction in values by providing variants of the *getter* method.

Amazon S3 example program

This section walks you through a simple example program to list the contents of an Amazon S3 bucket by calling `ListObjectsV2`.

Topics

- [Prerequisites](#)
- [Code](#)
- [Code sections](#)

Prerequisites

You must meet the following prerequisites to run this example program.

- You have an Amazon S3 bucket. In this tutorial, the bucket is named `demo-invoices.customer.com`.
- Transaction `/AWS1/IMG`:
 - Has a defined SDK profile named `DEMO_S3`.
 - In the SDK profile, the logical IAM role `TESTUSER` must map to an IAM role, such as `arn:aws:iam::111122223333:role/SapDemoFinance` which grants `s3:ListBucket` permission to list the contents of your Amazon S3 bucket.
 - Has a logical resource named `DEMO_BUCKET` that is mapped to your Amazon S3 bucket with the SID and client of your SAP system.
- Your user has a PFCG role that:
 - Authorizes the user to access `DEMO_S3` SDK profile via auth object - `/AWS1/SESS`.
 - Authorizes the user for logical IAM role `TESTUSER` access via auth object - `/AWS1/LROL`.
- Your SAP system can authenticate itself to AWS using the method defined in the SDK profile.
- Your Amazon EC2 instance profile grants your SAP system the rights to `sts:assumeRole` in the IAM role `arn:aws:iam::111122223333:role/SapDemoFinance` mapped in the SDK profile.

Code

The following code block demonstrates what your code would look like.

```
REPORT  zdemo_s3_listbuckets.

START-OF-SELECTION.
  PARAMETERS pv_lres TYPE  /aws1/rt_resource_logical
                DEFAULT 'DEMO_BUCKET' OBLIGATORY.

  DATA(go_session) = /aws1/cl_rt_session_aws=>create( 'DEMO_S3' ).
  DATA(gv_bucket)  = go_session->resolve_lresource( pv_lres ).

  DATA(go_s3)      = /aws1/cl_s3_factory=>create( go_session ).

  TRY.
    DATA(lo_output) = go_s3->listobjectsv2(
      iv_bucket = CONV string( gv_bucket )
      iv_maxkeys = 100
    ).
```

```

LOOP AT lo_output->get_contents( ) INTO DATA(lo_object).
  DATA lv_mdate TYPE datum.
  CONVERT TIME STAMP lo_object->get_lastmodified( )
    TIME ZONE 'UTC'
    INTO DATE lv_mdate.
  WRITE: / CONV text30( lo_object->get_key( ) ),
    lv_mdate, lo_object->get_size( ).
ENDLOOP.
CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
  DATA(lv_msg) = lo_ex->if_message~get_text( ).
  MESSAGE lv_msg TYPE 'I'.
ENDTRY.

```

Code sections

The following is a review of the code in sections.

```

PARAMETERS pv_lres TYPE /aws1/rt_resource_logical
  DEFAULT 'DEMO_BUCKET' OBLIGATORY.

```

The user can't specify a physical bucket name. They specify a logical bucket and the system administrators (specifically the business analyst) in coordination with the AWS administrator map logical buckets to physical buckets in /AWS1/IMG. In most business scenarios, the user doesn't have a chance to choose the logical bucket — the logical resource ID is hard coded in the code or configured in a custom configuration table.

```

DATA(go_session) = /aws1/cl_rt_session_aws=>create( 'DEMO_S3' ).

```

This line establishes a security session and declares that this ABAP program expects to use the DEMO_S3 SDK profile. This call is the connection to the SDK configuration and pulls in the default Region, authentication settings, and the desired IAM Role. A call to AUTHORIZATION-CHECK is automatically made to ensure that authorization object /AWS1/SESS is satisfied. Additionally, AUTHORIZATION-CHECK calls will be made to determine the most powerful (lower sequence number) logical IAM role the user is authorized for, based on the authorization object /AWS1/LROL. The SDK will assume that the IAM role is mapped to the logical IAM role for the SID and client. Then, the session object activates tracing based on the trace settings in the IMG.

If the user is not authorized for the requested SDK profile or for any available logical IAM role, an exception will be raised.

```
DATA(gv_bucket) = go_session->resolve_lresource( pv_lres ).
```

This line resolves the logical resource to a physical bucket name. If the logical resource cannot be resolved because the configuration has no mapping for this SID/client combination, an exception will be raised.

```
DATA(go_s3) = /aws1/cl_s3_factory=>create( go_session ).
```

This line creates an API object for Amazon S3 using the `create()` method of `/aws1/cl_s3_factory`. The returned object is of type `/aws1/if_s3` which is the interface for an Amazon S3 API. A separate API object must be created for each service. For example, if an ABAP program is consuming Amazon S3, AWS Lambda, and DynamoDB, then it creates API objects from `/aws1/cl_s3_factory`, `/aws1/cl_lmd_factory`, and `/aws1/cl_dyn_factory`.

There are some optional parameters to the constructor which enable you to specify the Region if you want to override the default Region configured IMG. In this way, there can be two instances of `/aws1/if_s3`, one for `us-east-1` and one for `us-west-2`, if you want to copy objects from a bucket in one Region to a bucket in another Region. Similarly, you can create two different security session objects and use them to create two separate instances of `/aws1/cl_s3`, if you need a report to read from a finance-related bucket and write objects to a logistics-related bucket.

```
DATA(lo_output) = go_s3->listobjectsv2(  
    iv_bucket = CONV string( gv_bucket )  
    iv_maxkeys = 100  
).
```

This line is a call to `ListObjectsv2`. It requires simple input arguments and returns a single object. These objects may represent deep JSON and XML data, de-serialized into an ABAP object-oriented construct. It can be quite complicated in some cases. Now, you only need to process the output to list the contents of the bucket.

```
LOOP AT lo_output->get_contents( ) INTO DATA(lo_object).
  DATA lv_mdate TYPE datum.
  CONVERT TIME STAMP lo_object->get_lastmodified( )
    TIME ZONE 'UTC'
    INTO DATE lv_mdate.
  WRITE: / CONV text30( lo_object->get_key( ) ),
    lv_mdate, lo_object->get_size( ).
ENDLOOP.
```

The data is accessed using a GET...() style method that hides the internal representation of the data. GET_CONTENTS() returns an ABAP table and each row itself contains an object representing a single Amazon S3 entry. In most cases, AWS SDK takes this object-oriented approach and all data is represented as objects and tables. The LastModified field is represented as a timestamp that can be converted to a date with the ABAP-native CONVERT TIME STAMP command. the GET_SIZE() returns an INT4 for easy math and formatting operations.

```
CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
  DATA(lv_msg) = lo_ex->if_message~get_text( ).
  MESSAGE lv_msg TYPE 'I'.
```

All errors – connection, 4xx client, 5xx server, or any ABAP error, such as authorization or configuration errors, are represented as exceptions. You can tackle each exception separately. You have the choice of whether an exception should be handled as an informational error, a retry, warning, short dump, or any other kind of handling.

SDK for SAP ABAP concepts

This section covers the basic concepts of AWS SDK for SAP ABAP.

Topics

- [API classes](#)
- [Additional objects](#)
- [Structure classes](#)
- [Arrays](#)
- [Maps](#)
- [Higher level functions](#)

API classes

Each AWS service is assigned a three-letter acronym or TLA. The service is represented by an interface in the `/AWS1/IF_<TLA>` format. We will call this the service interface. The API class is in the `/AWS1/API_<TLA>` package. The service interface consists of one method for each AWS operation (we will call these methods Operation Methods). To see a complete module list of AWS SDK for SAP ABAP TLAs, see [AWS SDK for SAP ABAP - Module List](#).

Each operation method has some IMPORTING arguments and at the most one RETURNING argument. Often, these arguments will be objects with complicated constructors and a long set of GET...() methods. In many cases, the objects will contain nested objects, recursive references, tables of objects, tables of tables, and so forth. This is because AWS services are passing deep XML and JSON structures, which cannot be represented by a flat set of arguments.

The RETURNING argument is always a class, even if the class contains only a single attribute.

Additional objects

In addition to containing the primary API class, each API package contains various related repository and data dictionary objects.

- A class for each structure-type object.
- A class for any primitive data type which appears in a table. For example, if a service returns a table of strings, the ABAP API will represent it as a table of objects, where each object is a wrapper class that encapsulates a string. This is so that the wrapper class can hide the details of representing a null string that cannot be represented natively in ABAP.
- An exception class for any specific errors defined by the service.
- Data elements for each primitive data type. Each data type has its own data element in order to be self-documenting.
- Additional objects for internal processing, such as XSLT transforms for serializing and de-serializing XML and JSON payloads.

Structure classes

Most AWS data, sent and received by the service, is represented by AWS SDK as classes. These classes represent structures of data and hide the internal details of the storage. In particular, the classes hide the way the SDK represents *this field has no value*.

For each field in a structure class, there are three methods.

GET_field()

The GET_field() method

- Returns the value of the field, or
- If the field has no value, it returns a default value, which you can set as an optional parameter.

For example, consider the following code that prints the location constraint of a bucket.

```
DATA(lo_location) = go_s3->getbucketlocation( iv_bucket = CONV string( gv_bucket ) ).  
WRITE: / 'Bucket Location: ',  
       lo_location->get_locationconstraint( ).
```

If the bucket has no location constraint at all (as in the case of us-east-1), then GET_LOCATIONCONSTRAINT() will return the empty string. You can override this behavior and specify the desired value if the field has no value at all.

```
DATA(lo_location) = go_s3->getbucketlocation( iv_bucket = CONV string( gv_bucket ) ).  
WRITE: / 'Bucket Location: ',  
       lo_location->get_locationconstraint( iv_value_if_missing = 'assuming us-east-1' ).
```

Now the program will write Bucket Location: assuming us-east-1 if getbucketlocation()'s result does not return a location.

It is possible to ask the GET() method to return a specific result if the requested value is completely missing, see the following code example.

```
data(lo_location) = go_s3->GETBUCKETLOCATION(  
  new /AWS1/CL_S3_GET_BUCKET_LOC_REQ( iv_bucket = gv_bucket )  
) .  
write: / 'Location constraint: ',  
       lo_location->GET_LOCATIONCONSTRAINT( 'NopeNopeNope' ).
```

In this case, if there is no location constraint, GET_LOCATIONCONSTRAINT() will return NopeNopeNope.

HAS_field()

HAS_field() method is a way to find out if the field has a value or not. See the following example.

```
if NOT lo_location->HAS_LOCATIONCONSTRAINT( ).
    write: / 'There is no location constraint'.
endif.
```

If a certain field is known to always have a value, there will be no HAS_field() method.

ASK_field()

The ASK_field() method returns the value of the field or raises an exception if it has no value. This is a convenient way to process a number of fields, and bail out of the logic and take a different approach if any of the fields have no value.

```
TRY.
    WRITE: / 'Location constraint: ', lo_location->ask_locationconstraint( ).
CATCH /aws1/cx_rt_value_missing.
    WRITE: / 'Never mind, there is no location constraint'.
ENDTRY.
```

Note that /AWS1/CX_RT_VALUE_MISSING is a static exception and you will get a warning if you choose not to catch it.

Best practices

In general, you can use the GET_field() method as it treats a null string as an empty string and is the most ABAP-like of the three options. However, it does not let you easily distinguish between situations where the field has a blank value and where the field has no value. If your business logic depends on distinguishing missing data versus blank data, then the HAS or ASK methods let you handle these cases.

Arrays

Arrays are represented as ABAP standard tables of objects.

A JSON array can contain null values, such as the following array: ['cat', 'dog', null, 'horse']. This is referred to as a sparse array. It is represented in ABAP as an internal table of object references, and the null value is represented in the table as a true ABAP null value. When iterating through a sparse table, you must check for null values to avoid accessing a null object and getting a CX_SY_REF_IS_INITIAL exception. In practice, sparse arrays are rare in AWS services.

To initialize an array of objects, it is convenient to use the new ABAP 7.40 constructs. Consider this launch of an Amazon EC2 instance with several security groups assigned:

```
ao_ec2->runinstances(  
  iv_imageid           = lo_latest_ami->get_imageid( )  
  iv_instancetype      = 't2.micro'  
  iv_maxcount          = 1  
  iv_mincount          = 1  
  it_securitygroupids  = VALUE /aws1/  
cl_ec2secgrpiddstrlist_w=>tt_securitygroupidstringlist(  
  ( NEW /aws1/  
cl_ec2secgrpiddstrlist_w( 'sg-12345678' ) )  
  ( NEW /aws1/  
cl_ec2secgrpiddstrlist_w( 'sg-55555555' ) )  
  ( NEW /aws1/  
cl_ec2secgrpiddstrlist_w( 'sg-99999999' ) )  
  )  
  iv_subnetid         = ao_snet->get_subnetid( )  
  it_tagsspecifications = make_tag_spec( 'instance' )  
)
```

Maps

JSON maps are represented in ABAP as Hashed Tables where each table row has only two components.

- KEY – a string which is the UNIQUE KEY of the table.
- VALUE – an object containing the value.

A map is one of the very few cases where AWS SDK uses a true structure, rather than a class. This is necessary because ABAP hashed tables cannot have an object reference as the key field, and AWS map keys are always non-null strings.

Higher level functions

The [API classes](#) described in the preceding section precisely mirror the AWS service APIs and represent those APIs as familiar ABAP classes. In some cases, the SDK also includes higher level functions that build on top of the API classes to simplify certain operations. The higher level functions are included for programmer convenience and do not replace the lower-level API classes.

If the SDK includes higher level functions for a module, they are included in the same transport and can be accessed through a factory class called `/AWS1/CL_TLA_L2_FACTORY`. The factory class includes methods to create various higher level clients for the module that are documented along with the rest of the API with the [API documentation](#).

AWS SDK for SAP ABAP features

AWS SDK for SAP ABAP provides the following features.

Topics

- [Programmatic configuration](#)
- [Waiters](#)
- [Paginators](#)
- [Retry behavior](#)

Programmatic configuration

Use `/n/AWS1/IMG` IMG transaction for AWS SDK for SAP ABAP, and Custom Business Configuration application for AWS SDK for SAP ABAP - BTP edition for programmatic configuration.

To begin programmatic configuration, begin by retrieving a configuration object with the `get_config()` command.

```
data(lo_config) = lo_s3->get_config( ).
```

Each configuration object implements `/AWS1/IF_RT_CONFIG` interface that includes GETters and SETters corresponding to the IMG. For example, the default region can be overridden. See the following example command.

```
lo_s3->get_config( )->/aws1/if_rt_config~set_region( 'us-east-1' ).
```

Some configuration objects have no IMG representation and can only be set programmatically, such as maximum retry attempts. See the following example command.

```
lo_s3->get_config( )->/aws1/if_rt_config~set_max_attempts( 10 ).
```

The configuration object of AWS services can also include service specific methods that are not represented in `/aws1/if_rt_config`. For example, Amazon S3 can address a bucket named `foobucket` using either `foobucket.s3.region.amazonaws.com` virtual endpoint or `s3.region.amazonaws.com/foobucket` path style. You can enforce the use of path style with the following example command.

```
lo_s3->get_config( )->set_forcepathstyle( abap_true ).
```

For more information about service configurations, see [AWS SDK for SAP ABAP – API Reference Guide](#).

Waiters

When working with asynchronous AWS APIs, you need to wait for a certain resource to become available before taking further actions. For example, the `CREATETABLE()` API of Amazon DynamoDB responds right away with table status `CREATING`. You can initiate read or write operations only after the status of the table has changed to `ACTIVE`. Waiters give you the ability to confirm that AWS resources are in a particular state before performing actions on them.

Waiters use service operations to poll the status of AWS resources until the resource reaches the intended state or until it is determined that the resource doesn't reach the desired state. It can be time-consuming and error-prone to write the code to poll AWS resources continually. Waiters help in simplifying this complexity by taking the responsibility of performing polls on your behalf.

See the following Amazon S3 example using waiter.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).  
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).
```

```
“ Create a bucket - initiates the process of creating an S3 bucket and might return  
before the bucket exists
```

```
lo_s3#createbucket( iv_bucket = |example-bucket| ).  
  
“ Wait until the newly created bucket becomes available  
lo_s3->get_waiter( )->bucketexists(  
    iv_max_wait_time = 200  
    iv_bucket = |example-bucket|  
).
```

- In this example, Amazon S3 client is used to create a bucket. The `get_waiter()` command is implemented to specify when the `bucketexists`.
- You must specify the `iv_max_wait_time` parameter for each waiter. It represents the total amount of time a waiter must wait before completion. In the preceding example, a waiter can run for 200 seconds.
- You may need to provide additional inputs for required parameters. In the preceding example, Amazon S3 bucket name is required for `iv_bucket` parameter.
- `/AWS1/CX_RT_WAITER_FAILURE` exception indicates that the waiter exceeded the maximum time specified in `iv_max_wait_time` parameter.
- `/AWS1/CX_RT_WAITER_TIMEOUT` exception indicates that the waiter has stopped due to not reaching the desired state.

Paginators

Some AWS service operations offer paged responses. They are paginated to return a fixed amount of data with each response. You need to make subsequent requests with a token or a marker to retrieve the entire set of results. For instance, the `ListObjectsV2` Amazon S3 operation return up to 1,000 objects at a time. You must make subsequent requests with the appropriate token to get the next page of results.

Pagination is the process of sending successive requests to pick up where a previous request left off. Paginators are iterators of results provided by SDK for SAP ABAP. You can use paginated APIs with ease, and without understanding the underlying mechanism of API using pagination tokens.

Working with paginators

You can create paginators with the `get_pagination()` method that returns a paginator object. The paginator object calls the operation being paginated. The paginator object accepts required

parameters to be provided to the underlying API. This process returns an iterator object that can be used to iterate over paginated results, using the `has_next()` and `get_next()` methods.

- `has_next()` returns a boolean value indicating if there are more responses or pages available for the called operation.
- `get_next()` returns the operation response.

The following example list all objects in an S3 bucket retrieved by using paginator.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( 'DEMO' ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

TRY.
  DATA(lo_paginator) = lo_s3->get_paginator( ).
  DATA(lo_iterator) = lo_paginator->listobjectsv2(
    iv_bucket = 'example_bucket'
  ).
  WHILE lo_iterator->has_next( ).
    DATA(lo_output) = lo_iterator->get_next( ).
    LOOP AT lo_output->get_contents( ) INTO DATA(lo_object).
      WRITE: / lo_object->get_key( ), lo_object->get_size( ).
    ENDLLOOP.
  ENDWHILE.
CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
  MESSAGE lo_ex->if_message~get_text( ) TYPE 'I'.
ENDTRY.
```

Retry behavior

SDK for SAP ABAP enables you to configure the maximum number of retries for requests to AWS services that fail due to throttling or transient errors. The number of retries allowed at the service client level, that is the number of times the SDK retries the operation before failing and raising an exception is specified by the `AV_MAX_ATTEMPTS` attribute in the service configuration object. When a service client object is created, the SDK configures `AV_MAX_ATTEMPTS` attribute to a default value of 3. The service configuration object may be used to programmatically set the maximum retry attempt to a desired value. See the following example for more details.

```
" Retrieve configuration object using Amazon S3 service's get_config( ) method
DATA(lo_config) = lo_s3->get_config( ).
```



```
" Set the maximum number of retries to 5
lo_config->/aws1/if_rt_config~set_max_attempts( 5 ).
```

```
" Get the value of the maximum retry attempt.
DATA(lv_max_retry_attempts) = lo_config->/aws1/if_rt_config~get_max_attempts( ).
```

Note

Although the configuration object ABAP SDK allows *retry mode* to be set with the `/AWS1/IF_RT_CONFIG~SET_RETRY_MODE()` method, the SDK only supports the standard retry mode. For more information, see [Retry behavior](#) in AWS SDKs and Tools Reference Guide.

Building products with SDK

A product or ABAP add-on that consumes AWS services can enhance and extend the capabilities of the SDK. You can build such products to use with the SDK.

Topics

- [Setting a product ID](#)

Setting a product ID

It is recommended that you set a product ID when establishing a session inside a product or add-on. See the following example for more details.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( 'DEMO' ).
lo_session->set_product_id( 'INVOICE_ANALYZER' ).
```

The product ID must only contain letters, numbers, and underscores with no spaces or special characters. You can match it to the product's technical name or any other identifier. If you develop multiple products or add-ons, the product ID must be unique for each product. For example, the product IDs for Invoice Analyzer, Tax Calculator, and Pricing Engine products can be `INVOICE_ANALYZER`, `TAX_CALCULATOR`, and `PRICING_ENGINE`.

Adding a product ID to the session enhances the telemetry that is sent to AWS with each service call. The product ID and the namespace of the object making the call is included in the telemetry. With this telemetry, AWS Support can identify the product that is making the call in case of your

customer facing issues with the SDK. It can help clarify that the call is actually being made by the product, and not your customer's code.

Limitations

AWS SDK for SAP ABAP includes SDK modules for all AWS services. Some of these modules may have limitations, as described here.

- Modules that rely on MQTT protocol bindings, such as `iotevents`, will not work. MQTT is not an HTTP-based protocol and is currently not supported by AWS SDK for SAP ABAP.
- Modules that rely on HTTP/2 streaming features are not yet supported. Certain operations of services that work with event streams are not yet supported, and media streaming operations of services, such as Amazon Kinesis Video Streams will not work.

AWS SDK for SAP ABAP has the following feature limitations.

- The following Amazon S3 features are not yet supported.
 - Multi-Region access points
 - Amazon S3 client-side encryption

AWS SDK for SAP ABAP - BTP edition has the following limitations during the developer preview.

- Some modules may not be available.
- It cannot be uninstalled.
- It is updated less frequently.

SDK for SAP ABAP code examples

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

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Cross-service examples are sample applications that work across multiple AWS services.

Examples

- [Actions and scenarios using SDK for SAP ABAP](#)

Actions and scenarios using SDK for SAP ABAP

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with AWS services.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Services

- [Amazon Bedrock Runtime examples using SDK for SAP ABAP](#)
- [CloudWatch examples using SDK for SAP ABAP](#)
- [DynamoDB examples using SDK for SAP ABAP](#)
- [Amazon EC2 examples using SDK for SAP ABAP](#)
- [Kinesis examples using SDK for SAP ABAP](#)
- [Lambda examples using SDK for SAP ABAP](#)
- [Amazon S3 examples using SDK for SAP ABAP](#)
- [SageMaker examples using SDK for SAP ABAP](#)

- [Amazon SNS examples using SDK for SAP ABAP](#)
- [Amazon SQS examples using SDK for SAP ABAP](#)
- [Amazon Textract examples using SDK for SAP ABAP](#)
- [Amazon Translate examples using SDK for SAP ABAP](#)

Amazon Bedrock Runtime examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Bedrock Runtime.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Anthropic Claude](#)
- [Stable Diffusion](#)

Anthropic Claude

InvokeModel

The following code example shows how to send a text message to Anthropic Claude, using the Invoke Model API.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Invoke the Anthropic Claude 2 foundation model to generate text. This example uses features of /US2/CL_JSON which might not be available on some NetWeaver versions.

```

"Claude V2 Input Parameters should be in a format like this:
* {
*   "prompt": "\n\nHuman:\n\nTell me a joke\n\nAssistant:\n",
*   "max_tokens_to_sample": 2048,
*   "temperature": 0.5,
*   "top_k": 250,
*   "top_p": 1.0,
*   "stop_sequences": []
* }

DATA: BEGIN OF ls_input,
      prompt                TYPE string,
      max_tokens_to_sample TYPE /aws1/rt_shape_integer,
      temperature           TYPE /aws1/rt_shape_float,
      top_k                 TYPE /aws1/rt_shape_integer,
      top_p                 TYPE /aws1/rt_shape_float,
      stop_sequences        TYPE /aws1/rt_stringtab,
END OF ls_input.

"Leave ls_input-stop_sequences empty.
ls_input-prompt = |\n\nHuman:\n\n{ iv_prompt }\n\nAssistant:\n|.
ls_input-max_tokens_to_sample = 2048.
ls_input-temperature = '0.5'.
ls_input-top_k = 250.
ls_input-top_p = 1.

"Serialize into JSON with /ui2/cl_json -- this assumes SAP_UI is installed.
DATA(lv_json) = /ui2/cl_json=>serialize(
  data = ls_input
  pretty_name = /ui2/cl_json=>pretty_mode-low_case ).

TRY.
  DATA(lo_response) = lo_bdr->invokemodel(
    iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
    iv_modelid = 'anthropic.claude-v2'
    iv_accept = 'application/json'
    iv_contenttype = 'application/json' ).

"Claude V2 Response format will be:
* {

```

```

*      "completion": "Knock Knock...",
*      "stop_reason": "stop_sequence"
*    }
DATA: BEGIN OF ls_response,
      completion TYPE string,
      stop_reason TYPE string,
END OF ls_response.

/ui2/cl_json=>deserialize(
  EXPORTING jsonx = lo_response->get_body( )
           pretty_name = /ui2/cl_json=>pretty_mode-camel_case
  CHANGING data = ls_response ).

DATA(lv_answer) = ls_response-completion.
CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
WRITE / lo_ex->get_text( ).
WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

ENDTRY.

```

Invoke the Anthropic Claude 2 foundation model to generate text using L2 high level client.

```

TRY.
  DATA(lo_bdr_l2_claude) = /aws1/cl_bdr_l2_factory=>create_claude_2( lo_bdr ).
  " iv_prompt can contain a prompt like 'tell me a joke about Java
  programmers'.
  DATA(lv_answer) = lo_bdr_l2_claude->prompt_for_text( iv_prompt ).
  CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
  WRITE / lo_ex->get_text( ).
  WRITE / |Don't forget to enable model access at https://
  console.aws.amazon.com/bedrock/home?#/modelaccess|.

  ENDTRY.

```

- For API details, see [InvokeModel](#) in *AWS SDK for SAP ABAP API reference*.

Stable Diffusion

InvokeModel

The following code example shows how to invoke Stability.ai Stable Diffusion XL on Amazon Bedrock to generate an image.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an image with Stable Diffusion.

```
"Stable Diffusion Input Parameters should be in a format like this:
*  {
*    "text_prompts": [
*      {"text":"Draw a dolphin with a mustache"},
*      {"text":"Make it photorealistic"}
*    ],
*    "cfg_scale":10,
*    "seed":0,
*    "steps":50
*  }
TYPES: BEGIN OF prompt_ts,
        text TYPE /aws1/rt_shape_string,
        END OF prompt_ts.

DATA: BEGIN OF ls_input,
        text_prompts TYPE STANDARD TABLE OF prompt_ts,
        cfg_scale    TYPE /aws1/rt_shape_integer,
        seed         TYPE /aws1/rt_shape_integer,
        steps       TYPE /aws1/rt_shape_integer,
        END OF ls_input.

APPEND VALUE prompt_ts( text = iv_prompt ) TO ls_input-text_prompts.
ls_input-cfg_scale = 10.
ls_input-seed = 0. "or better, choose a random integer.
ls_input-steps = 50.
```

```

DATA(lv_json) = /ui2/cl_json=>serialize(
  data = ls_input
  pretty_name = /ui2/cl_json=>pretty_mode-low_case ).

TRY.
  DATA(lo_response) = lo_bdr->invokemodel(
    iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
    iv_modelid = 'stability.stable-diffusion-xl-v0'
    iv_accept = 'application/json'
    iv_contenttype = 'application/json' ).

  "Stable Diffusion Result Format:
*   {
*     "result": "success",
*     "artifacts": [
*       {
*         "seed": 0,
*         "base64": "iVBORw0KGgoAAAANSUhEUgAAAgAAA...
*         "finishReason": "SUCCESS"
*       }
*     ]
*   }
  TYPES: BEGIN OF artifact_ts,
    seed          TYPE /aws1/rt_shape_integer,
    base64        TYPE /aws1/rt_shape_string,
    finishreason  TYPE /aws1/rt_shape_string,
  END OF artifact_ts.

  DATA: BEGIN OF ls_response,
    result        TYPE /aws1/rt_shape_string,
    artifacts     TYPE STANDARD TABLE OF artifact_ts,
  END OF ls_response.

  /ui2/cl_json=>deserialize(
    EXPORTING jsonx = lo_response->get_body( )
    pretty_name = /ui2/cl_json=>pretty_mode-camel_case
    CHANGING data = ls_response ).
  IF ls_response-artifacts IS NOT INITIAL.
    DATA(lv_image) =
      cl_http_utility=>if_http_utility~decode_x_base64( ls_response-artifacts[ 1 ]-
        base64 ).
    ENDIF.
  CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
  WRITE / lo_ex->get_text( ).

```



```
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

Invoke the Stability.ai Stable Diffusion XL foundation model to generate images using L2 high level client.

```
    TRY.
        DATA(lo_bdr_l2_sd) = /aws1/
cl_bdr_l2_factory=>create_stable_diffusion_10( lo_bdr ).
        " iv_prompt contains a prompt like 'Show me a picture of a unicorn reading an
enterprise financial report'.
        DATA(lv_image) = lo_bdr_l2_sd->text_to_image( iv_prompt ).
        CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

- For API details, see [InvokeModel](#) in *AWS SDK for SAP ABAP API reference*.

CloudWatch examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with CloudWatch.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

DeleteAlarms

The following code example shows how to use DeleteAlarms.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_cwt->deletealarms(  
    it_alarmnames = it_alarm_names  
  ).  
  MESSAGE 'Alarms deleted.' TYPE 'I'.  
CATCH /aws1/cx_cwtresourcenotfound .  
  MESSAGE 'Resource being accessed is not found.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for SAP ABAP API reference*.

DescribeAlarms

The following code example shows how to use DescribeAlarms.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_cwt->describealarms(
        " oo_result is returned
for testing purposes. "
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarms retrieved.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeAlarms](#) in *AWS SDK for SAP ABAP API reference*.

DisableAlarmActions

The following code example shows how to use `DisableAlarmActions`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Disables actions on the specified alarm. "
TRY.
    lo_cwt->disablealarmactions(
```

```
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarm actions disabled.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for SAP ABAP API reference*.

EnableAlarmActions

The following code example shows how to use `EnableAlarmActions`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Enable actions on the specified alarm."
TRY.
    lo_cwt->enablealarmactions(
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarm actions enabled.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for SAP ABAP API reference*.

ListMetrics

The following code example shows how to use ListMetrics.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"The following list-metrics example displays the metrics for Amazon CloudWatch."
TRY.
    oo_result = lo_cwt->listmetrics(           " oo_result is returned for
testing purposes. "
    iv_namespace = iv_namespace
    ).
    DATA(lt_metrics) = oo_result->get_metrics( ).
    MESSAGE 'Metrics retrieved.' TYPE 'I'.
CATCH /aws1/cx_cwtinvparamvalueex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListMetrics](#) in *AWS SDK for SAP ABAP API reference*.

PutMetricAlarm

The following code example shows how to use PutMetricAlarm.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
```

```
lo_cwt->putmetricalarm(  
    iv_alarmname           = iv_alarm_name  
    iv_comparisonoperator  = iv_comparison_operator  
    iv_evaluationperiods   = iv_evaluation_periods  
    iv_metricname         = iv_metric_name  
    iv_namespace          = iv_namespace  
    iv_statistic          = iv_statistic  
    iv_threshold          = iv_threshold  
    iv_actionsenabled     = iv_actions_enabled  
    iv_alarmdescription   = iv_alarm_description  
    iv_unit               = iv_unit  
    iv_period             = iv_period  
    it_dimensions        = it_dimensions  
).  
MESSAGE 'Alarm created.' TYPE 'I'.  
CATCH /aws1/cx_cwtlimitexceededfault.  
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with alarms

The following code example shows how to:

- Create an alarm.
- Disable alarm actions.
- Describe an alarm.
- Delete an alarm.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
  lo_cwt->putmetricalarm(
    iv_alarmname           = iv_alarm_name
    iv_comparisonoperator  = iv_comparison_operator
    iv_evaluationperiods   = iv_evaluation_periods
    iv_metricname          = iv_metric_name
    iv_namespace           = iv_namespace
    iv_statistic           = iv_statistic
    iv_threshold           = iv_threshold
    iv_actionsenabled      = iv_actions_enabled
    iv_alarmdescription    = iv_alarm_description
    iv_unit                = iv_unit
    iv_period              = iv_period
    it_dimensions          = it_dimensions
  ).
  MESSAGE 'Alarm created' TYPE 'I'.
CATCH /aws1/cx_cwtlimitexceededfault.
  MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the created alarm."
CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions."
TRY.
  lo_cwt->disablealarmactions(
    it_alarmnames          = lt_alarmnames
  ).
  MESSAGE 'Alarm actions disabled' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
  DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception->av_err_code }"|
- { lo_disablealarm_exception->av_err_msg }|.
  MESSAGE lv_disablealarm_error TYPE 'E'.
ENDTRY.

"Describe alarm using the same ABAP internal table."
TRY.
```

```

        oo_result = lo_cwt->describealarms(
returned for testing purpose " " oo_result is
        it_alarmnames          = lt_alarmnames
        ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
        CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDMETHOD.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
            it_alarmnames = lt_alarmnames
        ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
        CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDMETHOD.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [DeleteAlarms](#)
 - [DescribeAlarms](#)
 - [DisableAlarmActions](#)
 - [PutMetricAlarm](#)

DynamoDB examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with DynamoDB.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateTable

The following code example shows how to use CreateTable.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
    ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                          iv_keytype = 'HASH' ) )
    ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                          iv_keytype = 'RANGE' ) ) ).
  DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
    ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                     iv_attributetype = 'N' ) )
    ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                     iv_attributetype = 'S' ) ) ).

  " Adjust read/write capacities as desired.
  DATA(lo_dynprovthroughput) = NEW /aws1/cl_dynprovthroughput(
    iv_readcapacityunits = 5
    iv_writecapacityunits = 5 ).
  oo_result = lo_dyn->createtable(
    it_keyschema = lt_keyschema
```

```

        iv_tablename = iv_table_name
        it_attributedefinitions = lt_attributedefinitions
        io_provisionedthroughput = lo_dynprovthroughput ).
    " Table creation can take some time. Wait till table exists before
    returning.
    lo_dyn->get_waiter( )->tableexists(
        iv_max_wait_time = 200
        iv_tablename      = iv_table_name ).
    MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
    " This exception can happen if the table already exists.
    CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
        { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [CreateTable](#) in *AWS SDK for SAP ABAP API reference*.

DeleteItem

The following code example shows how to use DeleteItem.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
        DATA(lo_resp) = lo_dyn->deleteitem(
            iv_tablename      = iv_table_name
            it_key             = it_key_input ).
        MESSAGE 'Deleted one item.' TYPE 'I'.
    CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
        TYPE 'E'.
    CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.

```

```
CATCH /aws1/cx_dyntransactconflictex.  
    MESSAGE 'Another transaction is using the item' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteItem](#) in *AWS SDK for SAP ABAP API reference*.

DeleteTable

The following code example shows how to use DeleteTable.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_dyn->deletetable( iv_tablename = iv_table_name ).  
    " Wait till the table is actually deleted.  
    lo_dyn->get_waiter( )->tablenotexists(  
        iv_max_wait_time = 200  
        iv_tablename      = iv_table_name ).  
    MESSAGE 'Table ' && iv_table_name && ' deleted.' TYPE 'I'.  
CATCH /aws1/cx_dynresourcenotfoundex.  
    MESSAGE 'The table ' && iv_table_name && ' does not exist' TYPE 'E'.  
CATCH /aws1/cx_dynresourceinuseex.  
    MESSAGE 'The table cannot be deleted since it is in use' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteTable](#) in *AWS SDK for SAP ABAP API reference*.

DescribeTable

The following code example shows how to use DescribeTable.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_dyn->describetable( iv_tablename = iv_table_name ).
    DATA(lv_tablename) = oo_result->get_table( )->ask_tablename( ).
    DATA(lv_tablearn) = oo_result->get_table( )->ask_tablearn( ).
    DATA(lv_tablestatus) = oo_result->get_table( )->ask_tablestatus( ).
    DATA(lv_itemcount) = oo_result->get_table( )->ask_itemcount( ).
    MESSAGE 'The table name is ' && lv_tablename
            && '. The table ARN is ' && lv_tablearn
            && '. The tablestatus is ' && lv_tablestatus
            && '. Item count is ' && lv_itemcount TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table ' && lv_tablename && ' does not exist' TYPE 'E'.
    ENDTRY.

```

- For API details, see [DescribeTable](#) in *AWS SDK for SAP ABAP API reference*.

GetItem

The following code example shows how to use GetItem.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_item = lo_dyn->getitem(
        iv_tablename          = iv_table_name

```

```

        it_key                = it_key ).
DATA(lt_attr) = oo_item->get_item( ).
DATA(lo_title) = lt_attr[ key = 'title' ]-value.
DATA(lo_year) = lt_attr[ key = 'year' ]-value.
DATA(lo_rating) = lt_attr[ key = 'rating' ]-value.
MESSAGE 'Movie name is: ' && lo_title->get_s( )
        && 'Movie year is: ' && lo_year->get_n( )
        && 'Moving rating is: ' && lo_rating->get_n( ) TYPE 'I'.
CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
ENDTRY.

```

- For API details, see [GetItem](#) in *AWS SDK for SAP ABAP API reference*.

ListTables

The following code example shows how to use ListTables.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_dyn->listtables( ).
    " You can loop over the oo_result to get table properties like this.
    LOOP AT oo_result->get_tablenames( ) INTO DATA(lo_table_name).
        DATA(lv_tablename) = lo_table_name->get_value( ).
    ENDLOOP.
    DATA(lv_tablecount) = lines( oo_result->get_tablenames( ) ).
    MESSAGE 'Found ' && lv_tablecount && ' tables' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [ListTables](#) in *AWS SDK for SAP ABAP API reference*.

PutItem

The following code example shows how to use PutItem.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  DATA(lo_resp) = lo_dyn->putitem(  
    iv_tablename = iv_table_name  
    it_item      = it_item ).  
  MESSAGE '1 row inserted into DynamoDB Table' && iv_table_name TYPE 'I'.  
CATCH /aws1/cx_dyncondalcheckfaile00.  
  MESSAGE 'A condition specified in the operation could not be evaluated.'  
TYPE 'E'.  
CATCH /aws1/cx_dynresourcenotfoundex.  
  MESSAGE 'The table or index does not exist' TYPE 'E'.  
CATCH /aws1/cx_dyntransactconflictex.  
  MESSAGE 'Another transaction is using the item' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutItem](#) in *AWS SDK for SAP ABAP API reference*.

Query

The following code example shows how to use Query.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  " Query movies for a given year .
  DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
  ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_year }| ) ) ).
  DATA(lt_key_conditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
  ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
  key = 'year'
  value = NEW /aws1/cl_dyncondition(
  it_attributevaluelist = lt_attributelist
  iv_comparisonoperator = |EQ|
  ) ) ) ).
  oo_result = lo_dyn->query(
  iv_tablename = iv_table_name
  it_keyconditions = lt_key_conditions ).
  DATA(lt_items) = oo_result->get_items( ).
  "You can loop over the results to get item attributes.
  LOOP AT lt_items INTO DATA(lt_item).
  DATA(lo_title) = lt_item[ key = 'title' ]-value.
  DATA(lo_year) = lt_item[ key = 'year' ]-value.
  ENDLOOP.
  DATA(lv_count) = oo_result->get_count( ).
  MESSAGE 'Item count is: ' && lv_count TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
  MESSAGE 'The table or index does not exist' TYPE 'E'.
  ENDRTRY.

```

- For API details, see [Query](#) in *AWS SDK for SAP ABAP API reference*.

Scan

The following code example shows how to use Scan.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    " Scan movies for rating greater than or equal to the rating specified
    DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
    ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_rating }| ) ) ).
    DATA(lt_filter_conditions) = VALUE /aws1/
cl_dyncondition=>tt_filterconditionmap(
    ( VALUE /aws1/cl_dyncondition=>ts_filterconditionmap_maprow(
    key = 'rating'
    value = NEW /aws1/cl_dyncondition(
    it_attributevaluelist = lt_attributelist
    iv_comparisonoperator = |GE|
    ) ) ) ).
    oo_scan_result = lo_dyn->scan( iv_tablename = iv_table_name
    it_scanfilter = lt_filter_conditions ).
    DATA(lt_items) = oo_scan_result->get_items( ).
    LOOP AT lt_items INTO DATA(lo_item).
    " You can loop over to get individual attributes.
    DATA(lo_title) = lo_item[ key = 'title' ]-value.
    DATA(lo_year) = lo_item[ key = 'year' ]-value.
    ENDLOOP.
    DATA(lv_count) = oo_scan_result->get_count( ).
    MESSAGE 'Found ' && lv_count && ' items' TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
ENDTRY.

```

- For API details, see [Scan](#) in *AWS SDK for SAP ABAP API reference*.

UpdateItem

The following code example shows how to use UpdateItem.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_output = lo_dyn->updateitem(  
        iv_tablename      = iv_table_name  
        it_key            = it_item_key  
        it_attributeupdates = it_attribute_updates ).  
    MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.  
CATCH /aws1/cx_dyncondalcheckfaile00.  
    MESSAGE 'A condition specified in the operation could not be evaluated.'  
TYPE 'E'.  
CATCH /aws1/cx_dynresourcenotfoundex.  
    MESSAGE 'The table or index does not exist' TYPE 'E'.  
CATCH /aws1/cx_dyntransactconflictex.  
    MESSAGE 'Another transaction is using the item' TYPE 'E'.  
ENDTRY.
```

- For API details, see [UpdateItem](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.

- Scan for movies that were released in a range of years.
- Delete a movie from the table, then delete the table.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
" Create an Amazon Dynamo DB table.

TRY.
  DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
  DATA(lo_dyn) = /aws1/cl_dyn_factory=>create( lo_session ).
  DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
    ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                          iv_keytype = 'HASH' ) )
    ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                          iv_keytype = 'RANGE' ) ) ).
  DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
    ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                     iv_attributetype = 'N' ) )
    ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                     iv_attributetype = 'S' ) ) ).

  " Adjust read/write capacities as desired.
  DATA(lo_dynprovthroughput) = NEW /aws1/cl_dynprovthroughput(
    iv_readcapacityunits = 5
    iv_writecapacityunits = 5 ).
  DATA(oo_result) = lo_dyn->createtable(
    it_keyschema = lt_keyschema
    iv_tablename = iv_table_name
    it_attributedefinitions = lt_attributedefinitions
    io_provisionedthroughput = lo_dynprovthroughput ).
  " Table creation can take some time. Wait till table exists before
  returning.
  lo_dyn->get_waiter( )->tableexists(
    iv_max_wait_time = 200
```

```

        iv_tablename      = iv_table_name ).
    MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
    " It throws exception if the table already exists.
    CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
{ lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

    " Describe table
    TRY.
        DATA(lo_table) = lo_dyn->describetable( iv_tablename = iv_table_name ).
        DATA(lv_tablename) = lo_table->get_table( )->ask_tablename( ).
        MESSAGE 'The table name is ' && lv_tablename TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table does not exist' TYPE 'E'.
    ENDTRY.

    " Put items into the table.
    TRY.
        DATA(lo_resp_putitem) = lo_dyn->putitem(
            iv_tablename = iv_table_name
            it_item       = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
                key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
'Jaws' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
                key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1975' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
                key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.5' }| ) ) )
            ) ).
        lo_resp_putitem = lo_dyn->putitem(
            iv_tablename = iv_table_name
            it_item       = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
                key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s = 'Star
Wars' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
                key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1978' }| ) ) )

```

```

        ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
          key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '8.1' }| ) ) ) )
      ) ).
    lo_resp_putitem = lo_dyn->putitem(
      iv_tablename = iv_table_name
      it_item       = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
      ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
        key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
'Speed' ) ) ) )
      ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
        key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1994' }| ) ) ) )
      ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
        key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.9' }| ) ) ) )
      ) ).
    " TYPE REF TO ZCL_AWS1_dyn_PUT_ITEM_OUTPUT
    MESSAGE '3 rows inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
    CATCH /aws1/cx_dyncondalcheckfaile00.
    MESSAGE 'A condition specified in the operation could not be evaluated.'
    TYPE 'E'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
    CATCH /aws1/cx_dyntransactconflictex.
    MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Get item from table.
    TRY.
      DATA(lo_resp_getitem) = lo_dyn->getitem(
        iv_tablename           = iv_table_name
        it_key                 = VALUE /aws1/cl_dynattributevalue=>tt_key(
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
'Jaws' ) ) ) )
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
'1975' ) ) ) )
          ) ).
      DATA(lt_attr) = lo_resp_getitem->get_item( ).
      DATA(lo_title) = lt_attr[ key = 'title' ]-value.
      DATA(lo_year) = lt_attr[ key = 'year' ]-value.

```

```

    DATA(lo_rating) = lt_attr[ key = 'year' ]-value.
    MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
    MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
    MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
  ENDRY.

" Query item from table.
TRY.
  DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributelist(
    ( NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) ).
  DATA(lt_keyconditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
    ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
      key = 'year'
      value = NEW /aws1/cl_dyncondition(
        it_attributelist = lt_attributelist
        iv_comparisonoperator = |EQ|
      ) ) ) ).
  DATA(lo_query_result) = lo_dyn->query(
    iv_tablename = iv_table_name
    it_keyconditions = lt_keyconditions ).
  DATA(lt_items) = lo_query_result->get_items( ).
  READ TABLE lo_query_result->get_items( ) INTO DATA(lt_item) INDEX 1.
  lo_title = lt_item[ key = 'title' ]-value.
  lo_year = lt_item[ key = 'year' ]-value.
  lo_rating = lt_item[ key = 'rating' ]-value.
  MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
  MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
  MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
  MESSAGE 'The table or index does not exist' TYPE 'E'.
  ENDRY.

" Scan items from table.
TRY.
  DATA(lo_scan_result) = lo_dyn->scan( iv_tablename = iv_table_name ).
  lt_items = lo_scan_result->get_items( ).
  " Read the first item and display the attributes.
  READ TABLE lo_query_result->get_items( ) INTO lt_item INDEX 1.
  lo_title = lt_item[ key = 'title' ]-value.
  lo_year = lt_item[ key = 'year' ]-value.
  lo_rating = lt_item[ key = 'rating' ]-value.

```

```

    MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
    MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
    MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDRY.

" Update items from table.
TRY.
    DATA(lt_attributeupdates) = VALUE /aws1/
cl_dynattrvalueupdate=>tt_attributeupdates(
    ( VALUE /aws1/cl_dynattrvalueupdate=>ts_attributeupdates_maprow(
    key = 'rating' value = NEW /aws1/cl_dynattrvalueupdate(
    io_value = NEW /aws1/cl_dynattributevalue( iv_n = '7.6' )
    iv_action = |PUT| ) ) ) ).
    DATA(lt_key) = VALUE /aws1/cl_dynattributevalue=>tt_key(
    ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
    key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) )
    ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
    key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
'1980' ) ) ) ).
    DATA(lo_resp) = lo_dyn->updateitem(
    iv_tablename      = iv_table_name
    it_key            = lt_key
    it_attributeupdates = lt_attributeupdates ).
    MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
    CATCH /aws1/cx_dyncondalcheckfaile00.
    MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
    CATCH /aws1/cx_dyntransactconflictex.
    MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDRY.

" Delete table.
TRY.
    lo_dyn->deletetable( iv_tablename = iv_table_name ).
    lo_dyn->get_waiter( )->tablenotexists(
    iv_max_wait_time = 200
    iv_tablename     = iv_table_name ).
    MESSAGE 'DynamoDB Table deleted.' TYPE 'I'.
    CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.

```

```
CATCH /aws1/cx_dynresourceinuseex.  
  MESSAGE 'The table cannot be deleted as it is in use' TYPE 'E'.  
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Amazon EC2 examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon EC2.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

AllocateAddress

The following code example shows how to use AllocateAddress.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).    " oo_result is  
returned for testing purposes. "  
    MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [AllocateAddress](#) in *AWS SDK for SAP ABAP API reference*.

AssociateAddress

The following code example shows how to use AssociateAddress.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
```



```

        oo_result = lo_ec2->associateaddress(
            iv_allocationid = iv_allocation_id
            iv_instanceid = iv_instance_id
        ).
        MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [AssociateAddress](#) in *AWS SDK for SAP ABAP API reference*.

CreateKeyPair

The following code example shows how to use CreateKeyPair.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
        oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
        MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [CreateKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

CreateSecurityGroup

The following code example shows how to use CreateSecurityGroup.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->createsecuritygroup(
        iv_description = 'Security group example'
        iv_groupname = iv_security_group_name
        iv_vpcid = iv_vpc_id
    ).
    MESSAGE 'Security group created.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

DeleteKeyPair

The following code example shows how to use DeleteKeyPair.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_ec2->deletekeypair( iv_keyname = iv_key_name ).  
  MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.  
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
  MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

DeleteSecurityGroup

The following code example shows how to use DeleteSecurityGroup.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).  
  MESSAGE 'Security group deleted.' TYPE 'I'.  
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
  MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

DescribeAddresses

The following code example shows how to use DescribeAddresses.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describeaddresses( ) .
oo_result is returned for testing purposes. "
    DATA(lt_addresses) = oo_result->get_addresses( ).
    MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeAddresses](#) in *AWS SDK for SAP ABAP API reference*.

DescribeAvailabilityZones

The following code example shows how to use DescribeAvailabilityZones.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describeavailabilityzones( ) .
oo_result is returned for testing purposes. "
    DATA(lt_zones) = oo_result->get_availabilityzones( ).
    MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.
```

```

    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
      DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [DescribeAvailabilityZones](#) in *AWS SDK for SAP ABAP API reference*.

DescribeInstances

The following code example shows how to use DescribeInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
      oo_result = lo_ec2->describeinstances( ) .
      oo_result is returned for testing purposes. "

      " Retrieving details of EC2 instances. "
      DATA: lv_instance_id    TYPE /aws1/ec2string,
            lv_status          TYPE /aws1/ec2instancestatename,
            lv_instance_type   TYPE /aws1/ec2instancetype,
            lv_image_id        TYPE /aws1/ec2string.
      LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
        LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
          lv_instance_id = lo_instance->get_instanceid( ).
          lv_status = lo_instance->get_state( )->get_name( ).
          lv_instance_type = lo_instance->get_instancetype( ).
          lv_image_id = lo_instance->get_imageid( ).
        ENDLLOOP.
      ENDLLOOP.
      MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.

```

```

    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
      DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [DescribeInstances](#) in *AWS SDK for SAP ABAP API reference*.

DescribeKeyPairs

The following code example shows how to use DescribeKeyPairs.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
      oo_result = lo_ec2->describekeypairs( ) .
      " oo_result
is returned for testing purposes. "
      DATA(lt_key_pairs) = oo_result->get_keypairs( ).
      MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
      ENDTRY.

```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for SAP ABAP API reference*.

DescribeRegions

The following code example shows how to use DescribeRegions.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describeregions( ) .                " oo_result
is returned for testing purposes. "
    DATA(lt_regions) = oo_result->get_regions( ).
    MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeRegions](#) in *AWS SDK for SAP ABAP API reference*.

DescribeSecurityGroups

The following code example shows how to use DescribeSecurityGroups.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    DATA lt_group_ids TYPE /aws1/cl_ec2groupidstrlist_w=>tt_groupidstringlist.
    APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
lt_group_ids.
```

```

    oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
    " oo_result is returned for testing purposes. "
    DATA(lt_security_groups) = oo_result->get_securitygroups( ).
    MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for SAP ABAP API reference*.

MonitorInstances

The following code example shows how to use MonitorInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to monitor
the instance without actually making the request. "
        lo_ec2->monitorinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_true
        ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
required permissions to monitor this instance. "

```



```

    IF lo_exception->av_err_code = 'DryRunOperation'.
      MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
      " DryRun is set to false to enable detailed monitoring. "
      lo_ec2->monitorinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_false
      ).
      MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
      " If the error code returned is `UnauthorizedOperation`, then you don't
      have the required permissions to monitor this instance. "
      ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
        MESSAGE 'Dry run to enable detailed monitoring failed. User does not have
        the permissions to monitor the instance.' TYPE 'E'.
      ELSE.
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
        >av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
      ENDIF.
    ENDTRY.
  ENDTRY.

```

- For API details, see [MonitorInstances](#) in *AWS SDK for SAP ABAP API reference*.

RebootInstances

The following code example shows how to use RebootInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    DATA lt_instance_ids TYPE /aws1/
    cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
    lt_instance_ids.

    "Perform dry run"
    TRY.

```

```

    " DryRun is set to true. This checks for the required permissions to reboot
    the instance without actually making the request. "
    lo_ec2->rebootinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
    required permissions to reboot this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
        " DryRun is set to false to make a reboot request. "
        lo_ec2->rebootinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Instance rebooted.' TYPE 'I'.
        " If the error code returned is `UnauthorizedOperation`, then you don't
        have the required permissions to reboot this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to reboot instance failed. User does not have permissions
            to reboot the instance.' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
            >av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.

```

- For API details, see [RebootInstances](#) in *AWS SDK for SAP ABAP API reference*.

ReleaseAddress

The following code example shows how to use ReleaseAddress.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
  MESSAGE 'Elastic IP address released.' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- For API details, see [ReleaseAddress](#) in *AWS SDK for SAP ABAP API reference*.

RunInstances

The following code example shows how to use RunInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

" Create tags for resource created during instance launch. "
DATA lt_tag specifications TYPE /aws1/
cl_ec2tag specifications=>tt_tag specifications list.
DATA ls_tag specifications LIKE LINE OF lt_tag specifications.
ls_tag specifications = NEW /aws1/cl_ec2tag specification(
  iv_resource type = 'instance'
  it_tags = VALUE /aws1/cl_ec2tag=>tt_tag list(
    ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
  )
).
APPEND ls_tag specifications TO lt_tag specifications.

TRY.
  " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
  oo_result = lo_ec2->runinstances( " oo_result is
returned for testing purposes. "

```

```

        iv_imageid = iv_ami_id
        iv_instancetype = 't2.micro'
        iv_maxcount = 1
        iv_mincount = 1
        it_tagspecifications = lt_tagspecifications
        iv_subnetid = iv_subnet_id
    ).
    MESSAGE 'EC2 instance created.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [RunInstances](#) in *AWS SDK for SAP ABAP API reference*.

StartInstances

The following code example shows how to use StartInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to start
the instance without actually making the request. "
        lo_ec2->startinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_true

```

```

    ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
    required permissions to start this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
        " DryRun is set to false to start instance. "
        oo_result = lo_ec2->startinstances(          " oo_result is returned for
testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
        " If the error code returned is `UnauthorizedOperation`, then you don't
        have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to start instance failed. User does not have permissions
to start the instance.' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.

```

- For API details, see [StartInstances](#) in *AWS SDK for SAP ABAP API reference*.

StopInstances

The following code example shows how to use StopInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
  APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
  lt_instance_ids.

  "Perform dry run"
  TRY.
    " DryRun is set to true. This checks for the required permissions to stop
    the instance without actually making the request. "
    lo_ec2->stopinstances(
      it_instanceids = lt_instance_ids
      iv_dryrun = abap_true
    ).
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
    required permissions to stop this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
      MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
      " DryRun is set to false to stop instance. "
      oo_result = lo_ec2->stopinstances(          " oo_result is returned for
testing purposes. "
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_false
      ).
      MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
      " If the error code returned is `UnauthorizedOperation`, then you don't
      have the required permissions to stop this instance. "
      ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
        MESSAGE 'Dry run to stop instance failed. User does not have permissions
to stop the instance.' TYPE 'E'.
      ELSE.
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
      ENDIF.
    ENDTRY.
  ENDTRY.

```

- For API details, see [StopInstances](#) in *AWS SDK for SAP ABAP API reference*.

Kinesis examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Kinesis.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateStream

The following code example shows how to use CreateStream.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_kns->createstream(  
        iv_streamname = iv_stream_name  
        iv_shardcount = iv_shard_count  
    ).
```

```
    MESSAGE 'Stream created.' TYPE 'I'.
  CATCH /aws1/cx_knsinvalidargumentex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
  CATCH /aws1/cx_knslimitexceeddex .
    MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
  CATCH /aws1/cx_knsresourceinuseex .
    MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
  ENDTRY.
```

- For API details, see [CreateStream](#) in *AWS SDK for SAP ABAP API reference*.

DeleteStream

The following code example shows how to use DeleteStream.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
  TRY.
    lo_kns->deletestream(
      iv_streamname = iv_stream_name
    ).
    MESSAGE 'Stream deleted.' TYPE 'I'.
  CATCH /aws1/cx_knslimitexceeddex .
    MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
  CATCH /aws1/cx_knsresourceinuseex .
    MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
  ENDTRY.
```

- For API details, see [DeleteStream](#) in *AWS SDK for SAP ABAP API reference*.

DescribeStream

The following code example shows how to use DescribeStream.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->describestream(  
        iv_streamname = iv_stream_name  
    ).  
    DATA(lt_stream_description) = oo_result->get_streamdescription( ).  
    MESSAGE 'Streams retrieved.' TYPE 'I'.  
CATCH /aws1/cx_knslimitexceedex .  
    MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
CATCH /aws1/cx_knsresourcenotfoundex .  
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DescribeStream](#) in *AWS SDK for SAP ABAP API reference*.

GetRecords

The following code example shows how to use GetRecords.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_kns->getrecords(           " oo_result is returned for
testing purposes. "
        iv_sharditerator = iv_shard_iterator
    ).
    DATA(lt_records) = oo_result->get_records( ).
    MESSAGE 'Record retrieved.' TYPE 'I'.
CATCH /aws1/cx_knsexpirediteratorex .
    MESSAGE 'Iterator expired.' TYPE 'E'.
CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled.' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state.' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found.' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
    MESSAGE 'KMS key option is required.' TYPE 'E'.
CATCH /aws1/cx_knskmsstrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexcdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.

```

- For API details, see [GetRecords](#) in *AWS SDK for SAP ABAP API reference*.

ListStreams

The following code example shows how to use ListStreams.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->liststreams(           " oo_result is returned for testing  
purposes. "  
        "Set Limit to specify that a maximum of streams should be returned."  
        iv_limit = iv_limit  
    ).  
    DATA(lt_streams) = oo_result->get_streamnames( ).  
    MESSAGE 'Streams listed.' TYPE 'I'.  
CATCH /aws1/cx_knslimitexceededException .  
    MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListStreams](#) in *AWS SDK for SAP ABAP API reference*.

PutRecord

The following code example shows how to use PutRecord.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->putrecord(           " oo_result is returned for  
testing purposes. "
```

```

        iv_streamname = iv_stream_name
        iv_data        = iv_data
        iv_partitionkey = iv_partition_key
    ).
    MESSAGE 'Record created.' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled.' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found.' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
    MESSAGE 'KMS key option is required.' TYPE 'E'.
CATCH /aws1/cx_knskmsstrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexcdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutRecord](#) in *AWS SDK for SAP ABAP API reference*.

RegisterStreamConsumer

The following code example shows how to use RegisterStreamConsumer.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->registerstreamconsumer(      " oo_result is returned
for testing purposes. "
        iv_streamarn = iv_stream_arn
        iv_consumername = iv_consumer_name
    ).
    MESSAGE 'Stream consumer registered.' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotinuse.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [RegisterStreamConsumer](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with data streams

The following code example shows how to:

- Create a stream and put a record in it.
- Create a shard iterator.
- Read the record, then clean up resources.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

"Create stream."
TRY.
    lo_kns->createstream(
        iv_streamname = iv_stream_name
        iv_shardcount = iv_shard_count
    ).
    MESSAGE 'Stream created.' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knslimitexceededex .
    MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
CATCH /aws1/cx_knsresourceinuseex .
    MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.

"Wait for stream to becomes active."
lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
    IF sy-index = 30.
        EXIT.          "maximum 5 minutes"
    ENDIF.
    WAIT UP TO 10 SECONDS.
    lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
    lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
ENDWHILE.

"Create record."
TRY.
    lo_record_result = lo_kns->putrecord(
        iv_streamname = iv_stream_name
        iv_data        = iv_data
        iv_partitionkey = iv_partition_key
    ).
    MESSAGE 'Record created.' TYPE 'I'.
```

```

CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled.' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found.' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
    MESSAGE 'KMS key option is required.' TYPE 'E'.
CATCH /aws1/cx_knskmsstrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexcdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.

"Create a shard iterator in order to read the record."
TRY.
    lo_sharditerator = lo_kns->getsharditerator(
        iv_shardid = lo_record_result->get_shardid( )
        iv_sharditeratortype = iv_sharditeratortype
        iv_streamname = iv_stream_name
    ).
    MESSAGE 'Shard iterator created.' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexcdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.

"Read the record."
TRY.
    oo_result = lo_kns->getrecords(
        " oo_result is returned
for testing purposes. "

```

```

        iv_sharditerator = lo_sharditerator->get_sharditerator( )
    ).
    MESSAGE 'Shard iterator created.' TYPE 'I'.
    CATCH /aws1/cx_knsexpirediteratorex .
        MESSAGE 'Iterator expired.' TYPE 'E'.
    CATCH /aws1/cx_knsinvalidargumentex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knskmsaccessdeniedex .
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
    CATCH /aws1/cx_knskmsdisabledex .
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
    CATCH /aws1/cx_knskmsinvalidstateex .
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
    CATCH /aws1/cx_knskmsnotfoundex .
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
    CATCH /aws1/cx_knskmsoptinrequired .
        MESSAGE 'KMS key option is required.' TYPE 'E'.
    CATCH /aws1/cx_knskmsstrottlingex .
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
    CATCH /aws1/cx_knsprovthruputexcdex .
        MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
    CATCH /aws1/cx_knsresourcenotfoundex .
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Delete stream."
    TRY.
        lo_kns->deletestream(
            iv_streamname = iv_stream_name
        ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
    CATCH /aws1/cx_knslimitexceededex .
        MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
    CATCH /aws1/cx_knsresourceinuseex .
        MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
    ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [CreateStream](#)
- [DeleteStream](#)
- [GetRecords](#)
- [GetShardIterator](#)
- [PutRecord](#)

Lambda examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Lambda.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateFunction

The following code example shows how to use CreateFunction.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  lo_lmd->createfunction(
    iv_functionname = iv_function_name
    iv_runtime = `python3.9`
    iv_role = iv_role_arn
    iv_handler = iv_handler
    io_code = io_zip_file
    iv_description = 'AWS Lambda code example'
  ).
  MESSAGE 'Lambda function created.' TYPE 'I'.
CATCH /aws1/cx_lmdcodesigningcfgno00.
  MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdcodestorageexcdex.
  MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
CATCH /aws1/cx_lmdcodeverification00.
  MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
CATCH /aws1/cx_lmdinvalidcodesigex.
  MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
CATCH /aws1/cx_lmdinvparamvalueex.
  MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
  MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
  MESSAGE 'The requested resource does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
  MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
  MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.

```

- For API details, see [CreateFunction](#) in *AWS SDK for SAP ABAP API reference*.

DeleteFunction

The following code example shows how to use DeleteFunction.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_lmd->deletefunction( iv_functionname = iv_function_name ).  
  MESSAGE 'Lambda function deleted.' TYPE 'I'.  
CATCH /aws1/cx_lmdinvparamvalueex.  
  MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.  
CATCH /aws1/cx_lmdresourceconflictex.  
  MESSAGE 'Resource already exists or another operation is in progress.' TYPE  
'E'.  
CATCH /aws1/cx_lmdresourcenotfoundex.  
  MESSAGE 'The requested resource does not exist.' TYPE 'E'.  
CATCH /aws1/cx_lmdserviceexception.  
  MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
TYPE 'E'.  
CATCH /aws1/cx_lmdtoomanyrequestsex.  
  MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteFunction](#) in *AWS SDK for SAP ABAP API reference*.

GetFunction

The following code example shows how to use GetFunction.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).
" oo_result is returned for testing purposes. "
    MESSAGE 'Lambda function information retrieved.' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
    CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.

```

- For API details, see [GetFunction](#) in *AWS SDK for SAP ABAP API reference*.

Invoke

The following code example shows how to use Invoke.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
        `{` &&
        ` "action": "increment",` &&
        ` "number": 10` &&
        `}`
    ).
    oo_result = lo_lmd->invoke(
testing purposes. "
        iv_functionname = iv_function_name
        iv_payload = lv_json
    ).
    MESSAGE 'Lambda function invoked.' TYPE 'I'.

```

```

CATCH /aws1/cx_lmdinvparamvalueex.
  MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
CATCH /aws1/cx_lmdinvrequestcontex.
  MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
CATCH /aws1/cx_lmdinvalidzipfileex.
  MESSAGE 'The deployment package could not be unzipped.' TYPE 'E'.
CATCH /aws1/cx_lmdrequesttoolargeex.
  MESSAGE 'Invoke request body JSON input limit was exceeded by the request
payload.' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
  MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
  MESSAGE 'The requested resource does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
  MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
  MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
CATCH /aws1/cx_lmdunsupportedmediatyp00.
  MESSAGE 'Invoke request body does not have JSON as its content type.' TYPE
'E'.
ENDTRY.

```

- For API details, see [Invoke](#) in *AWS SDK for SAP ABAP API reference*.

ListFunctions

The following code example shows how to use ListFunctions.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  oo_result = lo_lmd->listfunctions( ).      " oo_result is returned for
testing purposes. "

```

```

    DATA(lt_functions) = oo_result->get_functions( ).
    MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
    MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_lmdserviceexception.
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
    CATCH /aws1/cx_lmdtoomanyrequestsex.
    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.

```

- For API details, see [ListFunctions](#) in *AWS SDK for SAP ABAP API reference*.

UpdateFunctionCode

The following code example shows how to use UpdateFunctionCode.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_lmd->updatefunctioncode(      " oo_result is returned for
testing purposes. "
        iv_functionname = iv_function_name
        iv_zipfile = io_zip_file
    ).

    MESSAGE 'Lambda function code updated.' TYPE 'I'.
    CATCH /aws1/cx_lmdcodesigningcfgno00.
    MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
    CATCH /aws1/cx_lmdcodestorageexc00.
    MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
    CATCH /aws1/cx_lmdcodeverification00.
    MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.

```

```

CATCH /aws1/cx_lmdinvalidcodesigex.
  MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
CATCH /aws1/cx_lmdinvparamvalueex.
  MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
  MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
  MESSAGE 'The requested resource does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
  MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
  MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.

```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for SAP ABAP API reference*.

UpdateFunctionConfiguration

The following code example shows how to use UpdateFunctionConfiguration.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  oo_result = lo_lmd->updatefunctionconfiguration(      " oo_result is returned
for testing purposes. "
    iv_functionname = iv_function_name
    iv_runtime = iv_runtime
    iv_description = 'Updated Lambda function'
    iv_memorysize = iv_memory_size
  ).
  MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.

```

```
CATCH /aws1/cx_lmdcodesigningcfgno00.
  MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdcodeverification00.
  MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
CATCH /aws1/cx_lmdinvalidcodesigex.
  MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
CATCH /aws1/cx_lmdinvparamvalueex.
  MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
  MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
  MESSAGE 'The requested resource does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
  MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
  MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.
- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see [Create a Lambda function with the console](#).

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
  "Create an AWS Identity and Access Management (IAM) role that grants AWS
  Lambda permission to write to logs."
  DATA(lv_policy_document) = `{` &&
    ` "Version": "2012-10-17",` &&
    ` "Statement": [` &&
      `{` &&
        ` "Effect": "Allow",` &&
        ` "Action": [` &&
          ` "sts:AssumeRole"` &&
        ` ],` &&
        ` "Principal": {` &&
          ` "Service": [` &&
            ` "lambda.amazonaws.com"` &&
          ` ]` &&
        ` }` &&
      ` }` &&
    ` ]` &&
  `}`.

TRY.
  DATA(lo_create_role_output) = lo_iam->createrole(
    iv_rolename = iv_role_name
    iv_assumerolepolicydocument = lv_policy_document
    iv_description = 'Grant lambda permission to write to logs'
  ).
  MESSAGE 'IAM role created.' TYPE 'I'.
  WAIT UP TO 10 SECONDS.           " Make sure that the IAM role is ready
for use. "
  CATCH /aws1/cx_iamentityalrddyex.
    MESSAGE 'IAM role already exists.' TYPE 'E'.
  CATCH /aws1/cx_iainvalidinputex.
    MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.

```

```
        MESSAGE 'Policy document in the request is malformed.' TYPE 'E'.
    ENDRY.

    TRY.
        lo_iam->attachrolepolicy(
            iv_rolename = iv_role_name
            iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole'
        ).
        MESSAGE 'Attached policy to the IAM role.' TYPE 'I'.
        CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
        CATCH /aws1/cx_iamplynnotattachableex.
        MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
        CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
    ENDRY.

    " Create a Lambda function and upload handler code. "
    " Lambda function performs 'increment' action on a number. "
    TRY.
        lo_lmd->createfunction(
            iv_functionname = iv_function_name
            iv_runtime = `python3.9`
            iv_role = lo_create_role_output->get_role( )->get_arn( )
            iv_handler = iv_handler
            io_code = io_initial_zip_file
            iv_description = 'AWS Lambda code example'
        ).
        MESSAGE 'Lambda function created.' TYPE 'I'.
        CATCH /aws1/cx_lmdcodestorageexcdex.
        MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
        CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
    ENDRY.

    " Verify the function is in Active state "
```

```

    WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
        IF sy-index = 10.
            EXIT.                " Maximum 10 seconds. "
        ENDIF.
        WAIT UP TO 1 SECONDS.
    ENDWHILE.

    "Invoke the function with a single parameter and get results."
    TRY.
        DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
            `{` &&
            ` "action": "increment",` &&
            ` "number": 10` &&
            `}`
        ).
        DATA(lo_initial_invoke_output) = lo_lmd->invoke(
            iv_functionname = iv_function_name
            iv_payload = lv_json
        ).
        ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
        " ov_initial_invoke_payload is returned for testing purposes. "
        DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
        CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT XML
lo_writer_json.
        DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
        MESSAGE 'Lambda function invoked.' TYPE 'I'.
        CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        CATCH /aws1/cx_lmdunsuppmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
TYPE 'E'.
    ENDTRY.

    " Update the function code and configure its Lambda environment with an
environment variable. "
    " Lambda function is updated to perform 'decrement' action also. "
    TRY.

```

```

        lo_lmd->updatefunctioncode(
            iv_functionname = iv_function_name
            iv_zipfile = io_updated_zip_file
        ).
    WAIT UP TO 10 SECONDS.          " Make sure that the update is
completed. "
    MESSAGE 'Lambda function code updated.' TYPE 'I'.
    CATCH /aws1/cx_lmdcodestorageexcex.
        MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
    ENDTRY.

    TRY.
        DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
        DATA ls_variable LIKE LINE OF lt_variables.
        ls_variable-key = 'LOG_LEVEL'.
        ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
'info' ).
        INSERT ls_variable INTO TABLE lt_variables.

        lo_lmd->updatefunctionconfiguration(
            iv_functionname = iv_function_name
            io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
lt_variables )
        ).
    WAIT UP TO 10 SECONDS.          " Make sure that the update is
completed. "
    MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
    ENDTRY.

    "Invoke the function with new parameters and get results. Display the
execution log that's returned from the invocation."
    TRY.

```

```

lv_json = /aws1/cl_rt_util=>string_to_xstring(
  `{`  &&
  ` "action": "decrement",`  &&
  ` "number": 10`  &&
  `}`
).
DATA(lo_updated_invoke_output) = lo_lmd->invoke(
  iv_functionname = iv_function_name
  iv_payload = lv_json
).
ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
" ov_updated_invoke_payload is returned for testing purposes. "
lo_writer_json = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT XML
lo_writer_json.
lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
MESSAGE 'Lambda function invoked.' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
CATCH /aws1/cx_lmdinvrequestcontex.
MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist.' TYPE 'E'.
CATCH /aws1/cx_lmdunsuppmediatyp00.
MESSAGE 'Invoke request body does not have JSON as its content type.'
TYPE 'E'.
ENDTRY.

" List the functions for your account. "
TRY.
DATA(lo_list_output) = lo_lmd->listfunctions( ).
DATA(lt_functions) = lo_list_output->get_functions( ).
MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
ENDTRY.

" Delete the Lambda function. "
TRY.
lo_lmd->deletefunction( iv_functionname = iv_function_name ).
MESSAGE 'Lambda function deleted.' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.

```

```

        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
    ENDTRY.

" Detach role policy. "
TRY.
    lo_iam->detachrolepolicy(
        iv_rolename = iv_role_name
        iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole'
    ).
    MESSAGE 'Detached policy from the IAM role.' TYPE 'I'.
    CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
    CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamplynottattachableex.
        MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
    CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
    ENDTRY.

" Delete the IAM role. "
TRY.
    lo_iam->deleterole( iv_rolename = iv_role_name ).
    MESSAGE 'IAM role deleted.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
    ENDTRY.

    CATCH /aws1/cx_rt_service_generic INTO lo_exception.
        DATA(lv_error) = lo_exception->get_longtext( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

Amazon S3 examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon S3.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CopyObject

The following code example shows how to use CopyObject.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_s3->copyobject(  
    iv_bucket = iv_dest_bucket  
    iv_key = iv_dest_object  
    iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|  
  ).  
  MESSAGE 'Object copied to another bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [CopyObject](#) in *AWS SDK for SAP ABAP API reference*.

CreateBucket

The following code example shows how to use CreateBucket.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_s3->createbucket(  
    iv_bucket = iv_bucket_name
```



```
    ).  
    MESSAGE 'S3 bucket created.' TYPE 'I'.  
  CATCH /aws1/cx_s3_bucketalrddyexists.  
    MESSAGE 'Bucket name already exists.' TYPE 'E'.  
  CATCH /aws1/cx_s3_bktalrddyownedbyyou.  
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.  
  ENDTRY.
```

- For API details, see [CreateBucket](#) in *AWS SDK for SAP ABAP API reference*.

DeleteBucket

The following code example shows how to use DeleteBucket.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
  TRY.  
  
    lo_s3->deletebucket(  
      iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.  
  CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
  ENDTRY.
```

- For API details, see [DeleteBucket](#) in *AWS SDK for SAP ABAP API reference*.

DeleteObject

The following code example shows how to use DeleteObject.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
  ).
  MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteObject](#) in *AWS SDK for SAP ABAP API reference*.

GetObject

The following code example shows how to use GetObject.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_s3->getobject(           " oo_result is returned for testing
purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
  ).
```

```
DATA(lv_object_data) = oo_result->get_body( ).
MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetObject](#) in *AWS SDK for SAP ABAP API reference*.

ListObjectsV2

The following code example shows how to use ListObjectsV2.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->listobjectsv2(           " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    ).
MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListObjectsV2](#) in *AWS SDK for SAP ABAP API reference*.

PutObject

The following code example shows how to use PutObject.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Get contents of file from application server."  
DATA lv_body TYPE xstring.  
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.  
READ DATASET iv_file_name INTO lv_body.  
CLOSE DATASET iv_file_name.  
  
"Upload/put an object to an S3 bucket."  
TRY.  
    lo_s3->putobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_file_name  
        iv_body = lv_body  
    ).  
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutObject](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with buckets and objects

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.

- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrddyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrddyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content
    ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
```

```
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
  DATA(lo_result) = lo_s3->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key
  ).
  DATA(lv_object_data) = lo_result->get_body( ).
  MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
  MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
  lo_s3->copyobject(
    iv_bucket = iv_bucket_name
    iv_key = |{ iv_copy_to_folder }/{ iv_key }|
    iv_copysource = |{ iv_bucket_name }/{ iv_key }|
  ).
  MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
  MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
TRY.
  DATA(lo_list) = lo_s3->listobjects(
    iv_bucket = iv_bucket_name
  ).
  MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
```

```
lv_object_key = lo_object->get_key( ).
CONCATENATE lv_object_key ' ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE'I'.

" Delete the objects in a bucket. "
TRY.
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key
  ).
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = |{ iv_copy_to_folder }/{ iv_key }|
  ).
  MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
  lo_s3->deletebucket(
    iv_bucket = iv_bucket_name
  ).
  MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

SageMaker examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with SageMaker.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateEndpoint

The following code example shows how to use CreateEndpoint.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_production_variants TYPE /aws1/  
cl_sgmproductionvariant=>tt_productionvariantlist.  
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.  
DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
```



```

"Create a production variant as an ABAP object."
"Identifies a model that you want to host and the resources chosen to deploy for
hosting it."
CREATE OBJECT lo_production_variants
  EXPORTING
    iv_variantname      = iv_variant_name
    iv_modelname        = iv_model_name
    iv_initialinstancecount = iv_initial_instance_count
    iv_instancetype     = iv_instance_type.

INSERT lo_production_variants INTO TABLE lt_production_variants.

"Create an endpoint configuration."
TRY.
  oo_ep_config_result = lo_sgm->createendpointconfig(
    iv_endpointconfigname = iv_endpoint_config_name
    it_productionvariants = lt_production_variants
  ).
  MESSAGE 'Endpoint configuration created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

"Create an endpoint."
TRY.
  oo_result = lo_sgm->createendpoint(      " oo_result is returned for testing
purposes. "
    iv_endpointconfigname = iv_endpoint_config_name
    iv_endpointname       = iv_endpoint_name
  ).
  MESSAGE 'Endpoint created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

```

- For API details, see [CreateEndpoint](#) in *AWS SDK for SAP ABAP API reference*.

CreateModel

The following code example shows how to use `CreateModel`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

"Create an ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
  EXPORTING
    iv_image          = iv_container_image
    iv_modeldataurl  = iv_model_data_url.

"Create an Amazon SageMaker model."
TRY.
  oo_result = lo_sgm->createmodel(          " oo_result is returned for testing
purposes. "
    iv_executionrolearn = iv_execution_role_arn
    iv_modelname       = iv_model_name
    io_primarycontainer = lo_primarycontainer
  ).
  MESSAGE 'Model created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateModel](#) in *AWS SDK for SAP ABAP API reference*.

CreateTrainingJob

The following code example shows how to use CreateTrainingJob.

SDK for SAP ABAP**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithm_spec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmresourceconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
```

"Create ABAP internal table for hyperparameters based on input variables."

"These hyperparameters are based on the Amazon SageMaker built-in algorithm, XGBoost."

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO
TABLE lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.  
INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE  
lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.  
INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE  
lt_hyperparameters.
```

```
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.  
INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE  
lt_hyperparameters.
```

"Create ABAP objects for training data sources."

```
CREATE OBJECT lo_trn_s3datasource  
EXPORTING  
    iv_s3datatype           = iv_trn_data_s3datatype  
    iv_s3datadistributiontype = iv_trn_data_s3datadistribution  
    iv_s3uri                = iv_trn_data_s3uri.
```

```
CREATE OBJECT lo_trn_datasource  
EXPORTING  
    io_s3datasource = lo_trn_s3datasource.
```

```
CREATE OBJECT lo_trn_channel  
EXPORTING  
    iv_channelname      = 'train'  
    io_datasource       = lo_trn_datasource  
    iv_compressiontype  = iv_trn_data_compressiontype  
    iv_contenttype      = iv_trn_data_contenttype.
```

```
INSERT lo_trn_channel INTO TABLE lt_input_data_config.
```

"Create ABAP objects for validation data sources."

```
CREATE OBJECT lo_val_s3datasource  
EXPORTING  
    iv_s3datatype           = iv_val_data_s3datatype  
    iv_s3datadistributiontype = iv_val_data_s3datadistribution  
    iv_s3uri                = iv_val_data_s3uri.
```

```
CREATE OBJECT lo_val_datasource  
EXPORTING  
    io_s3datasource = lo_val_s3datasource.
```

```
CREATE OBJECT lo_val_channel
```

```

EXPORTING
  iv_channelname      = 'validation'
  io_datasource       = lo_val_datasource
  iv_compressiontype = iv_val_data_compressiontype
  iv_contenttype      = iv_val_data_contenttype.

INSERT lo_val_channel INTO TABLE lt_input_data_config.

"Create an ABAP object for algorithm specification."
CREATE OBJECT lo_algorithm_specification
  EXPORTING
    iv_trainingimage      = iv_training_image
    iv_traininginputmode = iv_training_input_mode.

"Create an ABAP object for resource configuration."
CREATE OBJECT lo_resource_config
  EXPORTING
    iv_instancecount = iv_instance_count
    iv_instancetype  = iv_instance_type
    iv_volumesizeingb = iv_volume_sizeingb.

"Create an ABAP object for output data configuration."
CREATE OBJECT lo_output_data_config
  EXPORTING
    iv_s3outputpath = iv_s3_output_path.

"Create an ABAP object for stopping condition."
CREATE OBJECT lo_stopping_condition
  EXPORTING
    iv_maxruntimeinseconds = iv_max_runtime_in_seconds.

"Create a training job."
TRY.
  oo_result = lo_sgm->createtrainingjob( " oo_result is returned for
testing purposes. "
    iv_trainingjobname      = iv_training_job_name
    iv_rolearn              = iv_role_arn
    it_hyperparameters      = lt_hyperparameters
    it_inputdataconfig      = lt_input_data_config
    io_algorithmspecification = lo_algorithm_specification
    io_outputdataconfig     = lo_output_data_config
    io_resourceconfig       = lo_resource_config
    io_stoppingcondition    = lo_stopping_condition
  ).

```

```

    MESSAGE 'Training job created.' TYPE 'I'.
  CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
  CATCH /aws1/cx_sgmresourceNotFound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
  CATCH /aws1/cx_sgmresourceLimitExcd.
    MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
  ENDTRY.

```

- For API details, see [CreateTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

CreateTransformJob

The following code example shows how to use `CreateTransformJob`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
DATA lo_datasource TYPE REF TO /aws1/cl_sgmtransformdatasrc.
DATA lo_s3datasource TYPE REF TO /aws1/cl_sgmtransformS3datasrc.

```

"Create an ABAP object for an Amazon Simple Storage Service (Amazon S3) data source."

```

CREATE OBJECT lo_s3datasource
  EXPORTING
    iv_s3uri      = iv_tf_data_s3uri
    iv_s3datatype = iv_tf_data_s3datatype.

```

"Create an ABAP object for data source."

```

CREATE OBJECT lo_datasource
  EXPORTING

```

```

        io_s3datasource = lo_s3datasource.

"Create an ABAP object for transform data source."
CREATE OBJECT lo_transforminput
EXPORTING
    io_datasource      = lo_datasource
    iv_contenttype     = iv_tf_data_contenttype
    iv_compressiontype = iv_tf_data_compressiontype.

"Create an ABAP object for resource configuration."
CREATE OBJECT lo_transformresources
EXPORTING
    iv_instancecount = iv_instance_count
    iv_instancetype  = iv_instance_type.

"Create an ABAP object for output data configuration."
CREATE OBJECT lo_transformoutput
EXPORTING
    iv_s3outputpath = iv_s3_output_path.

"Create a transform job."
TRY.
    oo_result = lo_sgm->createtransformjob(      " oo_result is returned for
testing purposes. "
        iv_modelname = iv_tf_model_name
        iv_transformjobname = iv_tf_job_name
        io_transforminput = lo_transforminput
        io_transformoutput = lo_transformoutput
        io_transformresources = lo_transformresources
    ).
    MESSAGE 'Transform job created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourceNotFound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourceLimitExcd.
    MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

```

- For API details, see [CreateTransformJob](#) in *AWS SDK for SAP ABAP API reference*.

DeleteEndpoint

The following code example shows how to use DeleteEndpoint.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Delete an endpoint."
TRY.
    lo_sgm->deleteendpoint(
        iv_endpointname = iv_endpoint_name
    ).
    MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
    DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
{ lo_endpoint_exception->av_err_msg }|.
    MESSAGE lv_endpoint_error TYPE 'E'.
ENDTRY.

"Delete an endpoint configuration."
TRY.
    lo_sgm->deleteendpointconfig(
        iv_endpointconfigname = iv_endpoint_config_name
    ).
    MESSAGE 'Endpoint deleted.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
    DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
    MESSAGE lv_endpointconfig_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for SAP ABAP API reference*.

DeleteModel

The following code example shows how to use DeleteModel.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_sgm->deletemodel(
    iv_modelname = iv_model_name
  ).
  MESSAGE 'Model deleted.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteModel](#) in *AWS SDK for SAP ABAP API reference*.

DescribeTrainingJob

The following code example shows how to use DescribeTrainingJob.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_sgm->describetrainingjob( " oo_result is returned for
testing purposes. "
  iv_trainingjobname = iv_training_job_name
  ).
```

```

    MESSAGE 'Retrieved description of training job.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
  ENDTRY.

```

- For API details, see [DescribeTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

ListAlgorithms

The following code example shows how to use `ListAlgorithms`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

  TRY.
    oo_result = lo_sgm->listalgorithms(           " oo_result is returned for
testing purposes. "
    iv_namecontains = iv_name_contains
    ).
    MESSAGE 'Retrieved list of algorithms.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
  ENDTRY.

```

- For API details, see [ListAlgorithms](#) in *AWS SDK for SAP ABAP API reference*.

ListModels

The following code example shows how to use `ListModels`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sgm->listmodels(           " oo_result is returned for  
testing purposes. "  
        iv_namecontains = iv_name_contains  
    ).  
    MESSAGE 'Retrieved list of models.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListModels](#) in *AWS SDK for SAP ABAP API reference*.

ListNotebookInstances

The following code example shows how to use ListNotebookInstances.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sgm->listnotebookinstances(           " oo_result is returned  
for testing purposes. "  
        iv_namecontains = iv_name_contains
```

```

    ).
    MESSAGE 'Retrieved list of notebook instances.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [ListNotebookInstances](#) in *AWS SDK for SAP ABAP API reference*.

ListTrainingJobs

The following code example shows how to use ListTrainingJobs.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
        oo_result = lo_sgm->listtrainingjobs(      " oo_result is returned for
testing purposes. "
        iv_namecontains = iv_name_contains
        iv_maxresults = iv_max_results
    ).
    MESSAGE 'Retrieved list of training jobs.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- For API details, see [ListTrainingJobs](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with models and endpoints

The following code example shows how to:

- Start a training job and create a SageMaker model.
- Create an endpoint configuration.
- Create an endpoint, then clean up resources.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithm_spec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmresourceconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescrtrnjobrsp.
DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
DATA lv_model_data_url TYPE /aws1/sgmurl.

```

```
lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

"Create ABAP internal table for hyperparameters based on input variables."
"These hyperparameters are based on Amazon SageMaker built-in algorithm -
XGBoost"
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO
TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.
INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.
INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.
INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

"Create ABAP internal table for data based on input variables."
"Training data."
CREATE OBJECT lo_trn_s3datasource
EXPORTING
  iv_s3datatype           = iv_trn_data_s3datatype
  iv_s3datadistributiontype = iv_trn_data_s3datadistribution
  iv_s3uri                = iv_trn_data_s3uri.

CREATE OBJECT lo_trn_datasource EXPORTING io_s3datasource = lo_trn_s3datasource.
```

```
CREATE OBJECT lo_trn_channel
  EXPORTING
    iv_channelname      = 'train'
    io_datasource       = lo_trn_datasource
    iv_compressiontype  = iv_trn_data_compressiontype
    iv_contenttype      = iv_trn_data_contenttype.
INSERT lo_trn_channel INTO TABLE lt_input_data_config.

"Validation data."
CREATE OBJECT lo_val_s3datasource
  EXPORTING
    iv_s3datatype       = iv_val_data_s3datatype
    iv_s3datadistributiontype = iv_val_data_s3datadistribution
    iv_s3uri             = iv_val_data_s3uri.

CREATE OBJECT lo_val_datasource EXPORTING io_s3datasource = lo_val_s3datasource.

CREATE OBJECT lo_val_channel
  EXPORTING
    iv_channelname      = 'validation'
    io_datasource       = lo_val_datasource
    iv_compressiontype  = iv_val_data_compressiontype
    iv_contenttype      = iv_val_data_contenttype.
INSERT lo_val_channel INTO TABLE lt_input_data_config.

"Create an ABAP object for algorithm specification based on input variables."
CREATE OBJECT lo_algorithm_specification
  EXPORTING
    iv_trainingimage     = iv_training_image
    iv_traininginputmode = iv_training_input_mode.

"Create an ABAP object for resource configuration."
CREATE OBJECT lo_resource_config
  EXPORTING
    iv_instancecount    = iv_instance_count
    iv_instancetype     = iv_instance_type
    iv_volumesizeingb   = iv_volume_sizeingb.

"Create an ABAP object for output data configuration."
CREATE OBJECT lo_output_data_config EXPORTING iv_s3outputpath =
iv_s3_output_path.

"Create an ABAP object for stopping condition."
```

```
CREATE OBJECT lo_stopping_condition EXPORTING iv_maxruntimeinseconds =
iv_max_runtime_in_seconds.

TRY.
  lo_sgm->createtrainingjob(
    iv_trainingjobname      = iv_training_job_name
    iv_rolearn              = iv_role_arn
    it_hyperparameters      = lt_hyperparameters
    it_inputdataconfig      = lt_input_data_config
    io_algorithmspecification = lo_algorithm_specification
    io_outputdataconfig     = lo_output_data_config
    io_resourceconfig       = lo_resource_config
    io_stoppingcondition    = lo_stopping_condition
  ).
  MESSAGE 'Training job created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
  MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
  MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

"Wait for training job to be completed."
lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
  IF sy-index = 30.
    EXIT.          "Maximum 900 seconds."
  ENDIF.
  WAIT UP TO 30 SECONDS.
  lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
ENDWHILE.

"Create ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
EXPORTING
  iv_image      = iv_training_image
  iv_modeldataurl = lv_model_data_url.

"Create an Amazon SageMaker model."
TRY.
  lo_sgm->createmodel(
```



```

        iv_executionrolearn = iv_role_arn
        iv_modelname = iv_model_name
        io_primarycontainer = lo_primarycontainer
    ).
    MESSAGE 'Model created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

"Create an endpoint production variant."
CREATE OBJECT lo_production_variants
EXPORTING
    iv_variantname          = iv_ep_variant_name
    iv_modelname            = iv_model_name
    iv_initialinstancecount = iv_ep_initial_instance_count
    iv_instancetype         = iv_ep_instance_type.
INSERT lo_production_variants INTO TABLE lt_production_variants.

TRY.
    "Create an endpoint configuration."
    lo_ep_config_result = lo_sgm->createendpointconfig(
        iv_endpointconfigname = iv_ep_cfg_name
        it_productionvariants = lt_production_variants
    ).
    MESSAGE 'Endpoint configuration created.' TYPE 'I'.

    "Create an endpoint."
    oo_ep_output = lo_sgm->createendpoint(          " oo_ep_output is returned for
testing purposes. "
        iv_endpointconfigname = iv_ep_cfg_name
        iv_endpointname = iv_ep_name
    ).
    MESSAGE 'Endpoint created.' TYPE 'I'.
CATCH /aws1/cx_sgmresourcecelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
ENDTRY.

"Wait for endpoint creation to be completed."
DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
    IF sy-index = 30.
        EXIT.          "Maximum 900 seconds."
    ENDIF.

```

```
WAIT UP TO 30 SECONDS.
lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname = iv_ep_name ).
ENDWHILE.

TRY.
  "Delete an endpoint."
  lo_sgm->deleteendpoint(
    iv_endpointname = iv_ep_name
  ).
  MESSAGE 'Endpoint deleted' TYPE 'I'.

  "Delete an endpoint configuration."
  lo_sgm->deleteendpointconfig(
    iv_endpointconfigname = iv_ep_cfg_name
  ).
  MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.

  "Delete model."
  lo_sgm->deletemodel(
    iv_modelname = iv_model_name
  ).
  MESSAGE 'Model deleted.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
  DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception->
av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
  MESSAGE lv_endpointconfig_error TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [CreateEndpoint](#)
 - [CreateEndpointConfig](#)
 - [CreateModel](#)
 - [CreateTrainingJob](#)
 - [DeleteEndpoint](#)
 - [DeleteEndpointConfig](#)
 - [DeleteModel](#)
 - [DescribeEndpoint](#)
 - [DescribeTrainingJob](#)

Amazon SNS examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon SNS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateTopic

The following code example shows how to use CreateTopic.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result is  
returned for testing purposes. "  
    MESSAGE 'SNS topic created' TYPE 'I'.  
CATCH /aws1/cx_snstopiclimitexcdex.
```

```
MESSAGE 'Unable to create more topics. You have reached the maximum number
of topics allowed.' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateTopic](#) in *AWS SDK for SAP ABAP API reference*.

DeleteTopic

The following code example shows how to use DeleteTopic.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
  MESSAGE 'SNS topic deleted.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
  MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteTopic](#) in *AWS SDK for SAP ABAP API reference*.

GetTopicAttributes

The following code example shows how to use GetTopicAttributes.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
    DATA(lt_attributes) = oo_result->get_attributes( ).
    MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for SAP ABAP API reference*.

ListSubscriptions

The following code example shows how to use ListSubscriptions.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->listsubscriptions( ). " oo_result is
returned for testing purposes. "
    DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
    MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list subscribers.' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for SAP ABAP API reference*.

ListTopics

The following code example shows how to use ListTopics.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->listtopics( ).           " oo_result is returned for
testing purposes. "
    DATA(lt_topics) = oo_result->get_topics( ).
    MESSAGE 'Retrieved list of topics.' TYPE 'I'.
    CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list topics.' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTopics](#) in *AWS SDK for SAP ABAP API reference*.

Publish

The following code example shows how to use Publish.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->publish(                 " oo_result is returned for
testing purposes. "
    iv_topicarn = iv_topic_arn
    iv_message = iv_message
    ).
    MESSAGE 'Message published to SNS topic.' TYPE 'I'.
```

```
CATCH /aws1/cx_snsnotfoundexception.  
    MESSAGE 'Topic does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [Publish](#) in *AWS SDK for SAP ABAP API reference*.

SetTopicAttributes

The following code example shows how to use SetTopicAttributes.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_sns->settopicattributes(  
        iv_topicarn = iv_topic_arn  
        iv_attributename = iv_attribute_name  
        iv_attributevalue = iv_attribute_value  
    ).  
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.  
CATCH /aws1/cx_snsnotfoundexception.  
    MESSAGE 'Topic does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for SAP ABAP API reference*.

Subscribe

The following code example shows how to use Subscribe.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Subscribe an email address to a topic.

```
TRY.
    oo_result = lo_sns->subscribe(
        iv_topic_arn = iv_topic_arn
        iv_protocol = 'email'
        iv_endpoint = iv_email_address
        iv_returnsubscriptionarn = abap_true
    ).
    MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
CATCH /aws1/cx_snssubscriptionlmt00.
    MESSAGE 'Unable to create subscriptions. You have reached the maximum number
of subscriptions allowed.' TYPE 'E'.
ENDTRY.
```

"oo_result is returned for testing purposes."

- For API details, see [Subscribe](#) in *AWS SDK for SAP ABAP API reference*.

Unsubscribe

The following code example shows how to use Unsubscribe.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```

TRY.
  lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
  MESSAGE 'Subscription deleted.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
  MESSAGE 'Subscription does not exist.' TYPE 'E'.
CATCH /aws1/cx_snsinvalidparameterex.
  MESSAGE 'Subscription with "PendingConfirmation" status cannot be deleted/
  unsubscribed. Confirm subscription before performing unsubscribe operation.' TYPE
  'E'.
ENDTRY.

```

- For API details, see [Unsubscribe](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

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 SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic, subscribe an Amazon SQS FIFO queue to the topic, and publish a message to an Amazon SNS topic.

```

" Creates a FIFO topic. "
DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmw=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmw=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmw( iv_value = 'true' ).
INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

TRY.

```

```

        DATA(lo_create_result) = lo_sns->createtopic(
            iv_name = iv_topic_name
            it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.
    "
    ov_topic_arn is returned for testing purposes. "
    MESSAGE 'FIFO topic created' TYPE 'I'.
    CATCH /aws1/cx_snsstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum number
of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
    topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
    an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
            iv_topicarn = lv_topic_arn
            iv_protocol = 'sqs'
            iv_endpoint = iv_queue_arn
        ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.
    "
    ov_subscription_arn is returned for testing purposes. "
    MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum number
of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.
    
```

```
DATA(lo_result) = lo_sns->publish(
    iv_topicarn = lv_topic_arn
    iv_message = 'The price of your mobile plan has been increased from $19
to $23'
    iv_subject = 'Changes to mobile plan'
    iv_messagegroupid = 'Update-2'
    iv_messagededuplicationid = 'Update-2.1'
    it_messageattributes = lt_msg_attributes
).
ov_message_id = lo_result->get_messageid( ).
ov_message_id is returned for testing purposes. "
MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [CreateTopic](#)
 - [Publish](#)
 - [Subscribe](#)

Amazon SQS examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon SQS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateQueue

The following code example shows how to use CreateQueue.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
TRY.
    oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).      "
oo_result is returned for testing purposes. "
    MESSAGE 'SQS queue created.' TYPE 'I'.
    CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
queue with the same name.' TYPE 'E'.
    CATCH /aws1/cx_sqsqueueexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
ENDTRY.
```

Create an Amazon SQS queue that waits for a message to arrive.

```
TRY.
    DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
    DATA ls_attribute TYPE /aws1/
cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
    ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.      " Time in
seconds for long polling, such as how long the call waits for a message to arrive
in the queue before returning. "
```

```

        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
        oo_result = lo_sqs->createqueue(           " oo_result is returned
for testing purposes. "
            iv_queue_name = iv_queue_name
            it_attributes = lt_attributes
        ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
        CATCH /aws1/cx_sqsqueuedeletedrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
queue with the same name.' TYPE 'E'.
        CATCH /aws1/cx_sqsqueueexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDMETHOD.

```

- For API details, see [CreateQueue](#) in *AWS SDK for SAP ABAP API reference*.

DeleteQueue

The following code example shows how to use DeleteQueue.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

    TRY.
        lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
        MESSAGE 'SQS queue deleted' TYPE 'I'.
    ENDMETHOD.

```

- For API details, see [DeleteQueue](#) in *AWS SDK for SAP ABAP API reference*.

GetQueueUrl

The following code example shows how to use `GetQueueUrl`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->getqueueurl( iv_queue_name = iv_queue_name ).      "  
oo_result is returned for testing purposes. "  
    MESSAGE 'Queue URL retrieved.' TYPE 'I'.  
    CATCH /aws1/cx_sqsqueue_does_not_exist.  
        MESSAGE 'The requested queue does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for SAP ABAP API reference*.

ListQueues

The following code example shows how to use `ListQueues`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->listqueues( ).      " oo_result is returned for  
testing purposes. "  
    MESSAGE 'Retrieved list of queues.' TYPE 'I'.
```

```
ENDTRY.
```

- For API details, see [ListQueues](#) in *AWS SDK for SAP ABAP API reference*.

ReceiveMessage

The following code example shows how to use `ReceiveMessage`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue.

```
TRY.
    oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).    "
oo_result is returned for testing purposes. "
    DATA(lt_messages) = oo_result->get_messages( ).
    MESSAGE 'Message received from SQS queue.' TYPE 'I'.
CATCH /aws1/cx_sqsoverlimit.
    MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
ENDTRY.
```

Receive a message from an Amazon SQS queue using long-poll support.

```
TRY.
    oo_result = lo_sqs->receivemessage(                                " oo_result is returned for
testing purposes. "
        iv_queueurl = iv_queue_url
        iv_waittimeseconds = iv_wait_time    " Time in seconds for long
polling, such as how long the call waits for a message to arrive in the queue
before returning. "
    ).
    DATA(lt_messages) = oo_result->get_messages( ).
    MESSAGE 'Message received from SQS queue.' TYPE 'I'.
```

```
CATCH /aws1/cx_sqsoverlimit.  
    MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for SAP ABAP API reference*.

SendMessage

The following code example shows how to use SendMessage.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->sendmessage(           " oo_result is returned for  
testing purposes. "  
        iv_queueurl = iv_queue_url  
        iv_messagebody = iv_message  
    ).  
    MESSAGE 'Message sent to SQS queue.' TYPE 'I'.  
CATCH /aws1/cx_sqsinvalidmsgconts.  
    MESSAGE 'Message contains non-valid characters.' TYPE 'E'.  
CATCH /aws1/cx_sqsunsupportedop.  
    MESSAGE 'Operation not supported.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [SendMessage](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

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 SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic, subscribe an Amazon SQS FIFO queue to the topic, and publish a message to an Amazon SNS topic.

```

" Creates a FIFO topic. "
DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsm_w=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsm_w=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsm_w( iv_value = 'true' ).
INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

TRY.
  DATA(lo_create_result) = lo_sns->createtopic(
    iv_name = iv_topic_name
    it_attributes = lt_tpc_attributes
  ).
  DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
  ov_topic_arn = lv_topic_arn.
ov_topic_arn is returned for testing purposes. "
  MESSAGE 'FIFO topic created' TYPE 'I'.
CATCH /aws1/cx_snstopiclimitexcdex.
  MESSAGE 'Unable to create more topics. You have reached the maximum number
of topics allowed.' TYPE 'E'.
ENDTRY.

" Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
topic. "
" Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
an SNS FIFO topic. "
TRY.
  DATA(lo_subscribe_result) = lo_sns->subscribe(
    iv_topicarn = lv_topic_arn
    iv_protocol = 'sqs'
  )

```

```

        iv_endpoint = iv_queue_arn
    ).
    DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
    ov_subscription_arn = lv_subscription_arn.
    "
    ov_subscription_arn is returned for testing purposes. "
    MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
    CATCH /aws1/cx_snssubscriptionlmt00.
    MESSAGE 'Unable to create subscriptions. You have reached the maximum number
of subscriptions allowed.' TYPE 'E'.
    ENDRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from $19
to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes
        ).
        ov_message_id = lo_result->get_messageid( ).
        "
    ov_message_id is returned for testing purposes. "
    MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [CreateTopic](#)
- [Publish](#)
- [Subscribe](#)

Amazon Textract examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Textract.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

AnalyzeDocument

The following code example shows how to use AnalyzeDocument.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Detects text and additional elements, such as forms or tables,"
"in a local image file or from in-memory byte data."
"The image must be in PNG or JPG format."

"Create ABAP objects for feature type."
"Add TABLES to return information about the tables."
"Add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

"Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
  iv_name    = iv_s3object ).

"Create an ABAP object for the document."
DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).

"Analyze document stored in Amazon S3."
TRY.
  oo_result = lo_tex->analyzedocument(      "oo_result is returned for testing
purposes."
  io_document      = lo_document
  it_featuretypes  = lt_featuretypes ).
  LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
    IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
      MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
    ENDIF.
  ENDLOOP.
  MESSAGE 'Analyze document completed.' TYPE 'I'.
CATCH /aws1/cx_texaccessdeniedex.
  MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_texbaddocumentex.
  MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_texdocumenttoolargeex.
  MESSAGE 'The document is too large.' TYPE 'E'.
CATCH /aws1/cx_texhlquotaexceededex.
  MESSAGE 'Human loop quota exceeded.' TYPE 'E'.
CATCH /aws1/cx_texinternalservererr.
  MESSAGE 'Internal server error.' TYPE 'E'.

```

```
CATCH /aws1/cx_texinvalidparameterex.  
  MESSAGE 'Request has non-valid parameters.' TYPE 'E'.  
  
CATCH /aws1/cx_texinvalids3objectex.  
  MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.  
CATCH /aws1/cx_texprovthruputexcdex.  
  MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.  
CATCH /aws1/cx_texthrottlingex.  
  MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.  
CATCH /aws1/cx_texunsupporteddocex.  
  MESSAGE 'The document is not supported.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for SAP ABAP API reference*.

DetectDocumentText

The following code example shows how to use DetectDocumentText.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Detects text in the input document."  
"Amazon Textract can detect lines of text and the words that make up a line of  
text."  
"The input document must be in one of the following image formats: JPEG, PNG,  
PDF, or TIFF."  
  
"Create an ABAP object for the Amazon S3 object."  
DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket  
  iv_name   = iv_s3object ).  
  
"Create an ABAP object for the document."  
DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).
```

```

"Analyze document stored in Amazon S3."
TRY.
    oo_result = lo_tex->detectdocumenttext( io_document = lo_document ).
"oo_result is returned for testing purposes."
    LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
        IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR, '.
            MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
        ENDIF.
    ENDLOOP.
    DATA(lo_metadata) = oo_result->get_documentmetadata( ).
    MESSAGE 'The number of pages in the document is ' && lo_metadata->ask_pages( ) TYPE 'I'.
    MESSAGE 'Detect document text completed.' TYPE 'I'.
CATCH /aws1/cx_texaccessdeniedex.
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_texbaddocumentex.
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_texdocumenttoolargeex.
    MESSAGE 'The document is too large.' TYPE 'E'.
CATCH /aws1/cx_texinternalservererr.
    MESSAGE 'Internal server error.' TYPE 'E'.
CATCH /aws1/cx_texinvalidparameterex.
    MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
CATCH /aws1/cx_texinvalids3objectex.
    MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
CATCH /aws1/cx_texprovthruputexcdex.
    MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
CATCH /aws1/cx_texthrottlingex.
    MESSAGE 'The request processing exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_texunsupporteddocex.
    MESSAGE 'The document is not supported.' TYPE 'E'.
ENDTRY.

```

- For API details, see [DetectDocumentText](#) in *AWS SDK for SAP ABAP API reference*.

GetDocumentAnalysis

The following code example shows how to use GetDocumentAnalysis.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the results for an Amazon Textract"
"asynchronous operation that analyzes text in a document."
TRY.
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purposes."
    WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
        IF sy-index = 10.
            EXIT.                "Maximum 300 seconds.
        ENDIF.
        WAIT UP TO 30 SECONDS.
        oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
    ENDWHILE.

DATA(lt_blocks) = oo_result->get_blocks( ).
LOOP AT lt_blocks INTO DATA(lo_block).
    IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR, '.
        MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
    ENDIF.
ENDLOOP.
MESSAGE 'Document analysis retrieved.' TYPE 'I'.
CATCH /aws1/cx_texaccessdeniedex.
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_texinternalservererr.
    MESSAGE 'Internal server error.' TYPE 'E'.
CATCH /aws1/cx_texinvalidjobidex.
    MESSAGE 'Job ID is not valid.' TYPE 'E'.
CATCH /aws1/cx_texinvalidkmskeyex.
    MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
CATCH /aws1/cx_texinvalidparameterex.
    MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
CATCH /aws1/cx_texinvalids3objectex.
    MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
CATCH /aws1/cx_texprovthruputexcdex.
```

```

    MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
    CATCH /aws1/cx_textthrottlingex.
    MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    ENDTRY.

```

- For API details, see [GetDocumentAnalysis](#) in *AWS SDK for SAP ABAP API reference*.

StartDocumentAnalysis

The following code example shows how to use StartDocumentAnalysis.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Starts the asynchronous analysis of an input document for relationships"
"between detected items such as key-value pairs, tables, and selection"
"elements."

"Create ABAP objects for feature type."
"Add TABLES to return information about the tables."
"Add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
"Create an ABAP object for the Amazon S3 object."
DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
  iv_name = iv_s3object ).
"Create an ABAP object for the document."
DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

"Start async document analysis."

```



```

TRY.
    oo_result = lo_tex->startdocumentanalysis(      "oo_result is returned for
testing purposes."
    io_documentlocation      = lo_documentlocation
    it_featuretypes          = lt_featuretypes ).
    DATA(lv_jobid) = oo_result->get_jobid( ).

    MESSAGE 'Document analysis started.' TYPE 'I'.
    CATCH /aws1/cx_texaccessdeniedex.
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_texbaddocumentex.
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
    CATCH /aws1/cx_texdocumenttoolargeex.
    MESSAGE 'The document is too large.' TYPE 'E'.
    CATCH /aws1/cx_texidempotentprmmis00.
    MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
    CATCH /aws1/cx_texinternalservererr.
    MESSAGE 'Internal server error.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidkmskeyex.
    MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidparameterex.
    MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
    CATCH /aws1/cx_texinvalids3objectex.
    MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texlimitexceedex.
    MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
    CATCH /aws1/cx_texprovthruputexcdex.
    MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
    CATCH /aws1/cx_texthrottlingex.
    MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    CATCH /aws1/cx_texunsupporteddocex.
    MESSAGE 'The document is not supported.' TYPE 'E'.
ENDTRY.

```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for SAP ABAP API reference*.

StartDocumentTextDetection

The following code example shows how to use StartDocumentTextDetection.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Starts the asynchronous detection of text in a document."
"Amazon Textract can detect lines of text and the words that make up a line of
text."

"Create an ABAP object for the Amazon S3 object."
DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
      iv_name      = iv_s3object ).
"Create an ABAP object for the document."
DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).
"Start document analysis."
TRY.
      oo_result = lo_tex->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).
      DATA(lv_jobid) = oo_result->get_jobid( ).           "oo_result is returned
for testing purposes."
      MESSAGE 'Document analysis started.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
      MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
      MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
      MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texidempotentprmmis00.
      MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
      MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidkmskeyex.
      MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
      MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
      MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.

```

```
CATCH /aws1/cx_texlimitexceeddex.  
  MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.  
CATCH /aws1/cx_texprovthruputexcdex.  
  MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.  
CATCH /aws1/cx_texthrottlingex.  
  MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.  
CATCH /aws1/cx_texunsupporteddocex.  
  MESSAGE 'The document is not supported.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [StartDocumentTextDetection](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with document analysis

The following code example shows how to:

- Start asynchronous analysis.
- Get document analysis.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Create ABAP objects for feature type."  
"Add TABLES to return information about the tables."  
"Add FORMS to return detected form data."  
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes."  
  
DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(  
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )  
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
```

```
"Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
      iv_name   = iv_s3object ).

"Create an ABAP object for the document."
DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

"Start document analysis."
TRY.
    DATA(lo_start_result) = lo_tex->startdocumentanalysis(
        io_documentlocation    = lo_documentlocation
        it_featuretypes        = lt_featuretypes ).
    MESSAGE 'Document analysis started.' TYPE 'I'.
CATCH /aws1/cx_texaccessdeniedex.
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_texbaddocumentex.
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_texdocumenttoolargeex.
    MESSAGE 'The document is too large.' TYPE 'E'.
CATCH /aws1/cx_texidempotentprmmis00.
    MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
CATCH /aws1/cx_texinternalservererr.
    MESSAGE 'Internal server error.' TYPE 'E'.
CATCH /aws1/cx_texinvalidkmskeyex.
    MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
CATCH /aws1/cx_texinvalidparameterex.
    MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
CATCH /aws1/cx_texinvalids3objectex.
    MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
CATCH /aws1/cx_texlimitexceededex.
    MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
CATCH /aws1/cx_texprovthruputexcdex.
    MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
CATCH /aws1/cx_texthrottlingex.
    MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
CATCH /aws1/cx_texunsupporteddocex.
    MESSAGE 'The document is not supported.' TYPE 'E'.
ENDTRY.

"Get job ID from the output."
DATA(lv_jobid) = lo_start_result->get_jobid( ).
```

```
"Wait for job to complete."
oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).      " oo_result
is returned for testing purposes. "
WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
  IF sy-index = 10.
    EXIT.                  "Maximum 300 seconds."
  ENDIF.
  WAIT UP TO 30 SECONDS.
  oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).
ENDWHILE.

DATA(lt_blocks) = oo_result->get_blocks( ).
LOOP AT lt_blocks INTO DATA(lo_block).
  IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR, '.
    MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
  ENDIF.
ENDLOOP.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
 - [GetDocumentAnalysis](#)
 - [StartDocumentAnalysis](#)

Amazon Translate examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Translate.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

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

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

- [Scenarios](#)

Actions

DescribeTextTranslationJob

The following code example shows how to use `DescribeTextTranslationJob`.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the properties associated with an asynchronous batch translation job."
"Includes properties such as name, ID, status, source and target languages, and
input/output Amazon Simple Storage Service (Amazon S3) buckets."
TRY.
    oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
for testing purposes."
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Job description retrieved.' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

ListTextTranslationJobs

The following code example shows how to use ListTextTranslationJobs.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets a list of the batch translation jobs that you have submitted."

DATA lo_filter TYPE REF TO /aws1/cl_xl8textxl8tionjobfilt.

"Create an ABAP object for filtering using jobname."
CREATE OBJECT lo_filter
  EXPORTING
    iv_jobname = iv_jobname.

TRY.
  oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned for
testing purposes."
    EXPORTING
      io_filter      = lo_filter
    ).
  MESSAGE 'Jobs retrieved.' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
  MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidfilterex .
  MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex .
  MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
  MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTextTranslationJobs](#) in *AWS SDK for SAP ABAP API reference*.

StartTextTranslationJob

The following code example shows how to use StartTextTranslationJob.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts an asynchronous batch translation job."
"Use batch translation jobs to translate large volumes of text across multiple
documents at once."

DATA lo_inputdataconfig TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config."
CREATE OBJECT lo_inputdataconfig
EXPORTING
    iv_s3uri      = iv_input_data_s3uri
    iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config."
CREATE OBJECT lo_outputdataconfig
EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an internal table for target languages."
CREATE OBJECT lo_targetlanguagecodes
EXPORTING
    iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
    oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned for
testing purposes."
    EXPORTING
```



```

        io_inputdataconfig = lo_inputdataconfig
        io_outputdataconfig = lo_outputdataconfig
        it_targetlanguagecodes = lt_targetlanguagecodes
        iv_dataaccessrolelearn = iv_dataaccessrolelearn
        iv_jobname = iv_jobname
        iv_sourcelanguagecode = iv_sourcelanguagecode
    ).
    MESSAGE 'Translation job started.' TYPE 'I'.
    CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
    CATCH /aws1/cx_xl8invparamvalueex .
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
    CATCH /aws1/cx_xl8invalidrequestex .
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex .
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    CATCH /aws1/cx_xl8unsuppdedlanguage00 .
        MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

```

- For API details, see [StartTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

StopTextTranslationJob

The following code example shows how to use StopTextTranslationJob.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

"Stops an asynchronous batch translation job that is in progress."

TRY.

```

        oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned for
testing purposes."
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Translation job stopped.' TYPE 'I'.
    CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.

```

- For API details, see [StopTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

TranslateText

The following code example shows how to use TranslateText.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Translates input text from the source language to the target language."
TRY.
    oo_result = lo_xl8->translatetext(      "oo_result is returned for testing
purposes."
    EXPORTING
        iv_text      = iv_text
        iv_sourcelanguagecode = iv_sourcelanguagecode
        iv_targetlanguagecode = iv_targetlanguagecode
    ).
    MESSAGE 'Translation completed.' TYPE 'I'.
    CATCH /aws1/cx_xl8detectedlanguage00 .

```

```
MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
CATCH /aws1/cx_xl8internalserverex .
MESSAGE 'An internal server error occurred.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex .
MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8serviceunavailex .
MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE 'E'.
CATCH /aws1/cx_xl8textsizefmtexcdex .
MESSAGE 'The size of the text you submitted exceeds the size limit. ' TYPE
'E'.
CATCH /aws1/cx_xl8toomanyrequestsex .
MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
CATCH /aws1/cx_xl8unsuppdedlanguage00 .
MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
ENDTRY.
```

- For API details, see [TranslateText](#) in *AWS SDK for SAP ABAP API reference*.

Scenarios

Get started with translate jobs

The following code example shows how to:

- Start an asynchronous batch translation job.
- Wait for the asynchronous job to complete.
- Describe the asynchronous job.

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_inputdataconfig TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config."
CREATE OBJECT lo_inputdataconfig
  EXPORTING
    iv_s3uri          = iv_input_data_s3uri
    iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config."
CREATE OBJECT lo_outputdataconfig
  EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an internal table for target languages."
CREATE OBJECT lo_targetlanguagecodes
  EXPORTING
    iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
  DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
    EXPORTING
      io_inputdataconfig = lo_inputdataconfig
      io_outputdataconfig = lo_outputdataconfig
      it_targetlanguagecodes = lt_targetlanguagecodes
      iv_dataaccessrolelearn = iv_dataaccessrolelearn
      iv_jobname = iv_jobname
      iv_sourcelanguagecode = iv_sourcelanguagecode
    ).
  MESSAGE 'Translation job started.' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
  MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8invparamvalueex .
  MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex.
  MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
  MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
```

```

    CATCH /aws1/cx_xl8toomanyrequestsex.
      MESSAGE 'You have made too many requests within a short period of time. '
TYPE 'E'.
    CATCH /aws1/cx_xl8unsuppdedlanguage00 .
      MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.          "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
    ENDWHILE.

    TRY.
      oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
for testing purposes."
      EXPORTING
        iv_jobid      = lv_jobid
      ).
      MESSAGE 'Job description retrieved.' TYPE 'I'.
    CATCH /aws1/cx_xl8internalserverex .
      MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
      MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex.
      MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [DescribeTextTranslationJob](#)
- [StartTextTranslationJob](#)

Security in AWS SDK for SAP ABAP

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are 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 AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to AWS SDK for SAP ABAP, see [AWS services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This section covers the following topics.

Topics

- [SAP system authentication on AWS](#)
- [Best practices for IAM Security](#)
- [SAP authorizations](#)
- [Secure operations](#)
- [Using certificates with IAM Roles Anywhere](#)
- [Using SAP Credential Store](#)

SAP system authentication on AWS

Before an SAP system can make calls to AWS on behalf of SAP users, the SAP system must authenticate itself to AWS. AWS SDK for SAP ABAP supports the following three methods of authentication that are selected in the SDK profile settings in IMG.

AWS SDK for SAP ABAP - BTP edition can only be authenticated with the [the section called “Secret access key authentication”](#) method using SAP Credential Store.

Topics

- [Amazon EC2 instance metadata authentication](#)
- [Secret access key authentication](#)
- [Certificate-based authentication using IAM Roles Anywhere](#)
- [Next step](#)

Amazon EC2 instance metadata authentication

SAP systems running on Amazon EC2 can acquire short-lived, automatically-rotating credentials from Amazon EC2 instance metadata. For more information, see [Using credentials for Amazon EC2 instance metadata](#).

We strongly recommend this method of authentication while using SDK for SAP ABAP. To enable, the Basis administrator must enable outbound HTTP communication. No further Basis configuration is required.

Note

This method of authentication applies only if your SAP systems are running on Amazon EC2. SAP systems hosted on-premises or in other cloud environments cannot authenticate using this method.

Secret access key authentication

With this method, you use an Access Key ID and a Secret Access Key to authenticate your SAP system on AWS. The SAP system logs into AWS using an IAM user. For more information, see [Managing Access Keys for IAM Users](#).

The Basis administrator receives an Access Key ID and a Secret Access Key from the AWS IAM administrator. Your SAP system must be configured to store the Access Key ID and Secret Access Key.

- **Secure, store, and forward (SSF)**

- Use the SSF functionality to authenticate AWS SDK for SAP ABAP. For more information, see [Digital Signatures and Encryption](#).
- You can also test SSF's envelope and develop functionality with the SSF02 report. For more information, see [Testing the SSF Installation](#).
- The steps for configuring SSF for SDK for SAP ABAP are described in the /AWS1/IMG transaction. Go to **Technical Prerequisites**, and then select **Additional Settings** for On-Premises Systems.
- **SAP Credential Store**
 - Use SAP Credential Store to authenticate AWS SDK for SAP ABAP - BTP edition. For more information, see [What Is SAP Credential Store?](#)
 - See [Using SAP Credential Store](#) for configuration steps.

Certificate-based authentication using IAM Roles Anywhere

An X.509 certificate issued by your certificate authority (CA) can be used for authentication with AWS Identity and Access Management Roles Anywhere. The certificate must be configured in STRUST. The CA must be registered with IAM Roles Anywhere as a trust anchor, and a profile must be created to specify the roles and policies that IAM Roles Anywhere would assume. For more information, see [Creating a trust anchor and profile in AWS Identity and Access Management Roles Anywhere](#).

For detailed steps on how to use IAM Roles Anywhere with SDK for SAP ABAP, see [Using certificates with IAM Roles Anywhere](#).

Note

Certificate revocation is only supported through the use of imported certificate revocation lists. For more information, see [Revocation](#).

Next step

After authenticating your SAP system in AWS, SDK for SAP ABAP automatically performs an `sts:assumeRole` to assume the appropriate IAM role for the SAP user's business function.

Best practices for IAM Security

The IAM administrator will be responsible for the following three key areas.

- Ensuring that the SAP system can authenticate itself with Amazon EC2 metadata or Secret Key credentials.
- Ensuring that the SAP system has the permissions it needs to elevate itself with `sts:assumeRole`.
- For each logical IAM role, creating an IAM role for SAP users with the permissions required to perform the business functions (for example, the necessary permissions for Amazon S3, DynamoDB, or other services). These are the roles that SAP users will assume.

For more information, see the [Security](#) chapter in the SAP Lens: AWS Well-Architected Framework.

Topics

- [Best practice for Amazon EC2 instance profile](#)
- [IAM roles for SAP users](#)

Best practice for Amazon EC2 instance profile

The Amazon EC2 instance on which your SAP system runs has a set of authorizations based on its instance profile. Generally, the instance profile only needs to have permissions to call `sts:assumeRole`, to allow the SAP system to assume business-specific IAM roles as needed. This elevation to other roles ensures that an ABAP program can assume a role that gives the user the least privilege needed to do their job. For example, an instance profile might contain the following statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::0123456789:role/finance-cfo",
        "arn:aws:iam::0123456789:role/finance-auditor",
      ]
    }
  ]
}
```

```
        "arn:aws:iam::0123456789:role/finance-reporting"  
    ]  
  }  
]  
}
```

This preceding example allows the SAP system to assume the IAM roles for the CFO, AUDITOR, or REPORTING user. AWS SDK will choose the correct IAM role for the user based on the user's PFCG role in SAP.

Amazon EC2 instance profile can also be used for other functions.

- [AWS Backint Agent for SAP HANA](#)
- [SAP on AWS High Availability with Overlay IP Address Routing](#)

These solutions may also require `sts:assumeRole` permissions to roles specific to backup or failover or they may require permissions to be assigned directly to the instance profile.

IAM roles for SAP users

The ABAP program needs permissions to perform the user's job: read a DynamoDB table, invoke Amazon Textract on a PDF object in Amazon S3, run an AWS Lambda function. The same security model is used in all AWS SDKs. You can use an existing IAM role that was used for another AWS SDK.

The SAP business analyst will ask the IAM administrator for the `arn:aws:` of an IAM role for each logical role needed. For example, in a financial scenario, the business analyst may define the following logical IAM roles.

- CFO
- AUDITOR
- REPORTING

The IAM administrator will define IAM roles for each logical IAM role.

CFO

- `arn:aws:iam::0123456789:role/finance-cfo`
- read and write permissions to an Amazon S3 bucket

- read and write permissions to a DynamoDB database

AUDITOR

- `arn:aws:iam::0123456789:role/finance-auditor`
- read permissions to an Amazon S3 bucket
- read permissions to a DynamoDB database

REPORTING

- `arn:aws:iam::0123456789:role/finance-reporting`
- read permissions to a DynamoDB database
- no permission for the Amazon S3 bucket

The business analyst will enter the IAM roles into a mapping table to map the logical IAM roles with the physical IAM roles.

IAM roles for SAP users need to allow the `sts:assumeRole` action for trusted principals. The trusted principals can vary based on how the SAP system is authenticated on AWS. For more details, see [Specifying a principal](#).

The following are some examples of the most common SAP scenarios.

- **SAP system running on Amazon EC2 with an instance profile assigned** – here, an Amazon EC2 instance profile is attached to an IAM role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/SapInstanceProfile"
      }
    }
  ]
}
```

```
}
```

- **SAP systems running on Amazon EC2 without an instance profile** – here, Amazon EC2 assumes roles for SAP users.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": {
        "Service": [ "ec2.amazonaws.com" ]
      }
    }
  ]
}
```

- **SAP systems running on-premises** – SAP systems that run on-premises can only authenticate using the Secret Access Key. For more information, see [SAP system authentication on AWS](#).

Here, any IAM role assumed by an SAP user must have a trust relationship that trusts the SAP user.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/SAP_SYSTEM_S4H"
      }
    }
  ]
}
```

SAP authorizations

The authorization required to configure the SDK is dependent on the SDK edition.

Topics

- [Authorizations for configuration](#)
- [SAP authorizations for end users](#)

Authorizations for configuration

See the following tabs for more details.

SDK for SAP ABAP

The following authorizations are required to configure SDK for SAP ABAP.

- S_TCODE
 - TCD = /AWS1/IMG
- S_TABU_DIS
 - ACTVT = 02, 03
- DICBERCLS

Choose from the following authorization groups.

- /AWS1/CFG - AWS SDK for SAP ABAP v1 - Config
- /AWS1/MOD - AWS SDK for SAP ABAP v1 - Runtime
- /AWS1/PFL - AWS SDK for SAP ABAP v1 - SDK Profile
- /AWS1/RES - AWS SDK for SAP ABAP v1 - Logical Resources
- /AWS1/TRC - AWS SDK for SAP ABAP v1 - Trace

SDK for SAP ABAP - BTP edition

Use the following steps to allow SDK for SAP ABAP - BTP edition access to the configuration.

1. Create a new business role using the SAP_BR_BPC_EXPERT business role template. This template provides access to the Cutsom Business Configuration application.

2. Under **General Role Details**, go to **Access Categories**, and choose **Unrestricted** for *Read, Write, Value Help*.
3. Go to the **Business Catalog** tab, and assign the `/AWS1/RTBTP_BCAT` business catalog to provide access to the SDK configuration.
4. Go to the **Business Users** tab, and assign business users to grant access to the SDK configuration.

SAP authorizations for end users

Prerequisite: Define SDK Profiles

Before the SAP security administrator can define their roles, the Business Analyst will define SDK profiles in transaction `/AWS1/IMG` for AWS SDK for SAP ABAP or the Custom Business Configuration application for SDK for SAP ABAP - BTP edition. Typically, an SDK profile will be named according to its business function: ZFINANCE, ZBILLING, ZMFG, ZPAYROLL, etc. For each SDK profile, the Business Analyst will define logical IAM roles with short names, such as CFO, AUDITOR, REPORTING. These will be mapped to the real IAM roles by the IAM security administrator.

Define PFCG or Business Roles

Note

PFCG roles are called Business Roles in SAP BTP, ABAP environment.

The SAP security administrator will then add authorization object `/AWS1/SESS` to grant access to an SDK profile.

Auth Object `/AWS1/SESS`

- Field `/AWS1/PROF` = ZFINANCE

Users should also be mapped to logical IAM roles for each SDK profile, depending on their job function. For example, a financial auditor with reporting access might be authorized for a logical IAM role called AUDITOR.

Auth Object `/AWS1/LROL`

- Field /AWS1/PROF = ZFINANCE
- Field /AWS1/LROL = AUDITOR

Meanwhile, the CFO, with read/write authorizations, might have a PFCG role authorizing them the logical role of CF0.

Auth Object /AWS1/LROL

- Field /AWS1/PROF = ZFINANCE
- Field /AWS1/LROL = CFO

In general, a user should be authorized for only one logical IAM role per SDK profile. If a user is authorized for more than one IAM role (for example, if the CFO is authorized for both CF0 and AUDITOR logical IAM roles), then AWS SDK breaks the tie by ensuring that the higher priority (lower sequence number) role takes effect.

As with all security scenarios, users should be given least privilege to perform their job functions. A simple strategy for managing PFCG roles would be to name Single PFCG roles according to the SDK profile and logical role they authorize. For example, role Z_AWS_PFL_ZFINANCE_CF0 grants access to profile ZFINANCE and logical IAM role CF0. These single roles can then be assigned to composite roles that define job functions. Each company has their own strategy for role management, and we encourage you to define a PFCG strategy that works for you.

Secure operations

Encryption Of Data At Rest

AWS Secret Access Keys are used for authenticating the SDK. They are encrypted using the SSF or Credential Store functionality by SAP.

Encryption Of Data In Transit

All calls to AWS services are encrypted with HTTPS. The SAP ICM manages the HTTPS connection. AWS certificates must be trusted in STRUST.

API Usage

When an ABAP user assumes a role using `sts:assumeRole`, the session name is titled `USERID-SID-MANDT`, where:

- `USERID` is the ABAP user from `SY-UNAME` variable.
- `SID` is the ABAP system ID from `SY-SYSID` variable.
- `MANDT` is the ABAP client from `SY-MANDT` variable.

The session name appears in CloudTrail as *user name*. This ensures that API calls from an ABAP user can be traced back to the system, client, and user that initiated the call. For more information, see [What is AWS CloudTrail?](#)

Using certificates with IAM Roles Anywhere

SAP system can be authenticated on AWS by using certificated-based authentication with AWS Identity and Access Management Roles Anywhere. You must setup the certificate in STRUST, and configure the SDK profile in `/AWS1/IMG`.

Prerequisites

The following prerequisites must be met before commencing setup for certification.

- The X.509 certificate issued by your certificate authority (CA) must meet the following requirements.
 - The signing certificate must be a v3 certificate.
 - The chain must not exceed 5 certificates.
 - The certificate must support RSA or ECDSA algorithms.
- Register your CA with IAM Roles Anywhere as a trust anchor, and create a profile to specify the roles/policies for IAM Roles Anywhere. For more information, see [Creating a trust anchor and profile in AWS Identity and Access Management Roles Anywhere](#).
- IAM roles for SAP users must be created by the IAM administrator. The roles must have permissions to call the required AWS services. For more information, see [Best practices for IAM Security](#).
- Create authorization to run `/AWS1/IMG` transaction. For more information, see [Authorizations for configuration](#).

Procedure

Follow along these instructions to setup certificate-based authentication.

Steps

- [Step 1 – Define an SSF application by using SAP's Secure Store and Forward \(SSF\)](#)
- [Step 2 – Set SSF parameters](#)
- [Step 3 – Create the PSE and certificate request](#)
- [Step 4 – Import certificate response into the relevant PSE](#)
- [Step 5 – Configuring SDK profile to use IAM Roles Anywhere](#)

Step 1 – Define an SSF application by using SAP's Secure Store and Forward (SSF)

1. Run transaction code SE16 to define an SSF application.
2. Enter SSFAPPLIC table name, and select **New Entries**.
3. Enter a name for the SSF application in the APPLIC filed, a description in the DESCRIPT filed, and select Selected (X) option for the remaining fields.

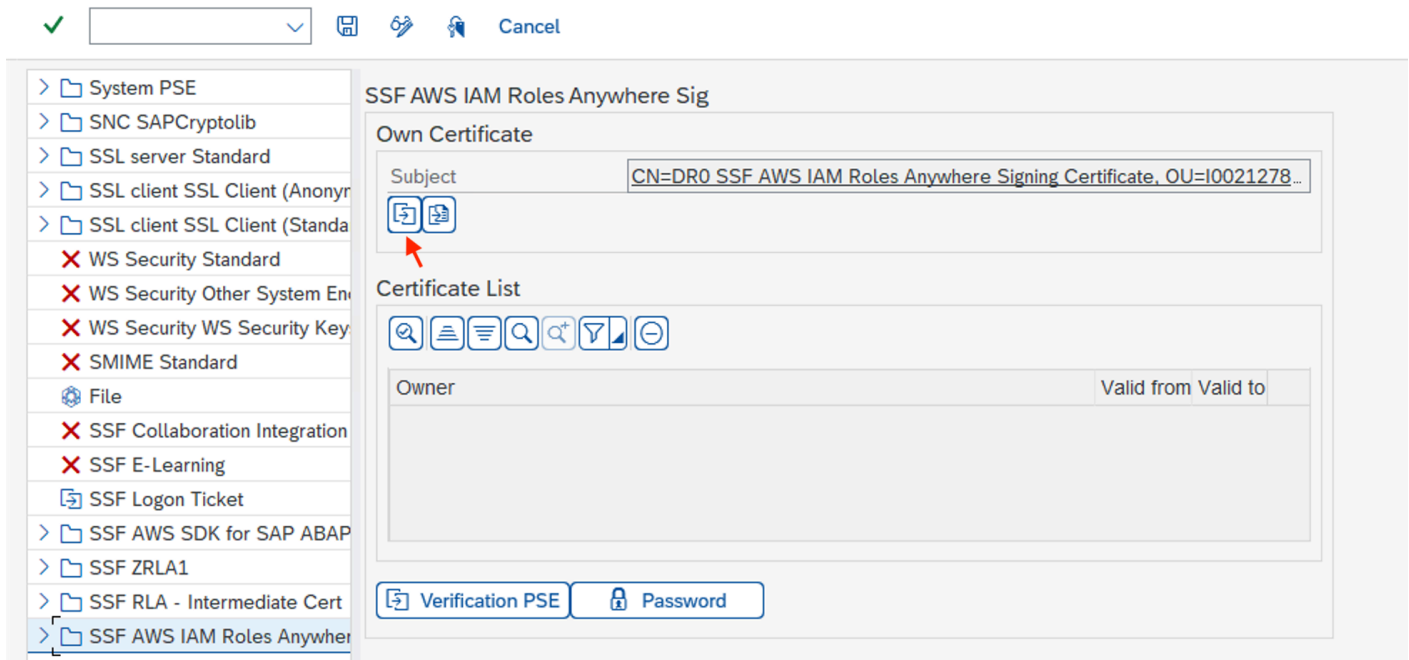
Step 2 – Set SSF parameters

1. Run the /n/AWS1/IMG to launch AWS SDK for SAP ABAP Implementation Guide (IMG).
2. Select **AWS SDK for SAP ABAP Settings > Technical Prerequisites > Additional Settings for On-Premises Systems**.
3. Run the **Set SSF Parameters** IMG activity.
4. Select **New Entries**, and choose the SSF application created in the previous step. Select **Save**.
5. Modify the hash algorithm to **SHA256**, and the encryption algorithm to **AES256-CBC**. Retain the other settings as default, and select **Save**.

Step 3 – Create the PSE and certificate request

1. Run the /n/AWS1/IMG transaction, and select **AWS SDK for SAP ABAP Settings > Technical Prerequisites > Additional Settings for On-Premises systems**.
2. Run the **Create PSE for SSF Application** IMG activity.

3. Select **Edit** for the STRUST transaction.
4. Right-select the SSF application created in [the section called "Step 1"](#), and choose **Create**. Retain all other default settings, and select **Continue**.
5. Select **Create Certificate Request**. See the following image. Retain the default options, and select **Continue**. Copy or export the generated certificate request, and provide it to your CA. Your CA verifies the request, and responds with a signed public-key certificate.



The signing process varies based on your CA, and the technology used by them. See [Issuing private end-entity certificates](#) with AWS Private Certificate Authority for an example.

Step 4 – Import certificate response into the relevant PSE

1. Run the `/n/AWS1/IMG` transaction, and select **AWS SDK for SAP ABAP Settings > Technical Prerequisites > Additional Settings for On-Premises systems**.
2. Run the `Create PSE for SSF Application IMG` activity.
3. Select **Edit** for the STRUST transaction.
4. Choose the SSF application, and then select **Import Certificate Response** located in the PSE section below the subject. Either copy and paste the certificate response into text box or import the file from the file system. Select **Continue > Save**.
5. The certificate details can be viewed by selecting the subject twice. The information is displayed in the certificate section.

Step 5 – Configuring SDK profile to use IAM Roles Anywhere

1. Run the `/n/AWS1/IMG` transaction, and select **AWS SDK for SAP ABAP Settings > Technical Prerequisites > Application Configurations**.
2. Create a new SDK profile, and name it.
3. Choose IAM Roles Anywhere as the authentication method.
 - In the left pane, select **Authentication and Settings**.
 - Create a new entry, and enter the information for your SAP system, and AWS Region.
 - Select **IAM Roles Anywhere** for the authentication method, and select **Save**.
 - Select **Enter Details**, and in the pop-up window, choose the SSF application created in [the section called “Step 1”](#). Enter the **Trust Anchor ARN**, and **Profile ARN** that were created in [the section called “Prerequisites”](#). See the following image. Select **Continue**.

Select Signing Certificate issued by your certificate authority (CA) from SSF

Certificate (SSF Application)

Enter your IAM Roles Anywhere details

Trust Anchor ARN

Profile ARN

✓ ✗

4. In the left pane, select **IAM Role Mapping**. Enter a name, and provide the IAM role's ARN provided by your IAM administrator.

For more information, see [Application configuration](#).

Using SAP Credential Store

SAP Credential Store is used in SAP Business Technology Platform to securely store credentials for secret access key authentication to AWS. You must have a subscription to use the service.

The following instructions assume that you have already configured an SDK profile. For more information, see [Configuring AWS SDK for SAP ABAP](#).

Before commencing the configuration, ensure that you meet the prerequisites. For more information, see [SAP Credential Store](#).

Topics

- [Configuration steps](#)
- [Using SAP Credential Store with the SDK](#)

Configuration steps

Steps

- [Step 1: Configure settings for authentication](#)
- [Step 2: Create a service key](#)
- [Step 3: Convert service key into .p12 format](#)
- [Step 4: Connect to SAP BTP, ABAP environment](#)

Step 1: Configure settings for authentication

Use the following steps to configure the Credential Store settings for authentication.

1. Navigate to the **Settings** tab of the SAP Credential Store instance.
2. Select **Edit Configurations**:
 - Choose **Mutual TLS** as the Default Authentication Type.
 - Select **Disabled** for Payload Encryption Status. The payload is encrypted in transit with HTTPS. However, the payload cannot currently be double-encrypted.
3. Select **Save**.

Step 2: Create a service key

Use the following steps to create a service key for Credential Store.

1. In the left pane of the SAP Credential Store application, navigate to **Service Keys**.
2. Select **Create Service Key**.

3. Enter a name for the service key, and select **Create**.

The service key is created on the basis of the chosen authentication type. Download the service key, and keep it secure for later usage.

Step 3: Convert service key into .p12 format

A client certificate in the .p12 format is required to create an outbound user for communication system. Use the following steps to generate a .p12 certificate from the certificate details provided in the Credential Store Service key.

1. Download the **SAP Cloud Root CA** certificate (required by SAP) from [SAP Trust Center Services](#).
2. Open the SAP Cloud Root CA certificate in any text file format. At the end of the file, press Enter, and copy-paste the certificate from the certificate field of the service key. Replace new line characters \n with actual new line (Enter), and save the entire certificate in .cer file format.
3. Copy the key from the key field of the service key. This private key must be treated as sensitive data. Paste it in a text file, and replace new line characters \n with actual new line (Enter). Save the private key in a text file.
4. With the certificate and private key generated in the previous steps, run the following command to generate a .p12 certificate.

```
openssl pkcs12 -export -out <.p12_<filename>> -inkey <private_key.key> -in  
<certificate.cer>
```

The command required verification of the export password. Retain the password for further use.

Delete the .key text file saved in your private key.

Step 4: Connect to SAP BTP, ABAP environment

Configure SAP BTP, ABAP environment to connect with SAP Credential Store.

Topics

- [Communication system](#)
- [Communication arrangement](#)

Communication system

Use the following steps to create a communication system that enables communication from SAP BTP, ABAP environment to SAP Credential Store.

1. Open the Fiori launchpad of the ABAP environment system.
2. Select the **Communication Systems** tile to open the application.
3. Select **New**.
4. Enter a name and ID for the communication system, and select **Create**. For example, you can name the system ZSAP_CREDSTORE.
5. Enter other required information:
 - **Host name:** Copy the host name from the Service Key URL. For example, if the URL is `https://credstore.mesh.cf.us10.hana.ondemand.com/api/v1/credentials`, then the host name is `credstore.mesh.cf.us10.hana.ondemand.com`.
 - **Users for Outbound Communication:** Select + to add a new user.
 - a. Select **SSL Client Certificate** as the Authentication mechanism.
 - b. Select **Upload New Certificate**:
 - Browse the .p12 certificate generated in the preceding step.
 - Enter a description.
 - Enter the export password that was used to generate the .p12 certificate.
 - Select **Upload**.
 - c. Select **Create** to create an outbound user.
6. Select **Save**.
7. Delete the service key downloaded in the previous step.

Communication arrangement

Use the following steps to create a communication arrangement to provide a communication scenario for outbound communication.

1. Open the Fiori launchpad of the ABAP environment system.
2. Select the **Communication Arrangements** tile to open the application.
3. Select **New**.

4. Select communication scenario `/AWS1/CRED_COMM_SCENARIO`, and enter a name for the communication arrangement. For example, `Z_AWS_SDK_TO_SAP_CREDSTORE`.
5. Select **Create**.
6. In the Communication System field, browse for the the Communication System created in the previous step. Other information is auto-populated post selection of the system.
7. Select **Save**.
8. Select **Check Connection** to test your connection.

Once this setup is complete, the ABAP environment can use the communication arrangement to use the SAP Credential Store service via outbound service (HTTP).

Using SAP Credential Store with the SDK

Steps

- [Step 1: Create a namespace and credential\(s\)](#)
- [Step 2: Configure Custom Business Configuration application](#)

Step 1: Create a namespace and credential(s)

Create a namespace and credential in SAP Credential Store with SAP Help – [Create, Edit, and Delete a Credential](#).

Enter the following details to create a credential of type **Key**.

- **Namespace** – Enter a name for the namespace, and group related credentials together.
- **Name** – Enter a name for the key. We recommend `aws-0123456789012-username`, where:
 - `0123456789012` is the AWS account ID to which the credential grants access
 - `username` is the IAM user name to which the credential belongs
- **Value** – Enter a base-64 encoded secret access key. Use the following command to base-64 encode your secret access key.

```
xargs echo -n | base64 # just press enter, do not enter arguments on the command line
MySecretAccessKey
Ctrl-D
```


The command reads the secret access key from standard input, and passes it to base64 without a trailing newline. It outputs the base-64 encoded secret access key to the screen. Clear or close your terminal after copying the value into SAP Credential Store.

- **Username** – Enter your access key ID.
- Select **Create**.

A new namespace with one credential is created, and credentials can be added, deleted or modified within this namespace.

Follow the principle of least privilege to manage access to the credentials stored in the namespace.

Step 2: Configure Custom Business Configuration application

Use the following steps to configure Custom Business Configuration application to define the credential to use for authentication by the SDK.

1. Open the Fiori launchpad of the ABAP environment system.
2. Browse **Custom Business Configuration** tile to open the application.
3. Open **SDK Profile** Business Configuration.
4. Select the SDK profile for which authentication settings must be configured for SAP Credential Store.
5. In the **Authentication and Settings** tab for the selected profile, select **Edit**, and enter the following details:
 - **Authentication Method** – Select **Credentials from SAP Credential Store**.
 - **Namespace** – Enter the namespace created in SAP Credential Store. For more information, see [the section called “Step 1: Create a namespace and credential\(s\)”](#).
 - **Key Name** – Enter the name of the created service key. For more information, see [the section called “Step 2: Create a service key”](#).
 - **Communication Arrangement** – Enter the name of the created communication arrangement. For more information, see [the section called “Communication arrangement”](#).
6. Select **Apply** to go to the **AWS SDK Profile** screen.
7. Select **Select Transport** to select the transport using the value help.
8. Select **Save**.

Troubleshoot AWS SDK for SAP ABAP

This section provides troubleshooting steps for possible error scenarios.

Topics

- [Import failure](#)
- [Unspecified location constraint](#)
- [SSL errors](#)
- [Profile configuration](#)
- [IAM authorization](#)
- [Authorization for performing required actions](#)
- [Active scenario](#)
- [Special characters in code](#)
- [Connectivity](#)

Import failure

Problem – Class 'CL_SYSTEM_UUID' doesn't contain an interface 'IF_SYSTEM_UUID_RFC4122_STATIC'

Cause – SAP Note 0002619546 is missing on your system.

Resolution – Ensure that the [SAP Note 0002619546](#) is applied to your system.

Unspecified location constraint

Problem – The unspecified location constraint is incompatible for the region specific endpoint this request was sent to

Cause – Your Amazon S3 bucket is missing the AWS Region in `io_createbucketconfiguration` parameter.

Resolution – When creating a bucket in any Region, except `us-east-1`, specify your Amazon S3 bucket's Region using `io_createbucketconfiguration` parameter in `createbucket()`. You don't have to specify a constraint for `us-east-1`.

The following example shows a correctly configured `io_createbucketconfiguration` parameter.

```
createbucket(  
  iv_bucket = 'test-bucket'  
  io_createbucketconfiguration = NEW /aws1/c1_s3_createbucketconf( 'us-west-1' )  
).
```

SSL errors

Problem – SSL Server Certificate Hostname Mismatch or SSL handshake with docs.aws.amazon.com:443 failed: SSSLERR_NO_SSL_RESPONSE

Cause – `icm/HTTPS/client_sni_enabled` parameter is not set to TRUE in the DEFAULT profile.

Resolution – Use the following steps to troubleshoot the given problems or any other SSL-related problem.

1. Open the SAPGUI and go to the command bar.
2. Run transaction RZ10.
3. Go to **Profile** and choose DEFAULT profile. The version is populated automatically.
4. In the **Edit Profile** section, select **Extended maintenance**, and then select **Change**.
5. Search for the `icm/HTTPS/client_sni_enabled` parameter.
 - If the parameter exists, edit the **Parameter value** and set it to TRUE.
 - If the parameter doesn't exist, create a parameter using the following steps.

1. Select **Parameter**.

Note

Ensure that you are selecting the Parameter for creation, and not editing (pencil icon).

2. Enter `icm/HTTPS/client_sni_enabled` in the **Parameter Name** field.
3. Enter TRUE in the **Parameter value** field.
4. Select **Save**.

6. Save these changes in the DEFAULT profile, and Exit.

Profile configuration

Problem – Could not find configuration under profile <profile_name> with scenario DEFAULT for <sid>:<client>

Causes – The <profile_name> is incorrect or hasn't been configured.

Resolution – Use the following steps to configure the profile.

1. Open SAPGUI and run transaction /n/AWS1/IMG.
2. Go to **Application Configuration > SDK Profile**.
 - If your profile is configured, verify that the profile name is correct.
 - If your profile is not configured, follow along the steps to configure a profile.
3. Select **New Entries**.
 - a. Enter a Name and Description for the profile.
 - b. Select **Save**.
4. Choose the entry you created in the previous step, and then select **Authentication and Settings**.
5. Select **New Entries**, enter the following details, and then select **Save**.
 - SID
 - Client
 - Scenario ID
 - AWS Region
 - Authentication Method
 - Select *Instance Role via Metadata* for SAP systems running in AWS.
 - Select *Credentials from SSF Storage* for SAP systems running on-premises or other cloud.
6. Select **IAM Role Mapping > New Entries**, enter the following details, and select **Save**.
 - Sequence number
 - Logical IAM Role
 - IAM Role ARN

IAM authorization

Problem – Could not assume role <iam_role_arn> or User: <user_arn> is not authorized to perform: sts:AssumeRole on resource:<iam_role_arn>

Causes – the following may be the possible reasons for this error.

- Incorrect IAM role ARN has been specified
- IAM user lacks permission to access the IAM role
- Lack of trust relationship between the assumed IAM role and the assuming IAM role or IAM user

Resolution – Use the following steps to ensure that the IAM role ARN is correct.

1. Open SAPGUI and run transaction /n/AWS1/IMG.
2. Go to **Application Configuration > SDK Profile**, and choose the profile that has been configured with your IAM role.
3. Select **IAM Role Mapping** and verify or correct your IAM role ARN.
 - If your IAM role ARN is correct, ensure that your IAM role has been configured properly. For more information, see [Troubleshooting IAM roles](#).

Authorization for performing required actions

Problem – User <user_arn> is not authorized to perform: <action> on resource: <resource_arn>

Cause – User does not have permissions to perform an action.

Resolution – user_arn must be set up with required permissions on resource_arn to perform a specified action. For more information, see [Permissions required to access IAM resources](#).

Active scenario

Problem – No active scenario configured

Cause – The setup of active scenario was missed.

Resolution – See [Runtime settings](#) to configure an active scenario.

Special characters in code

Warning – The character 0x00A0 cannot be part of an ABAP word

Note

This warning may be preceded by varied error messages.

Cause – Copying and pasting code from different sources can insert special characters in your code.

Resolution – When you paste any code in the ABAP source code editor, you see the following pop-up.

Non-breaking space characters were detected. Convert to spaces?

Choose **Yes** to answer this question. Also, we recommend selecting the code to copy it, instead of using the copy button in code boxes.

Connectivity

Problem – SCLNT_HTTP(411) : Direct connect to tla.region.amazonaws.com:443 failed: NIECONN_REFUSED(-10)

Cause – The SAP system does not have internet connectivity, and cannot establish a TCP/IP connection to port 443 of tla.region.amazonaws.com.

Resolution – The SAP system must be able to establish connection to AWS endpoints on HTTPS port 443, either directly or through a proxy server. You can establish/verify internet connectivity with one of the following options.

- Direct outbound connection to internet through a NAT or internet gateway
- Connection through a proxy server

For more information, see [Connection through a proxy server](#).

Additional topics

This section covers the following topics.

Topics

- [AWS SDK for SAP ABAP releases](#)
- [SAP licensing](#)

AWS SDK for SAP ABAP releases

AWS SDK for SAP ABAP is delivered in transports, and AWS SDK for SAP ABAP - BTP edition is delivered as add-ons. The mechanism to import transports and add-ons is different but the technical functionality is the same. For more information, see [Setting up](#).

Topics

- [Release strategy](#)
- [Best practices](#)
- [Patching SDK for SAP ABAP](#)
- [Installing an additional module](#)
- [Uninstalling SDK for SAP ABAP](#)

Release strategy

Version 1 of AWS SDK for SAP ABAP is updated frequently. New patches are released weekly or daily based on the releases and updates of AWS services. The patches for AWS services can include bug fixes and other changes that update patch level of the SDK. For more information, see [AWS SDKs and Tools maintenance policy](#).

Best practices

We recommended retaining same patch level of SDK for SAP ABAP for all SAP systems (development, QA, and production).

When patching the SDK, import the latest version in your sandbox. You can then import it to the development, QA, and production systems, following your normal change control procedures.

Patching SDK for SAP ABAP

Each SDK for SAP ABAP release is delivered as a set of cumulative transports, including all bug fixes, features, and updates. There is no difference between a patch and an installation transport. You must import the latest transports to patch SDK for SAP ABAP.

Due to dependencies of core Runtime and API modules, you must patch the core module and all other modules that you installed, even if you are not using those modules anymore. For example, if you imported the core, ec2, and lmd transports when you installed the SDK, you must import the latest transports for core, ec2, and lmd when patching.

Installing an additional module

Import the transport for the new module at the same patch level as your existing core and modules to install an additional API module in your SAP system. Follow the guidelines in [the section called "Patching SDK for SAP ABAP"](#) if you want to import a more recent version of the module. This ensures that the patch levels are compatible across all SDK modules.

Uninstalling SDK for SAP ABAP

To uninstall SDK for SAP ABAP, you must download a *deletion transports* kit from <https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/uninstall-abapsdk-LATEST.zip>.

```
curl "https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/uninstall-abapsdk-LATEST.zip" -o "uninstall-abapsdk-LATEST.zip"
```

You can download a signature file from <https://sdk-for-sapabap.aws.amazon.com/awsSdkSapabapV1/release/uninstall-abapsdk-LATEST.sig>. To validate the file, see [Verify SDK for SAP ABAP](#).

For each SDK module installed on your SAP system, the corresponding *deletion transport* must be imported from the preceding ZIP file. You can remove a single module without uninstalling the entire SDK. You can do so by importing only the *deletion transport* for module you want to remove. If you are uninstalling the entire SDK with all of its modules, then the *core deletion transport* must be imported last.

We recommend that you test the uninstallation in a sandbox before attempting in development, QA or production systems.

Considerations

Before uninstalling the SDK, see the following considerations.

- SDK configuration settings will be lost. The IMG must be reconfigured on installation.
- If you have Z programs that rely on the SDK, they will generate syntax errors after the removal of the SDK.
- PFCG or Business roles containing SDK authorization references will have invalid authorizations after the removal of the SDK. Remove SDK authorization references from the PFCG roles before uninstalling the SDK.

Note

AWS SDK for SAP ABAP - BTP edition cannot be uninstalled during the developer preview.

SAP licensing

The use of SAP software is subject to SAP's terms. You are responsible for complying with SAP licensing terms, including software distribution and indirect licensing conditions. Any information provided is not legal advice, and should not be relied upon for licensing compliance purposes. If you have questions about your licensing or rights to SAP software, consult your legal team, SAP, and/or your SAP reseller.

Question : Will SDK for SAP ABAP usage affect my SAP license?

Answer : AWS SDK for SAP ABAP enables you to consume AWS services with your own ABAP code. It is used in integration scenarios between an SAP system and AWS services. Any scenario where data from the SAP system is sent to a third-party (non-SAP) system, or created by that system, may have implications for indirect licensing. SAP has multiple approaches for defining indirect access, such as user-based calculations and outcome-based calculations. The methodology to define indirect access depends on your contract with SAP. You must be aware of the guidance provided in your contract with SAP, and you can further discuss this with SAP or their reseller.

In 2018, SAP released two documents – *Indirect Access Guide for SAP Installed Base Customers* and *SAP ERP Pricing for Digital Age - Addressing Indirect/Digital Access*. These documents can be found on SAP websites, and are examples of indirect licensing approaches. However, these documents do not reflect your particular agreement with SAP.

Document history for AWS SDK for SAP ABAP Developer Guide

The following table describes the documentation releases for AWS SDK for SAP ABAP.

Change	Description	Date
New content	Developer preview of SDK for SAP ABAP - BTP edition.	May 31, 2024
New content	Added Using certificates with IAM Roles Anywhere .	December 1, 2023
New content	Added Building products with SDK .	December 1, 2023
New content	Added Retry behavior .	December 1, 2023
New content	Added SAP licensing .	September 22, 2023
Public release	Initial relaunch of AWS SDK for SAP ABAP Developer Guide.	June 30, 2023
New content	Added AWS SDK for SAP ABAP features .	May 30, 2023
New content	Added Troubleshoot AWS SDK for SAP ABAP .	February 17, 2023
Developer preview	Developer preview of AWS SDK for SAP ABAP Developer Guide.	November 17, 2022