



**S3 for SAP – AWS S3
SDK for ABAP.
Community edition
version Z**

Table of Contents

WHAT IS AWS S3 SDK FOR ABAP	3
USE CASES.....	4
BENEFITS.....	4
PREREQUISITES	6
INSTALLATION	6
PRECONDITION	6
ACTIONS TO DO IN AWS CONSOLE	7
SCREENSHOTS	9
ACTIONS TO DO IN SAP TARGET SYSTEM.....	13
AUTHORIZATIONS TO RUN S3 FOR SAP.....	13
INSTALLATION.....	13
INITIAL LOAD OF TABLES	16
STRUST.....	17
LOG VIEWER.....	25
DEMO PROGRAMS	25
PROGRAM ZLNKEAWS_S3_DEMO_IAM.....	25
TECHNICAL EXPLANATION	26
PROGRAM ZLNKEAWS_S3_DEMO_BUCKET	27
TECHNICAL EXPLANATION	28
PROGRAM ZLNKEAWS_S3_DEMO_FILE.....	34
TECHNICAL EXPLANATION	35
PROGRAM ZLNKEAWS_S3_DEMO_FOLDER	39
TECHNICAL EXPLANATION	39
PROGRAM ZLNKEAWS_S3_DEMO_S3.....	43
TECHNICAL EXPLANATION	43
CONCLUSION	44

REVISIONS

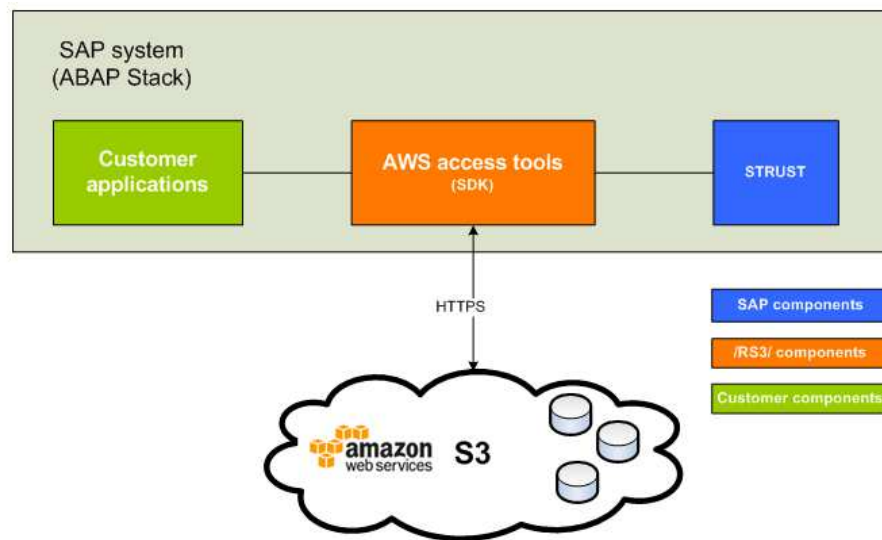
Document version	Author	Date
V1.0	Jordi Escoda	Dec 19 th 2016

What is AWS S3 SDK for ABAP

AWS S3 for ABAP is an ABAP AddOn which enables native integration from ABAP to AWS S3. You will be able to manage buckets, folders and files on AWS S3.

It has two editions: Community and Commercial.

The community edition is the ABAP SDK which you can use as a tool to write your own programs to read from AWS S3 and write to AWS S3. Demo programs are provided as a reference.



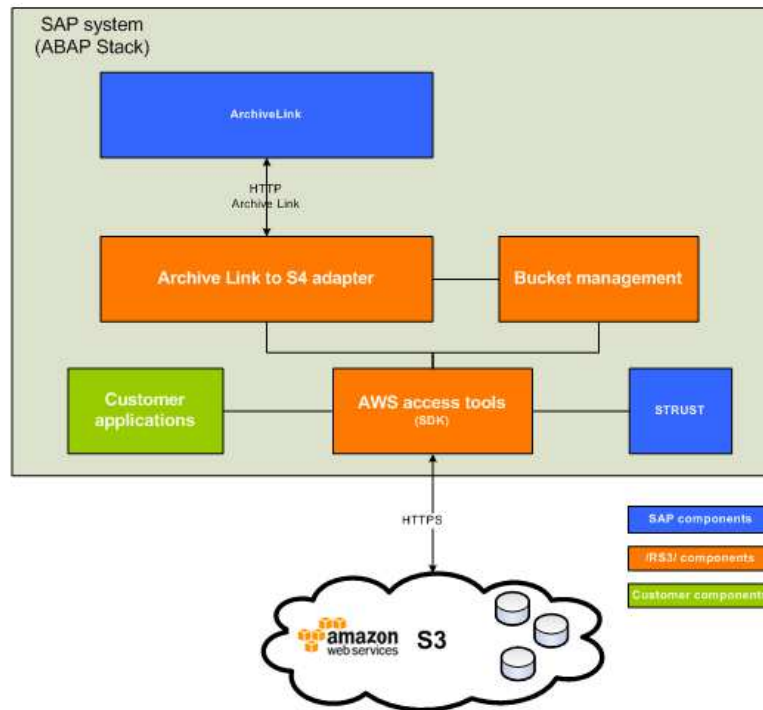
There are two repositories for the Community Edition:

- Under namespace /RS3/
https://github.com/LinkeIT/AWS_S3_SDK_for_ABAP
- Under namespace ZLNKE
https://github.com/LinkeIT/AWS_S3_SDK_for_ABAP_Z

Feel free to use Z version if you want to contribute by adding new functionalities.

The installation instructions given in this document refers to Z version.

The Commercial edition (complete solution) maps from ArchiveLink to AWS S3. In this way you can use S3 as a Content Server. As well you can make use of the ABAP SDK to develop your own programs.



To learn about AWS S3, read: <https://aws.amazon.com/s3/>

Contact with us if you want to use the Commercial edition.

Use cases

- Replace your Content Server by S3 for SAP.
- Store attachments in S3 for SAP
- Store Business Documents in S3 for SAP
- Store archiving sessions in S3 for SAP
- Integration with AWS services which use S3 as input or output, for example: Big Data, Machine Learning, etc...
- Develop custom programs (Z) that integrate with AWS S3 and leverage AWS advantages.

Benefits

Using S3 for SAP gives you these benefits:

- **Simplicity.** AddOn which can be installed in any SAP system, without the need of additional servers. Only depends on BASIS package.
- **Quickly available.** You install and run S3 for SAP in just one hour.

- **No practical storage limits.** You don't need to take care about space limitations.
- **Pay as you go,** pricing is based on the actual storage you use. No need to invest money in any infrastructure.
- **Cloud Compliance.** Your data will be safe in S3. Take advantage of AWS Compliance, meeting plenty of standards, regulations and best practises. Read <http://aws.amazon.com/compliance/> for further information.
- **Data persistence** is guaranteed 99.999999999%.
- **Data availability** is guaranteed 99.99%
- **Reduce your IT infrastructure.** Forget about content servers and related costs (purchase, licenses, maintenance, backups, power consumption, cooling, etc...)
- **Testing.** Test easily your archiving projects using development or quality systems. You don't need to ask for any storage server.
- **Seamlessly use.** You don't need to acquire additional knowledge, you will be able use the standard archiving tools (SARA) and GOS in the same way you are used to.

With AWS S3 SDK for ABAP community edition you can:

- Manage your Buckets
- Choose the region where your buckets are stored. This is convenient to meet country regulations
- Write your own programs to read from AWS S3 and write to AWS S3.
- Leverage other AWS services from S3.

With the Commercial solution, in addition you can:

- Easily manage your archiving objects
- Store archiving sessions in AWS S3
- Store business documents in AWS S3
- Retrieve your archiving sessions from AWS S3
- Retrieve your business documents from AWS S3
- On-the-fly encrypting / decrypting with your own SSL certificate
- On-the-fly compression and decompression
- Storage encrypted on server side
- Automatically move to AWS Glacier your data after the period of time to further save costs
- Migrate your stored data to AWS S3
- Migrate back your data to your servers. Your data is own by you and you will always be free to migrate back your data to your servers.

The Commercial solution offers support in any incident which may arise and in case of upgrades and support packages installation. Therefore is the most adequate for productive environments, where the companies require a support of business level.

Prerequisites

To run AWS S3 SDK for ABAP Community edition in your SAP system, following prerequisites should be met.

- An AWS account + Privileges to create IAM users
- SAP Netweaver 7.0 or higher (SAP_BASIS 700 0028 SAPKB70028)
- SAP Kernel release 720 or higher
- SAP Cryptolib properly installed
- ICM Services HTTP and HTTPS configured and active
- Connectivity to the AWS endpoints (either directly or through a Proxy properly configured).
- OpenSSL installed on the OS (Linux or Windows)
- To install the add-on, a user with enough privileges (please read Authorization)

The service of installing and configuring is included in the Commercial edition.

Installation

Precondition

If you choose to use Z version, you should check that namespace ZLNKE is free to use, i.e. no collision is possible.

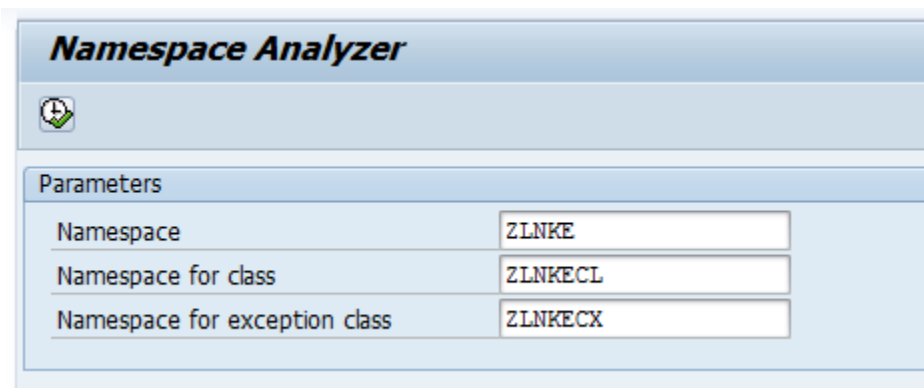
To do so install report ZLNKE_NAMESPACE_ANALYZER. You can download it from <https://github.com/LinkerIT/abapNamespaceAnalyzer/>

Run ZLNKE_NAMESPACE_ANALYZER with paramters:

Namespace ZLNKE

Namespace for class ZLNKECL

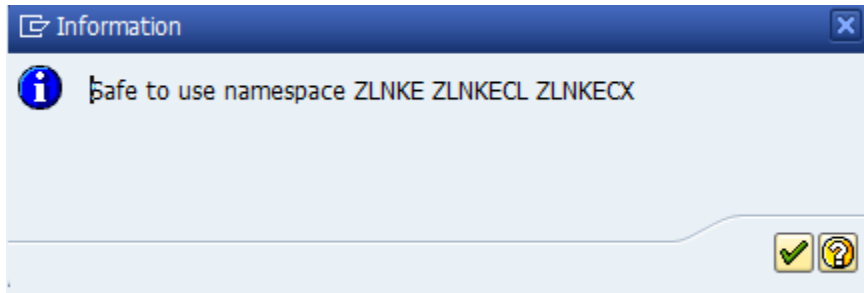
Namespace for exception class ZLNKECX



The screenshot shows the 'Namespace Analyzer' report interface. It has a title bar with the text 'Namespace Analyzer' and a green checkmark icon. Below the title bar is a section labeled 'Parameters' containing three input fields:

Parameters	
Namespace	ZLNKE
Namespace for class	ZLNKECL
Namespace for exception class	ZLNKECX

It is safe to use ZLNKE If it shows:



Don't install S3 for SAP version Z if you see a list with possible collisions.

Actions to do in AWS Console

Create an IAM user with following privileges.

For Bucket operations:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::<sid>-*"
      ]
    }
  ]
}
```

Where <sid> is the SID of your SAP system, **in lowercase**.

You can download this json file from

https://github.com/LinkerIT/AWS_S3_SDK_for_ABAP/blob/master/Bucket_Policy.json

For IAM GetUser:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser"
      ],
      "Resource": [
        "arn:aws:iam::<aws_account_id>:user/<iam_user>"
      ]
    }
  ]
}
```

Where <aws_account_id> is your AWS account ID and <iam_user> is your iam user

You can download this json file from

https://github.com/LinkerIT/AWS_S3_SDK_for_ABAP/blob/master/IAM_Policy.json

For Listing Buckets:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

You can download this json file from

https://github.com/LinkeIT/AWS_S3_SDK_for_ABAP/blob/master/S3_Policy.json

Example: If SID is DES, AWS account is 997663152801 and user is S3_user the resulting policy should be:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::des-*"
      ]
    }
  ]
}

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser"
      ],
      "Resource": [
        "arn:aws:iam::997663152801:user/s3_user"
      ]
    }
  ]
}

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

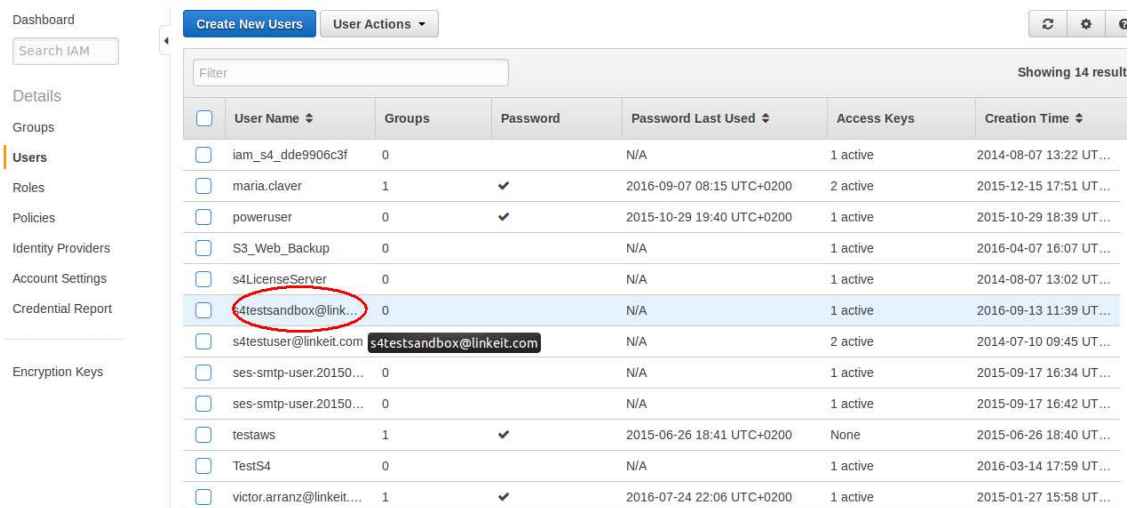
To learn more about Bucket Policies, read

<http://docs.aws.amazon.com/AmazonS3/latest/dev/using-iam-policies.html>

To learn more about IAM Policies, read http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

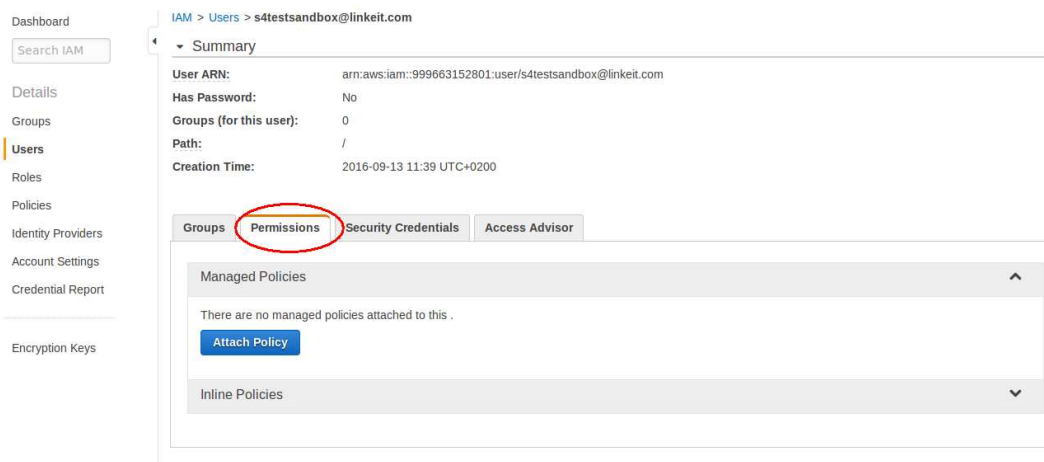
Screenshots

To attach the policy follow these steps:
On AWS console go to IAM.
Click on the user



	User Name	Groups	Password	Password Last Used	Access Keys	Creation Time
<input type="checkbox"/>	iam_s4_dde9906c3f	0		N/A	1 active	2014-08-07 13:22 UT...
<input type="checkbox"/>	maria.claver	1	✓	2016-09-07 08:15 UTC+0200	2 active	2015-12-15 17:51 UT...
<input type="checkbox"/>	poweruser	0	✓	2015-10-29 19:40 UTC+0200	1 active	2015-10-29 18:39 UT...
<input type="checkbox"/>	S3_Web_Backup	0		N/A	1 active	2016-04-07 16:07 UT...
<input type="checkbox"/>	s4LicenseServer	0		N/A	1 active	2014-08-07 13:02 UT...
<input type="checkbox"/>	s4testsandbox@linkeit.com	0		N/A	1 active	2016-09-13 11:39 UT...
<input type="checkbox"/>	s4testuser@linkeit.com	s4testsandbox@linkeit.com		N/A	2 active	2014-07-10 09:45 UT...
<input type="checkbox"/>	ses-smtp-user.20150...	0		N/A	1 active	2015-09-17 16:34 UT...
<input type="checkbox"/>	ses-smtp-user.20150...	0		N/A	1 active	2015-09-17 16:42 UT...
<input type="checkbox"/>	testaws	1	✓	2015-06-26 18:41 UTC+0200	None	2015-06-26 18:40 UT...
<input type="checkbox"/>	TestS4	0		N/A	1 active	2016-03-14 17:59 UT...
<input type="checkbox"/>	victor.arranz@linkeit....	1	✓	2016-07-24 22:06 UTC+0200	1 active	2015-01-27 15:58 UT...

Permissions



Dashboard

Search IAM

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

IAM > Users > s4testsandbox@linkeit.com

Summary

User ARN: arn:aws:iam::999663152801:user/s4testsandbox@linkeit.com

Has Password: No

Groups (for this user): 0

Path: /

Creation Time: 2016-09-13 11:39 UTC+0200

Groups Permissions Security Credentials Access Advisor

Managed Policies

There are no managed policies attached to this .

Attach Policy

Inline Policies

Inline policies...

Dashboard

Search IAM

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

IAM > Users > s4testsandbox@linkeit.com

Summary

User ARN: arn:aws:iam::999663152801:user/s4testsandbox@linkeit.com

Has Password: No

Groups (for this user): 0

Path: /

Creation Time: 2016-09-13 11:39 UTC+0200

Groups Permissions Security Credentials Access Advisor

Managed Policies

There are no managed policies attached to this .

Attach Policy

Inline Policies

There are no inline policies to show. To create one, [click here](#) .

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

☐ Policy Generator

☒ Custom Policy

Use the policy editor to customize your own set of permissions.

Select

Policy name, and paste the policy. Apply Policy

Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the [Using IAM](#) guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

Policy Name

BucketOperations

Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::DES-*"
      ]
    }
  ]
}
```

☒ Use autoformatting for policy editing

Cancel Validate Policy Apply Policy

Create another Policy for IAM

Creation Time: 2016-09-13 11:39 UTC+0200

Groups Permissions Security Credentials Access Advisor

Managed Policies

There are no managed policies attached to this .

Attach Policy

Inline Policies

This view shows all inline policies that apply to this user, including policies that are embedded in this user and policies that are embedded in groups that this user is in.

Create User Policy

Policy Name	Actions
BucketOperations	Show Policy Edit Policy Remove Policy Simulate Policy

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

☐ Policy Generator

☒ Custom Policy

Use the policy editor to customize your own set of permissions.

Select

Policy name, and paste the policy. Apply Policy

Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

Policy Name

IAM

Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser"
      ],
      "Resource": [
        "arn:aws:iam::997663152801:user/s3_user"
      ]
    }
  ]
}
```

☒ Use autoformatting for policy editing

Cancel Validate Policy Apply Policy

Create another Policy for ListAllMyBuckets.

Creation Time: 2016-09-13 11:39 UTC+0200

Groups Permissions Security Credentials Access Advisor

Managed Policies

There are no managed policies attached to this .

Attach Policy

Inline Policies

This view shows all inline policies that apply to this user, including policies that are embedded in this user and policies that are embedded in groups that this user is in.

Create User Policy

Policy Name	Actions
BucketOperations	Show Policy Edit Policy Remove Policy Simulate Policy
IAM	Show Policy Edit Policy Remove Policy Simulate Policy

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

☐ Policy Generator

☒ Custom Policy

Use the policy editor to customize your own set of permissions.

Select

Policy name, and paste the policy. Apply Policy

Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

Policy Name

ListAllMyBuckets

Policy Document

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "s3:ListAllMyBuckets"
8       ],
9       "Resource": [
10        "arn:aws:s3:*:*"
11      ]
12    }
13  ]
14 }
```

☒ Use autoformatting for policy editing

Cancel Validate Policy Apply Policy

This ends IAM user preparation

Actions to do in SAP target system

Authorizations to run S3 for SAP

To run S3 for SAP prepare a role with the following privileges:

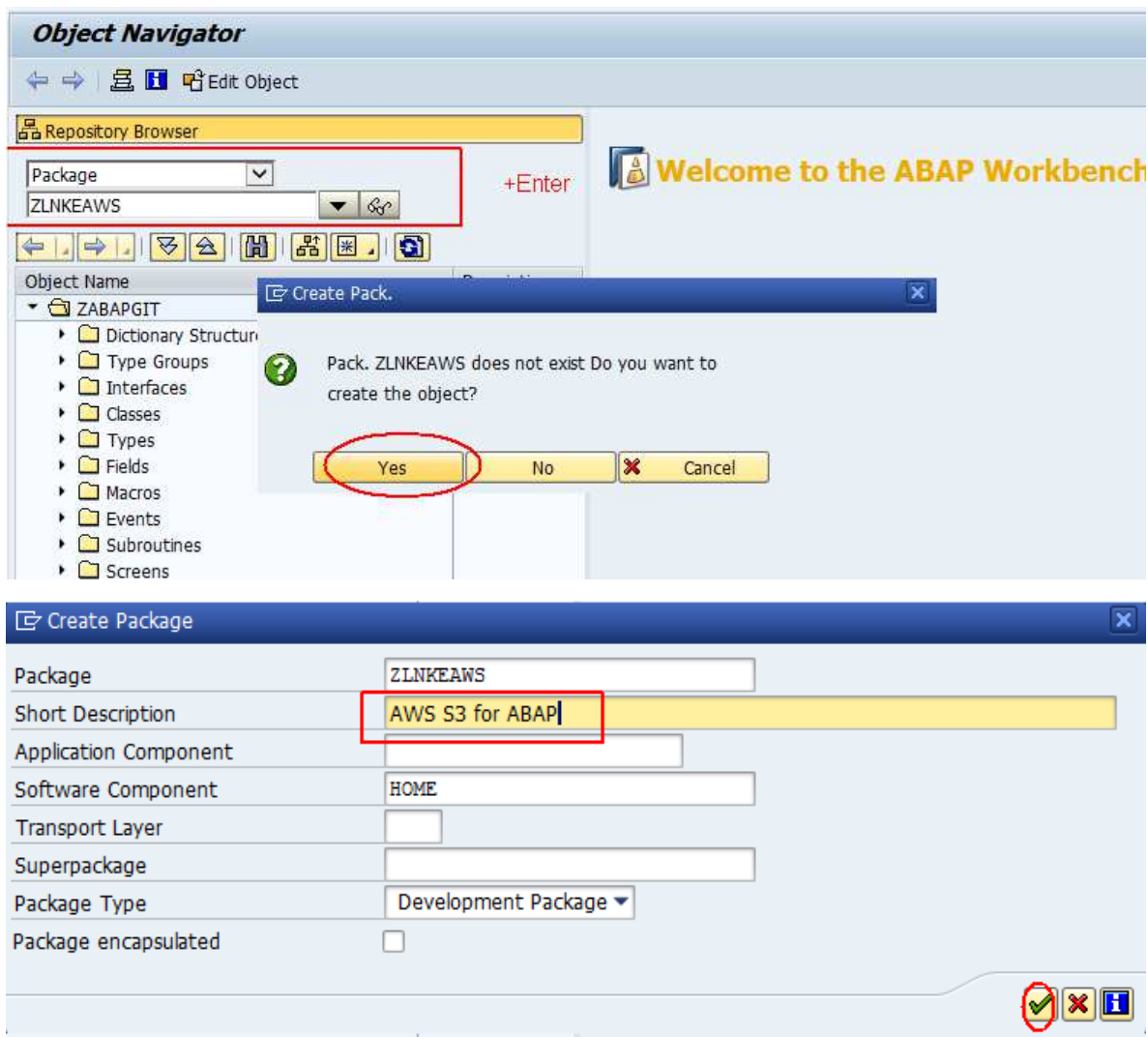
S_RFC_ADM with ACTVT=*, RFCDEST=RS3_*

Installation

You can easily install by using abapGit, from Lars Hvam.

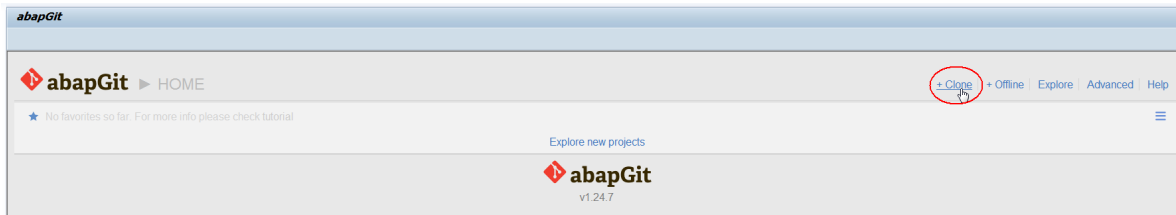
If you don't have abapGit yet in your system check <https://github.com/larshp/abapGit> and install it.

Create in SE80 package ZLNKEAWS



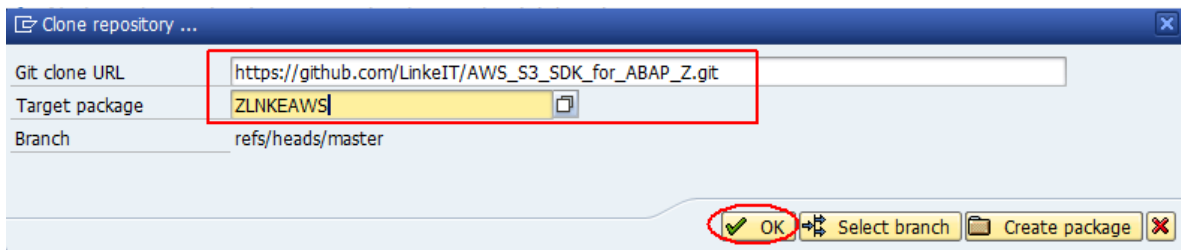
Give a transport request

Execute SE38 ZABAPGIT



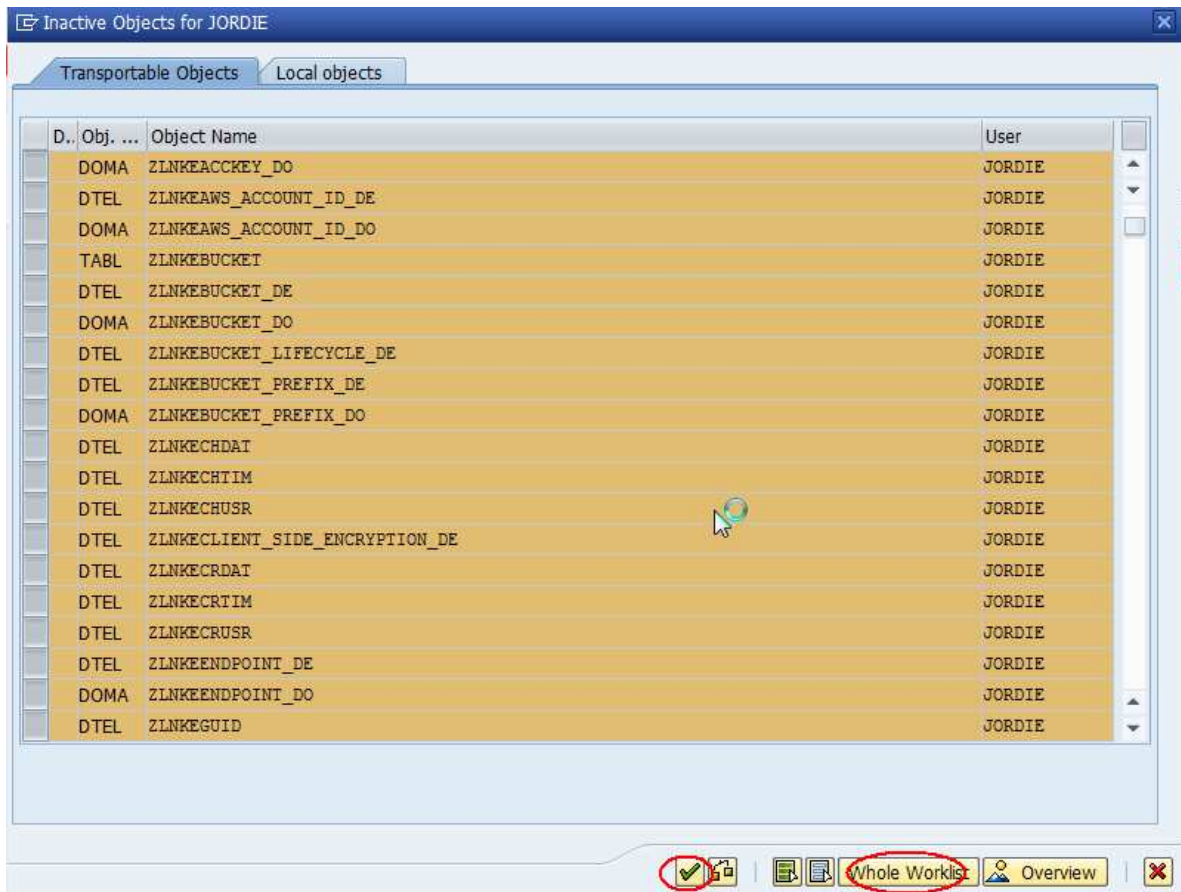
Git clone URL: https://github.com/LinkeIT/AWS_S3_SDK_for_ABAP_Z.git

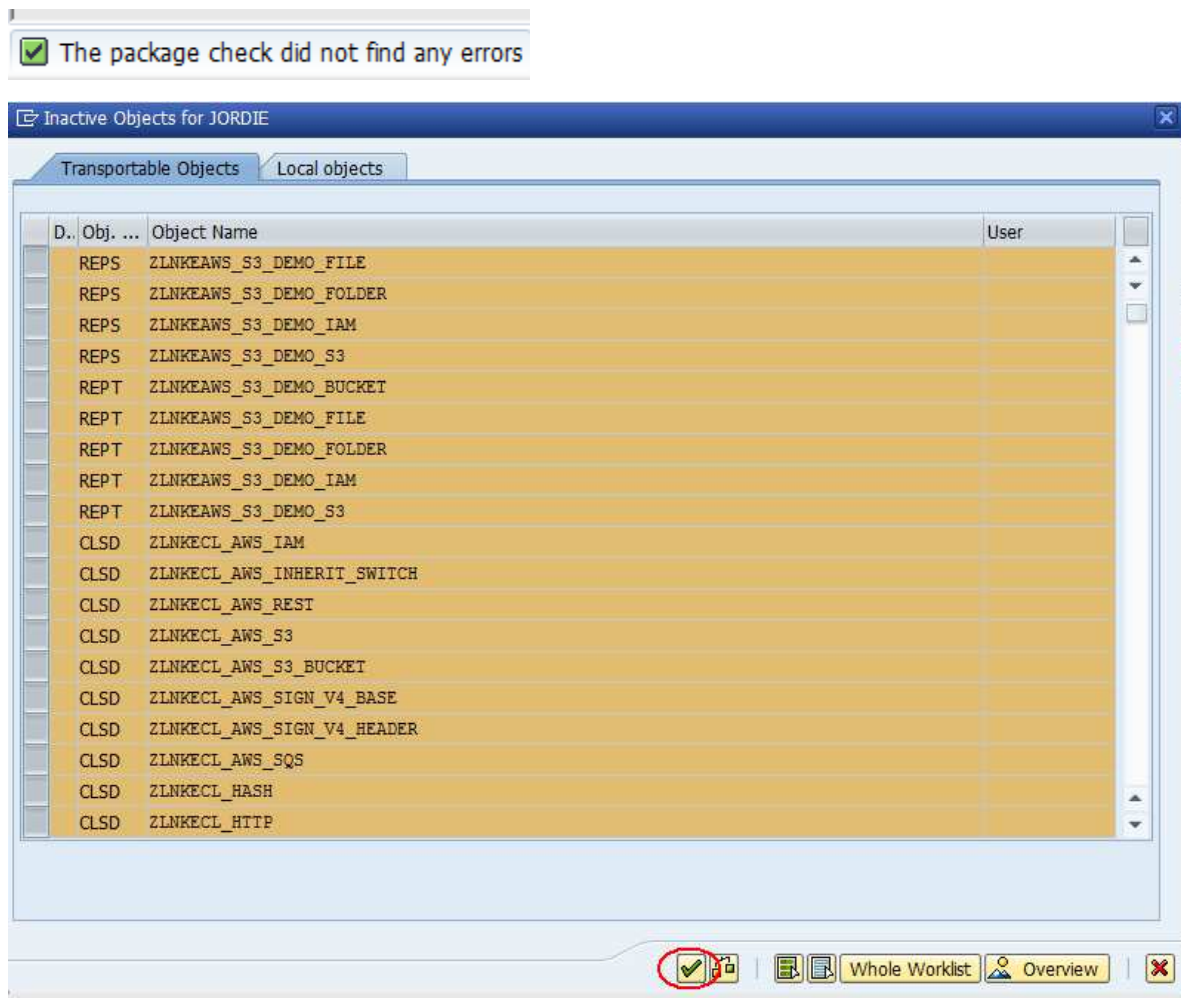
Target Package: ZLNKEAWS



First subpackage to import will be ZLNKEAWS_S3_SDK. Give transport request for every prompt (expect around 68 dialogs).

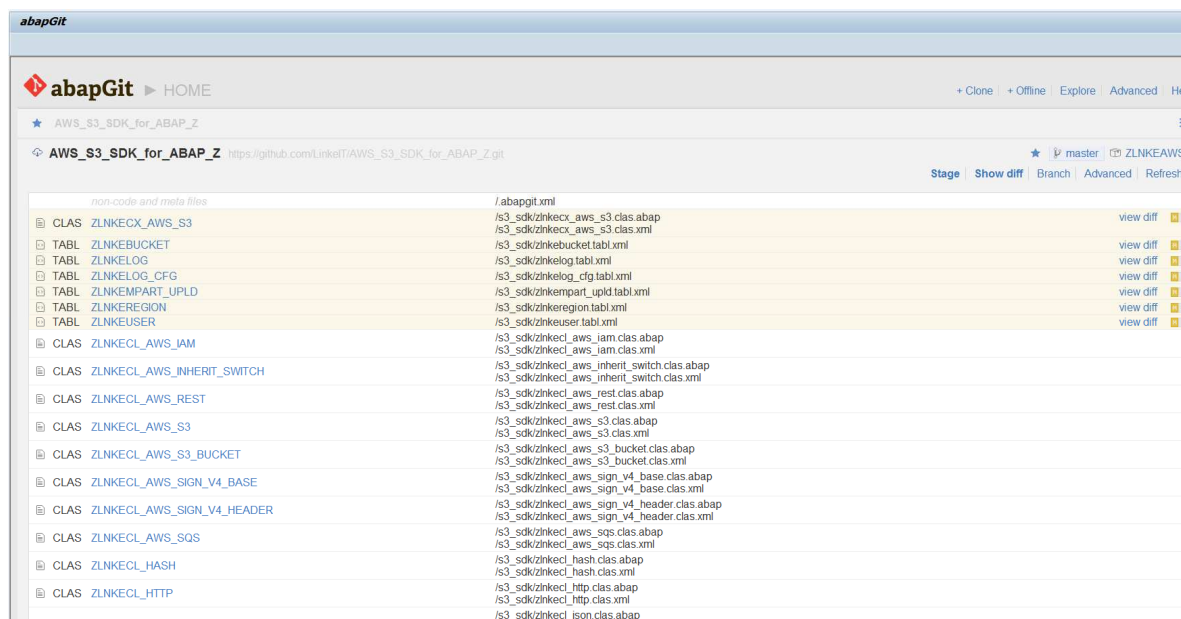
Activate Whole Worklist





Second subpackage to import will be ZLNKEAWS_S3_DEMO_PROGRAMS. Give transport request for every prompt (expect around 5 dialogs).

abapGit will show imported objects



Once installed in SE80 you should see this tree:

▼ ZLNKEAWS	ABAP AWS tools
▼ Subpackages	
▼ ZLNKEAWS_S3_DEMO_PROGRAMS	ABAP AWS S3 Demo programs
▼ Programs	
▶ ZLNKEAWS_S3_DEMO_BUCKET	Demo Create / Delete Bucket
▶ ZLNKEAWS_S3_DEMO_FILE	Demo working with Files
▶ ZLNKEAWS_S3_DEMO_FOLDER	Demo working with Folders
▶ ZLNKEAWS_S3_DEMO_IAM	Demo creation for a IAM user
▶ ZLNKEAWS_S3_DEMO_S3	Demo S3 List Buckets
▼ ZLNKEAWS_S3_SDK	ABAP AWS SDK Version Z
▼ Dictionary Objects	
▶ Database Tables	
▶ Table Types	
▶ Structures	
▶ Data Elements	
▶ Domains	
▼ Class Library	
▼ Classes	
▶ ZLNKECL_AWS_IAM	RocketSteam. Identity & Access Management
▶ ZLNKECL_AWS_INHERIT_SW	RocketSteam. Inheritance Switch
▶ ZLNKECL_AWS_REST	RocketSteam. AWS Rest
▶ ZLNKECL_AWS_S3	RocketSteam. S3 Service
▶ ZLNKECL_AWS_S3_BUCKET	RocketSteam. Bucket
▶ ZLNKECL_AWS_SIGN_V4_BA	RocketSteam. Common methods and properties
▶ ZLNKECL_AWS_SIGN_V4_HE	RocketSteam. AWS V4 Signer, requests with Aut
▶ ZLNKECL_AWS_SQS	RocketSteam. AWS SQS class
▶ ZLNKECL_HASH	RocketSteam. Hash class
▶ ZLNKECL_HTTP	RocketSteam. HTTP protocol constants
▶ ZLNKECL_JSON	Copy of standard class /UI2/CL_JSON
▶ ZLNKECL_LOG	RocketSteam. Log class
▶ ZLNKECL_RFC_CONNECTION	RocketSteam. SM59 tools
▶ ZLNKECL_SSF	RocketSteam. Secure Store & Forward
▶ ZLNKECL_STRING_CONVERS	RocketSteam. String conversions
▶ ZLNKECL_XML_UTILS	RocketSteam. XML Utilities
▶ ZLNKECX_AWS_S3	RocketSteam. Exception class
▼ Interfaces	
▼ Programs	
▶ ZLNKERS3_LOG_VIEWER	RocketSteam. RS3 Log Viewer
▶ ZLNKERS3_PURGE_MPART_UPLD	RocketSteam. Abort Multipart Upload
▶ ZLNKERS3_STRUCT	RocketSteam. STRUCT automatic maintenance f

Initial load of tables

Tables ZLNKERECTION and ZLNKELOG_CFG needs initial load.

To do so, run SE38 ZLNKERS3_TABLES_INITIAL_LOAD (unckeck test run)

Initial load for AWS Regions and log configuration



☐ Test run

Initial load for AWS Regions and log configuration

Initial load for AWS Regions and log configuration

Table LNKEREGION initial load success
Table ZLNKELOG_CFG initial load success

After initial load table should ZLNKEREGION looks like:

Data Browser: Table ZLNKEREGION Select Entries 14

Table: ZLNKEREGION
Displayed Fields: 3 of 3 Fixed Columns: 1 List Width 0250

REGION	REGION_NAME	ENDPOINT
ap-northeast-1	Asia Pacific (Tokyo)	s3-ap-northeast-1.amazonaws.com
ap-northeast-2	Asia Pacific (Seoul)	s3-ap-northeast-2.amazonaws.com
ap-south-1	Asia Pacific (Mumbai)	s3-ap-south-1.amazonaws.com
ap-southeast-1	Asia Pacific (Singapore)	s3-ap-southeast-1.amazonaws.com
ap-southeast-2	Asia Pacific (Sydney)	s3-ap-southeast-2.amazonaws.com
ca-central-1	Canada (Central)	s3-ca-central-1.amazonaws.com
eu-central-1	EU (Frankfurt)	s3-eu-central-1.amazonaws.com
eu-west-1	EU (Ireland)	s3-eu-west-1.amazonaws.com
eu-west-2	EU (London)	s3-eu-west-2.amazonaws.com
sa-east-1	South America (Sao Paulo)	s3-sa-east-1.amazonaws.com
us-east-1	US Standard *	s3.amazonaws.com
us-east-2	US East (Ohio)	s3-us-east-2.amazonaws.com
us-west-1	US West (Northern California)	s3-us-west-1.amazonaws.com
us-west-2	US West (Oregon)	s3-us-west-2.amazonaws.com

After initial load table ZLNKELOG_CFG looks like:

Data Browser: Table ZLNKELOG_CFG Select Entries 1

Table: ZLNKELOG_CFG
Displayed Fields: 5 of 5 Fixed Columns: 1 List Width 0250

DUMMYID	KEEP_DAYS	LOG_XML_RESP_AWS	LOG_REQ_RESP_AWS	LOG_CONTSERV_REQ
	3			

STRUST

AWS S3 uses HTTPS protocol. In order to be able to communicate with AWS S3 endpoints the system must have a proper SSL certificate for each endpoint.

In SAP SSL certificates are installed in transaction STRUST.

To get AWS SSL certificates it is used OpenSSL from Operating System.

SSL certificates have a validity period, typically one year. Passed this period the certificate is not valid and cannot be used anymore.

AWS may invalidate any SSL certificate at any moment, even if the validity period is not expired. If this happens, the certificate cannot be used anymore and the communication with the endpoint will fail.

Manually maintaining SSL certificates may be a hard task. To ease this task we provide an automation tool, the program ZLNKERS3_STRUST.

To learn more read

http://help.sap.com/saphelp_nw73ehp1/helpdata/en/4c/5b218c980a7514e10000000a42189b/content.htm

Program ZLNKERS3_STRUST

The system must be configured to add to STRUST in SSL client the AWS certificates.

This is automatically done with program ZLNKERS3_STRUST.

A preparation must be done prior running ZLNKERS3_STRUST. Run transaction SM69. If the operating system is Linux: Create a new command:

The image shows two screenshots from the SAP SM69 transaction. The top screenshot displays the 'External Operating System Commands' table with columns: Type, Command name, Op.system, Prog. name, Parameters of external program, Additional, Trace, Created By, and Current D. It lists three commands: ARCAUTO, BACKUP_HISTORY, and BRARCHIVE. The bottom screenshot shows the 'Create an External Command' dialog. In this dialog, the 'Command Name' field is set to 'ZOPENSSL', the 'Operating System' is 'Linux', and the 'Definition' section shows 'Operating System Command' set to 'openssl'. The 'Additional Parameters Allowed' checkbox is checked, and the 'Trace' checkbox is unchecked.

Type	Command name	Op.system	Prog. name	Parameters of external program	Additional	Trace	Created By	Current D
SAP	ARCAUTO	ANYOS	arcauto		X		SAP	07.08.19
SAP	BACKUP_HISTORY	ANYOS	sddb6his		X		SAP	08.11.19
SAP	BRARCHIVE	ANYOS	brarchive		X		SAP	02.08.19

Create an External Command

Command

Command Name: ZOPENSSL

Operating System: Linux

Type:

Create and Last Change

Created By:

Last Changed By:

Definition

Operating System Command: openssl

Parameters for Operating System Command:

☒ Additional Parameters Allowed

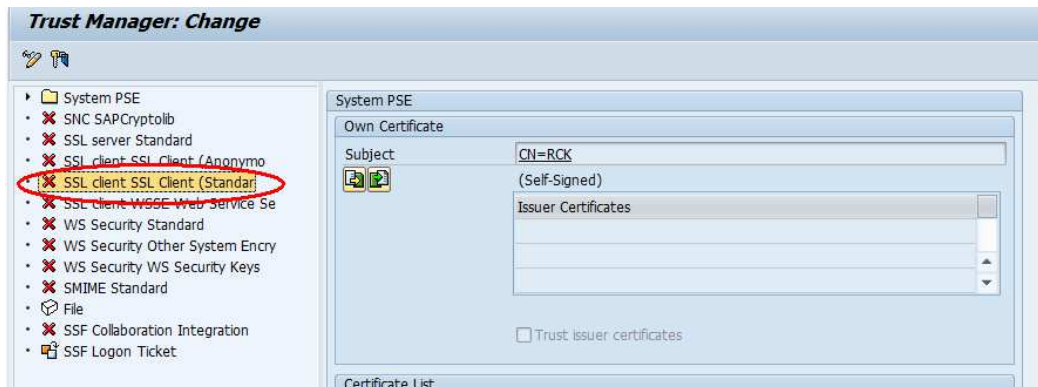
☐ Trace

Check Module:

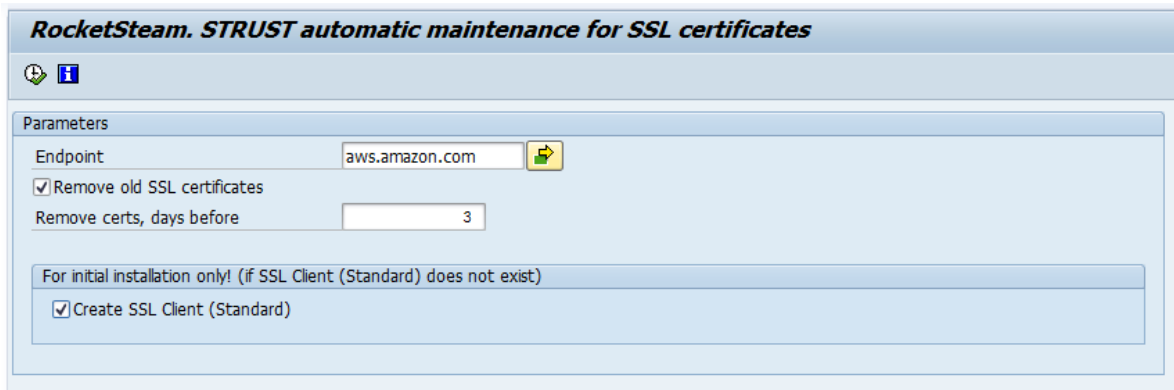
Note: openssl must be installed on linux

If your operating system is Windows install openssl and act in the same way.

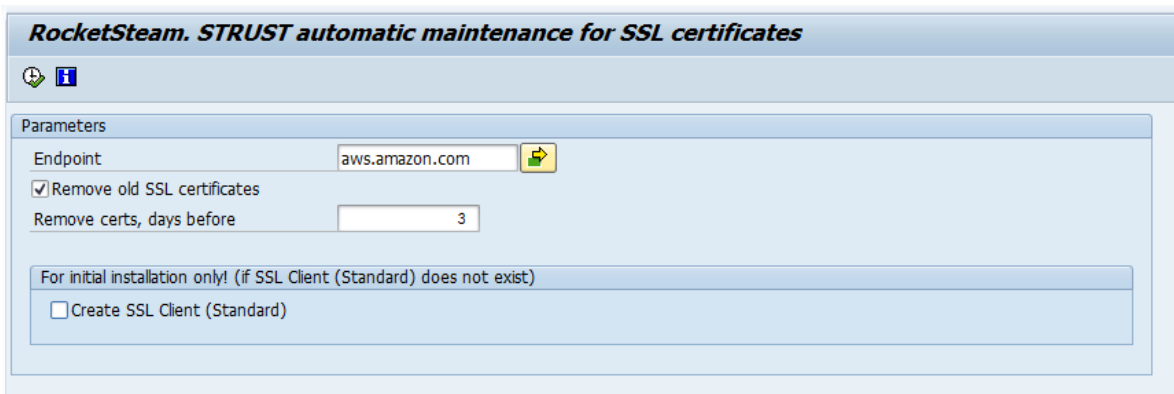
Go to transaction STRUST and check if SSL client SSL Client (standard) exists (in the sample screenshot is not existing):



Run report (SE38) ZLNKERS3_STRUST with these parameters if SSL client SSL Client (standard) does not exist



Run report (SE38) ZLNKERS3_STRUST with these parameters if SSL client SSL Client (standard) already exists:



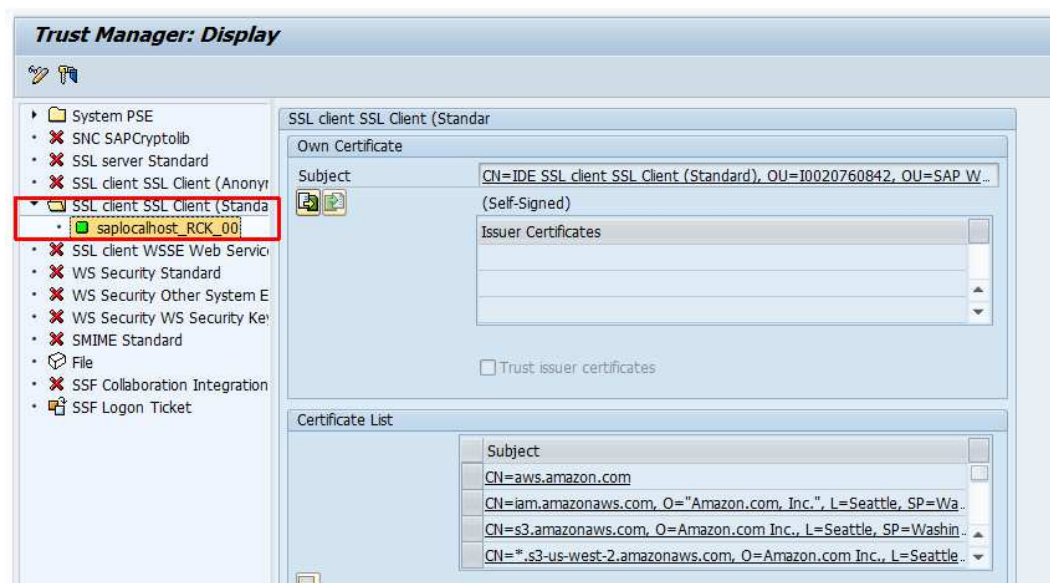
Expect to have this result:

```
RocketSteam. STRUST automatic maintenance for SSL certificates

RocketSteam. STRUST automatic maintenance for SSL certificates

PSE for SSL Client (Standard) created
Certificate import success for aws.amazon.com
Certificate import success for s3-ap-northeast-1.amazonaws.com
Certificate import success for s3-ap-southeast-1.amazonaws.com
Certificate import success for s3-ap-southeast-2.amazonaws.com
Certificate import success for s3.eu-central-1.amazonaws.com
Certificate import success for s3-eu-west-1.amazonaws.com
Certificate import success for s3-sa-east-1.amazonaws.com
Certificate import success for s3.amazonaws.com
Certificate import success for s3-us-west-1.amazonaws.com
Certificate import success for s3-us-west-2.amazonaws.com
Certificate import success for iam.amazonaws.com
ICM restarted
```

You can check the result in transaction STRUST:



Note: In case some exception occurs it is shown in red, for example:

```
RocketSteam. STRUST automatic maintenance for AWS certificates

RocketSteam. STRUST automatic maintenance for AWS certificates

Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
Exception!: Function Module SSFPSE_ENQUEUE raised exception FOREIGN_LOCK
```

In this case the exception is due to STRUST foreign lock (i.e. someone is editing in transaction STRUST).

Programming job ZLNKERS3_STRUST

AWS certificates are invalidated or are expired from time to time.

If this happens, new certificate(s) must be installed in STRUST to ensure S3 for SAP can run properly.

The program ZLNKERS3_STRUST will take care of it.

We recommend programming a job ZLNKERS3_STRUST on a daily basis.

To do so, go to transaction SM36:

Note: Please replace /RS3/RS3_STRUST by ZLNKERS3_STRUST

Define Background Job

Start Condition **Step** Job Selection Own Jobs Job Wizard Standard Jobs

General data

Job name **/RS3/RS3_STRUST**

Job class C

Status Scheduled

Exec. Target **Spool list recipient**

User

Program values

ABAP program External command External program

ABAP program

Name **/RS3/RS3_STRUST**

Variant

Language EN

External command (command pre-defined by system administrator)

Name

Parameters

Operating sys.

Target server

External program (direct command input by system administrator)

Name

Parameter

Target host

☒ Check ☐ Print Specifications ☐

Step List Overview

No.	Program name/command	Prog. type	Spool list	Parameters	User	Lang.
1	/RS3/RS3_STRUST	ABAP			LINKEIT	EN

Define Background Job

Start Condition Step Job Selection Own Jobs Job Wizard Standard Job

General data

Job name /RS3/RS3_STRUST

Job class C

Status Scheduled

Exec. Target

Spool list recipient

Start Time

Immediate Date/Time After job After event At operation mode

Date/Time

After job

At operation mode

After event

Check

Inform the next day and the time you wish this job to run (the job takes a few seconds to execute, light workload).

Start Time

Immediate Date/Time After job After event At operation mode

Date/Time

Scheduled start Date 09.04.2016 Time 08:00

No Start After Date Time

After job At operation mode

After event

☐ Periodic job

Check Save **Period Values** Restrictions Close

Choose daily period

Period Values

Hourly Daily Weekly Monthly Other period

Check Save Close

Save

Start Time

Immediate Date/Time After job After event At operation mode

Date/Time

Scheduled start Date 09.04.2016 Time 08:00:00

No Start After Date Time

After job

At operation mode

After event

☒ Periodic job

Check Period Values Restrictions

And save

Job Edit Goto System Help

Define Background Job

Start Condition Step Job Selection Own Jobs Job Wizard Standard Jobs

General data

Job name /RS3/RS3_STRUST

Job class C

Status Scheduled

Exec. Target

Spool list recipient

☒ Job /RS3/RS3_STRUST saved with status: Released

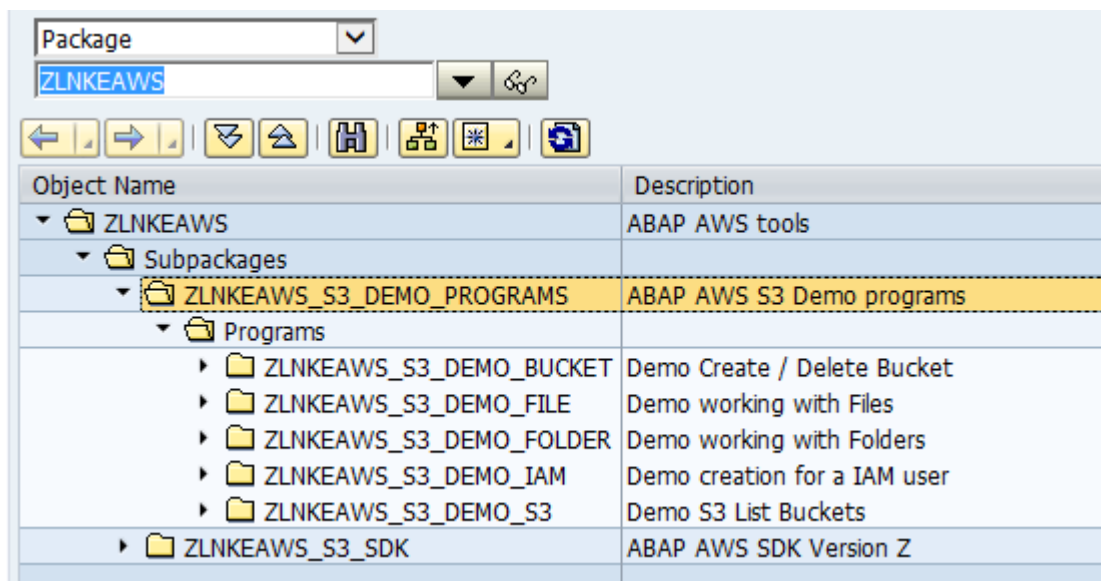
Log viewer

The solution logs every operation according to configuration in table ZLNKELOG_CFG. Feel free to play with the flags.

Run program ZLNKERS3_LOG_VIEWER to view log entries.

Demo programs

Demo programs are provided under the package ZLNKEAWS_S3_DEMO_PROGRAMS.



The screenshot shows the SAP Object Navigator with the package ZLNKEAWS selected. The hierarchy is as follows:

Object Name	Description
Package ZLNKEAWS	ABAP AWS tools
Subpackages	
Package ZLNKEAWS_S3_DEMO_PROGRAMS	ABAP AWS S3 Demo programs
Programs	
Package ZLNKEAWS_S3_DEMO_BUCKET	Demo Create / Delete Bucket
Package ZLNKEAWS_S3_DEMO_FILE	Demo working with Files
Package ZLNKEAWS_S3_DEMO_FOLDER	Demo working with Folders
Package ZLNKEAWS_S3_DEMO_IAM	Demo creation for a IAM user
Package ZLNKEAWS_S3_DEMO_S3	Demo S3 List Buckets
Package ZLNKEAWS_S3_SDK	ABAP AWS SDK Version Z

These programs can be used to test the initial setup and are intended to be a reference for developing your own applications.

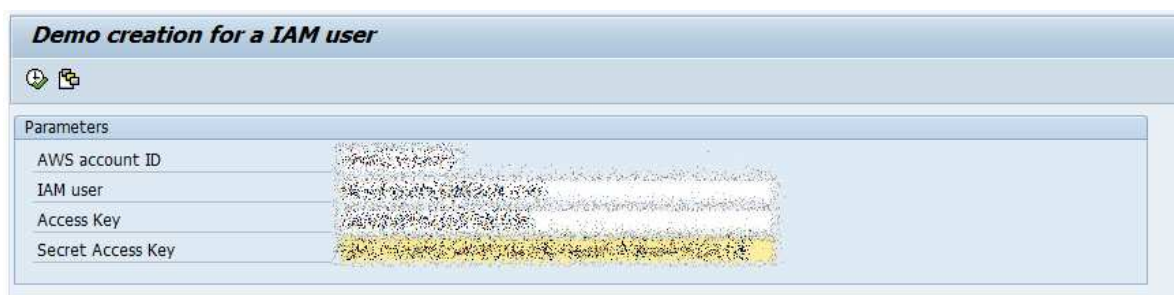
Note: Where you see /RS3/ in screenshots understand ZLNKE

Program ZLNKEAWS_S3_DEMO_IAM

To use S3 for SAP you need to create an IAM user on your SAP system. This IAM user must already been created on AWS, and must have proper permissions.

Demo program ZLNKEAWS_S3_DEMO_IAM shows how to create the IAM user in your SAP system.

Fill your credentials and run the program



The screenshot shows a dialog box titled "Demo creation for a IAM user". It contains a "Parameters" section with the following fields:

Parameter	Value
AWS account ID	123456789012
IAM user	demo-user
Access Key	AKIAIOSFODNN7EXAMPLE
Secret Access Key	wJalrXU3WhJOdzBsCZ3wPbWVnRQz1w1Yw2PYe9f2X

On success, expect to have this result:

Demo creation for a IAM user
Demo creation for a IAM user
IAM user insert success in table /RS3/USER

In case an exception occurs, for example:

Demo creation for a IAM user
Demo creation for a IAM user
User validation failed:User: arn:aws:iam::444444444444:user is not authorized to perform: iam:GetUser on resource: user arn:aws:iam::444444444444:user

Double check:

- Your AWS account ID
- Your IAM user
- The Attached Policies to the IAM user

From now you can start operating on buckets by using this IAM user.

The IAM user is inserted on database table ZLNKEUSER.

Technical explanation

Local class lcl_iam_demo has the method execute.

```
METHOD execute.  
DATA: lv_msg TYPE string.  
DATA: lv_user_name TYPE /rs3/username_de.  
DATA: lv_aws_account_id TYPE /rs3/aws_account_id_de.  
DATA: lv_access_key TYPE /rs3/acckey_de.  
DATA: lv_secret_access_key TYPE /rs3/secacckey_de.  
DATA: lv_user_id TYPE string. "SEC NEEDED  
DATA: ls_rs3_user TYPE /rs3/user.  
DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.  
  
TRY.  
    IF /rs3/cl_rfc_connections=>http_dest_to_ext_exists_iam( ) = abap_false.  
        /rs3/cl_rfc_connections=>create_http_dest_to_ext_iam( ).  
        WRITE:/ 'Created AWS destination for IAM endpoint'.  
    ENDIF.  
  
    lv_user_name = p_iam.  
    lv_aws_account_id = p_aws.  
    lv_access_key = p_key.  
    lv_secret_access_key = p_seckey.  
    CALL METHOD /rs3/cl_aws_iam=>check_aws_user  
        EXPORTING  
            i_user_name = lv_user_name  
            i_aws_account_id = lv_aws_account_id  
            i_access_key = lv_access_key  
            i_secret_access_key = lv_secret_access_key  
        RECEIVING  
            e_user_id = lv_user_id.  
  
    ls_rs3_user-user_name = lv_user_name.  
    ls_rs3_user-access_key = lv_access_key.  
    ls_rs3_user-secr_access_key = lv_secret_access_key.  
    ls_rs3_user-aws_account_id = lv_aws_account_id.  
    ls_rs3_user-crusr = sy-uname.  
    ls_rs3_user-crdat = sy-datum.  
    ls_rs3_user-crtim = sy-uzeit.  
    INSERT /rs3/user FROM ls_rs3_user.  
    IF sy-subrc = 0.  
        WRITE:/ 'IAM user insert success in table /RS3/USER'.  
    ENDIF.  
  
    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.  
        lv_msg = lr_cx_aws_s3->get_text( ).  
        WRITE:/ lv_msg.  
    ENDTRY.
```

The static method ZLNKECL_AWS_IAM=>CHECK_AWS_USER is called prior inserting the user on table ZLNKEUSER.

Any exception which may arise, for example if the user does not exist will be caught and shown the exception text.

Program ZLNKEAWS_S3_DEMO_BUCKET

Demo program ZLNKEAWS_S3_DEMO_BUCKET shows how to operate on Buckets.

Possible operations are:

- Create Bucket
- Delete Bucket
- List Bucket
- List Bucket Location
- Head Bucket

Selection screen:

Demo Create / Delete Bucket

Parameters

☒ Create Bucket
☐ Delete Bucket
☐ List Bucket
☐ Bucket location
☐ Head Bucket
☐ Write headers
Bucket name
IAM user
AWS Region
☐ Bucket already exists on AWS

Note1: If you already have a bucket created on AWS, just fill the bucket name, IAM user and AWS region where the bucket exists and mark the flag "Bucket already exists on AWS".

Note2: In order to protect the buckets from cross reading / writing from development systems and production systems, the SID is concatenated in front of the bucket name, in lower case and is created with this name in AWS.

Demo Create / Delete Bucket

Parameters

☐ Create Bucket
☒ Delete Bucket
☐ List Bucket
☐ Bucket location
☐ Head Bucket
☐ Write headers
Bucket name

Write headers parameter is to see the request and response headers.

Technical explanation

Each operation is implemented in a static method of the local class lcl_demo_bucket.

create_bucket

The static method zlnkecl_aws_s3_bucket=>create_bucket is called to create a Bucket on AWS. On success it returns an instance of the Bucket created.

```
METHOD create_bucket.
  DATA: lv_xml TYPE string.
  DATA: lv_msg TYPE string.
  DATA: lv_http_status TYPE i.
  DATA: ls_/rs3/bucket TYPE /rs3/bucket.
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.      "#EC NEEDED
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

  TRY.
    CALL METHOD /rs3/cl_aws_s3_bucket=>create_bucket
      EXPORTING
        i_bucket_name      = p_bucket
        i_user_name        = p_iam
        i_region           = p_region
        i_dbg              = p_dbg
      IMPORTING
        e_http_status      = lv_http_status
        e_response_content = lv_xml
        e_aws_s3_bucket    = lr_bucket. "Reference to the bucket created

    IF lv_xml IS NOT INITIAL.
      /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
    ENDIF.

    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.
      ls_/rs3/bucket-bucket = p_bucket.
      ls_/rs3/bucket-user_name = p_iam.
      ls_/rs3/bucket-region = p_region.
      ls_/rs3/bucket-crusr = sy-uname.
      ls_/rs3/bucket-crdat = sy-datum.
      ls_/rs3/bucket-crtim = sy-uzeit.
      INSERT /rs3/bucket FROM ls_/rs3/bucket.
      CONCATENATE 'Bucket ' p_bucket ' created successfully'
        INTO lv_msg RESPECTING BLANKS.
    ELSE.
      CONCATENATE 'Bucket ' p_bucket ' could not be created'
        INTO lv_msg RESPECTING BLANKS.
    ENDIF.
    CONDENSE lv_msg.
    WRITE:/ lv_msg.

    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
      lv_msg = lr_cx_aws_s3->get_text( ).
      WRITE:/ lv_msg.
  ENDTRY.
```

Any exception which may arise will be caught and shown the exception text.

The bucket created is inserted in table ZLNKEBUCKET

create_bucket_only_db

If you already have your bucket created on AWS you use this method to create a register in table ZLNKEBUCKET

```
*-----*
* Creates a Bucket only on DB. Makes sense when your bucket
* is already existing on AWS.
*-----*
METHOD create_bucket_only_db.
  DATA: lv_msg TYPE string.
  DATA: lv_bucket TYPE /rs3/bucket-bucket.
  DATA: ls_/rs3/bucket TYPE /rs3/bucket.

  SELECT SINGLE bucket
        INTO lv_bucket
  FROM /rs3/bucket
  WHERE bucket = p_bucket.
  IF sy-subrc <> 0.
    ls_/rs3/bucket-bucket = p_bucket.
    ls_/rs3/bucket-user_name = p_iam.
    ls_/rs3/bucket-region = p_region.
    ls_/rs3/bucket-no_prefix = abap_true.
    ls_/rs3/bucket-crusr = sy-uname.
    ls_/rs3/bucket-crdat = sy-datum.
    ls_/rs3/bucket-crtim = sy-uzeit.
    INSERT /rs3/bucket FROM ls_/rs3/bucket.
    CONCATENATE 'Bucket ' p_bucket ' created successfully'
      INTO lv_msg RESPECTING BLANKS.
  ELSE.
    CONCATENATE 'Bucket ' p_bucket ' already exists in DB'
      INTO lv_msg RESPECTING BLANKS.
  ENDIF.
  CONDENSE lv_msg.
  WRITE:/ lv_msg.

ENDMETHOD.                                "create bucket only db
```

delete_bucket

The bucket object lr_bucket is instantiated giving the bucket name. After the method delete_bucket is called.

The bucket must be empty to be deleted.

On success the bucket is deleted from AWS and from table ZLNKEBUCKET.

```
*-----*
* Deletes a Bucket (must be empty)
*-----*
METHOD delete_bucket.
    DATA: lv_xml TYPE string.
    DATA: lv_msg TYPE string.
    DATA: lv_http_status TYPE i.
    DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
    DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

    TRY.
        CREATE OBJECT lr_bucket
            EXPORTING
                i_bucket_name = p_bucket
                i_dbg          = p_dbg.

        CALL METHOD lr_bucket->delete_bucket
            IMPORTING
                e_http_status      = lv_http_status
                e_response_content = lv_xml.

        IF lv_xml IS NOT INITIAL.
            /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
        ENDIF.

        IF lv_http_status = /rs3/cl_http=>c_status_204_no_content.
            DELETE FROM /rs3/bucket WHERE bucket = p_bucket.
            CONCATENATE 'Bucket ' p_bucket ' deleted successfully'
                INTO lv_msg RESPECTING BLANKS.
        ELSE.
            CONCATENATE 'Bucket ' p_bucket ' could not be deleted'
                INTO lv_msg RESPECTING BLANKS.
        ENDIF.
        CONDENSE lv_msg.
        WRITE:/ lv_msg.

        CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
        lv_msg = lr_cx_aws_s3->get_text( ).
        WRITE:/ lv_msg.
    ENDTRY.

ENDMETHOD.                                "delete_bucket
```

Any exception which may arise will be caught and shown the exception text.

list_bucket

The bucket object lr_bucket is instantiated giving the bucket name. After the method list_objects is called. It returns an XML containing the list, limited to a maximum of 1000 entries. You can use i_prefix parameter to filter the list by a prefix.

Parameter i_marker is used for paging in case you want to get more than 1000 entries.

Parameter i_max_keys is used if you wish to limit to a lower number of entries.

```
*-----*
* Lists Bucket content
*-----*

METHOD list_bucket.
    DATA: lv_xml TYPE string.
    DATA: lv_msg TYPE string.
    DATA: lv_http_status TYPE i.                                "#EC NEEDED
    DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
    DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

    TRY.
        CREATE OBJECT lr_bucket
            EXPORTING
                i_bucket_name = p_bucket
                i_dbg          = p_dbg.

        CALL METHOD lr_bucket->list_objects|

        EXPORTING
            i_prefix          =
            i_marker          =
            i_max_keys        =
        IMPORTING
            e_http_status     = lv_http_status
            e_response_content = lv_xml.

        IF lv_xml IS NOT INITIAL.
            /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
        ENDIF.

        CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
            lv_msg = lr_cx_aws_s3->get_text( ).
            WRITE:/ lv_msg.
        ENDTRY.
    ENDMETHOD.                                "list_bucket
```

Any exception which may arise will be caught and shown the exception text.

bucket_location

The bucket object lr_bucket is instantiated giving the bucket name. After the method get_bucket_location is called. It returns an XML containing the AWS region where the Bucket is located

```
*-----*
* Shows Bucket location
*-----*

METHOD bucket_location.
    DATA: lv_xml TYPE string.
    DATA: lv_msg TYPE string.
    DATA: lv_http_status TYPE i.                                "#EC NEEDED
    DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
    DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

    TRY.
        CREATE OBJECT lr_bucket
            EXPORTING
                i_bucket_name = p_bucket
                i_dbg          = p_dbg.

        CALL METHOD lr_bucket->get_bucket_location
            IMPORTING
                e_http_status      = lv_http_status
                e_response_content = lv_xml.

        IF lv_xml IS NOT INITIAL.
            /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
        ENDIF.

        CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
            lv_msg = lr_cx_aws_s3->get_text( ).
            WRITE:/ lv_msg.
        ENDTRY.

    ENDMETHOD.                                "bucket_location
```

Any exception which may arise will be caught and shown the exception text.

head_bucket

This can be used to check that the bucket exists.

The bucket object lr_bucket is instantiated giving the bucket name. After the method head_bucket is called. It will return HTTP Status.

```
*-----*
* Head
*-----*
METHOD head_bucket.
    DATA: lv_msg TYPE string.
    DATA: lv_http_status TYPE i.
    DATA: lt_response_headers TYPE tihttpnvp.           "#EC NEEDED
    DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
    DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

    TRY.
        CREATE OBJECT lr_bucket
            EXPORTING
                i_bucket_name = p_bucket
                i_dbg         = p_dbg.

        CALL METHOD lr_bucket->head_bucket
            IMPORTING
                e_http_status      = lv_http_status
                e_response_headers = lt_response_headers.

        lv_msg = /rs3/cl_http=>get_reason_by_status( lv_http_status ).
        WRITE:/ lv_http_status, lv_msg.

    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
        lv_msg = lr_cx_aws_s3->get_text( ).
        WRITE:/ lv_msg.

    ENDTRY.
ENDMETHOD.           "head_bucket
```

Any exception which may arise will be caught and shown the exception text.

Program ZLNKEAWS_S3_DEMO_FILE

Demo program ZLNKEAWS_S3_DEMO_FILE shows how to operate on files.

Possible operations are:

- Put file
- Delete file
- Get file
- Head file

Selection screen:

The image displays two screenshots of a web application titled "Demo working with Files". Both screenshots show a "Parameters" section with radio buttons for "Put file", "Delete file", "Get file", and "Head file", and a checkbox for "Write headers".

The top screenshot shows "Put file" selected. It has input fields for "Bucket name" and "Folder".

The bottom screenshot shows "Get file" selected. It has input fields for "Bucket name", "Folder", and "File name".

Write headers parameter is to see the request and response headers.

Technical explanation

Each operation is implemented in a static method of the local class lcl_demo_file.

put_file

The method select_and_get_file_bin is called to show a file select dialog. Once the file is selected it is read and the content is set in lv_content.

Filename and folder are escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method put_object is called, giving the file name and content.

```
METHOD put_file.
  DATA: lv_filename TYPE string,
         lv_folder   TYPE string.
  DATA: lv_content  TYPE xstring.
  DATA: lv_msg      TYPE string.
  DATA: lv_xml      TYPE string.
  DATA: lv_http_status TYPE i.
  DATA: lr_bucket   TYPE REF TO /rs3/cl_aws_s3_bucket.
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

  TRY.
    select_and_get_file_bin( IMPORTING ex_filename = lv_filename
                             ex_content   = lv_content ).

*   Escape for considering special characters in file name
    lv_filename = /rs3/cl_http=>escape_url( lv_filename ).
    IF p_folder IS NOT INITIAL.
      lv_folder = /rs3/cl_http=>escape_url( p_folder ).
      CONCATENATE lv_folder '/' lv_filename INTO lv_filename.
    ENDIF.

    CREATE OBJECT lr_bucket
      EXPORTING
        i_bucket_name = p_bucket
        i_dbg          = p_dbg.

    CALL METHOD lr_bucket->put_object
      EXPORTING
        i_object_name      = lv_filename
        i_xcontent         = lv_content
        i_escape_url       = abap_false
      IMPORTING
        e_http_status      = lv_http_status
        e_response_content = lv_xml.

    IF lv_xml IS NOT INITIAL.
      /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
    ENDIF.

    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.
      CONCATENATE 'File ' lv_filename ' created successfully'
        INTO lv_msg RESPECTING BLANKS.
    ELSE.
      CONCATENATE 'File ' lv_filename ' could not be created'
        INTO lv_msg RESPECTING BLANKS.
    ENDIF.
  CATCH cx_aws_s3 EXCEPTION.
    lv_msg = cx_aws_s3->get_text( ).
  ENDTRY.
```

Any exception which may arise will be caught and shown the exception text.

delete_file

Filename and folder are escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method head_object is called, giving the file name. If the file exists the method delete_object is called. HTTP 204 No content is returned on success.

```
METHOD delete_file.  
  DATA: lv_filename TYPE string,  
        lv_folder TYPE string.  
  DATA: lv_msg TYPE string.  
  DATA: lv_xml TYPE string.  
  DATA: lv_http_status TYPE i.  
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.  
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.  
  
  TRY.  
*    Escape for considering special characters in file name  
    lv_filename = /rs3/cl_http=>escape_url( p_fname ).  
    IF p_folder IS NOT INITIAL.  
      lv_folder = /rs3/cl_http=>escape_url( p_folder ).  
      CONCATENATE lv_folder '/' lv_filename INTO lv_filename.  
    ENDIF.  
  
    CREATE OBJECT lr_bucket  
      EXPORTING  
        i_bucket_name = p_bucket  
        i_dbg          = p_dbg.  
  
    CALL METHOD lr_bucket->head_object  
      EXPORTING  
        i_object_name = lv_filename  
      IMPORTING  
        e_http_status = lv_http_status.  
  
    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.  
      CALL METHOD lr_bucket->delete_object  
        EXPORTING  
          i_object_name      = lv_filename  
        IMPORTING  
          e_http_status      = lv_http_status  
          e_response_content = lv_xml.  
  
      IF lv_xml IS NOT INITIAL.  
        /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).  
      ENDIF.  
  
      IF lv_http_status = /rs3/cl_http=>c_status_204_no_content.  
        CONCATENATE 'File ' lv_filename ' deleted successfully'  
          INTO lv_msg RESPECTING BLANKS.  
      ELSE.  
        CONCATENATE 'File ' lv_filename ' could not be deleted'  
          INTO lv_msg RESPECTING BLANKS.  
      ENDIF.  
    ENDIF.
```

Any exception which may arise will be caught and shown the exception text.

get_file

Filename and folder are escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method get_object is called, giving the file name. The file content is returned in lv_file_content (binary string). If the file is not existing, in lv_file_content is returned an XML with the error.

```
METHOD get_file.  
  DATA: lv_filename TYPE string,  
         lv_folder   TYPE string.  
  DATA: lv_msg TYPE string.  
  DATA: lv_xml TYPE string.  
  DATA: lv_file_content TYPE xstring.           "#EC NEEDED  
  DATA: lv_http_status TYPE i.  
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.  
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.  
  
  TRY.  
*    Escape for considering special characters in file name  
    lv_filename = /rs3/cl_http=>escape_url( p_fname ).  
    IF p_folder IS NOT INITIAL.  
      lv_folder = /rs3/cl_http=>escape_url( p_folder ).  
      CONCATENATE lv_folder '/' lv_filename INTO lv_filename.  
    ENDIF.  
  
    CREATE OBJECT lr_bucket  
      EXPORTING  
        i_bucket_name = p_bucket  
        i_dbg         = p_dbg.  
  
    CALL METHOD lr_bucket->get_object  
      EXPORTING  
        i_object_name = lv_filename  
      IMPORTING  
        e_http_status = lv_http_status  
        e_response_xcontent = lv_file_content.  "File content is returned here  
  
    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.  
      CONCATENATE 'File ' lv_filename ' retrieved successfully'  
        INTO lv_msg RESPECTING BLANKS.  
    ELSEIF lv_http_status = /rs3/cl_http=>c_status_404_not_found.  
      CONCATENATE 'File ' lv_filename ' not found'  
        INTO lv_msg RESPECTING BLANKS.  
  
      /rs3/cl_string_conversions=>xstring_to_string(  
        EXPORTING input = lv_file_content  
        IMPORTING output = lv_xml ).  
      IF lv_xml IS NOT INITIAL.  
        /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).  
      ENDIF.  
    ENDIF.  
    CONDENSE lv_msg.  
    WRITE:/ lv_msg.
```

Any exception which may arise will be caught and shown the exception text.

head_file

Filename and folder are escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method head_object is called, giving the file name. HTTP status is returned. File length is returned in HTTP response headers.

```
*-----*
* This shows how to get file information without retrieving file content
* File lenght comes in response headers
*-----*

METHOD head_file.
    DATA: lv_filename TYPE string,
           lv_folder TYPE string.
    DATA: lv_msg TYPE string.
    DATA: lv_http_status TYPE i.
    DATA: lt_response_headers TYPE tihttpnvp.           "#EC NEEDED
    DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
    DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

    TRY.
        *      Escape for considering special characters in file name
        lv_filename = /rs3/cl_http=>escape_url( p_fname ).
        IF p_folder IS NOT INITIAL.
            lv_folder = /rs3/cl_http=>escape_url( p_folder ).
            CONCATENATE lv_folder '/' lv_filename INTO lv_filename.
        ENDIF.

        CREATE OBJECT lr_bucket
        EXPORTING
            i_bucket_name = p_bucket
            i_dbg          = p_dbg.

        CALL METHOD lr_bucket->head_object
        EXPORTING
            i_object_name      = lv_filename
        IMPORTING
            e_http_status      = lv_http_status
            e_response_headers = lt_response_headers.

        lv_msg = /rs3/cl_http=>get_reason_by_status( lv_http_status ).
        WRITE:/ lv_http_status, lv_msg.

        CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
            lv_msg = lr_cx_aws_s3->get_text( ).
            WRITE:/ lv_msg.
        ENDTRY.
    ENDMETHOD.                                     "head_file
```

Any exception which may arise will be caught and shown the exception text.

Program ZLNKEAWS_S3_DEMO_FOLDER

Demo program ZLNKEAWS_S3_DEMO_FILE shows how to operate on folders.

Possible operations are:

- Put folder
- Delete folder
- Head folder

Selection screen:

Demo working with Folders

Parameters

☒ Put folder
☐ Delete folder
☐ Head folder
☐ Write headers

Bucket name

Folder

Write headers parameter is to see the request and response headers.

Technical explanation

Each operation is implemented in a static method of the local class `lcl_demo_folder`.

put_folder

The folder is escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method put_object is called, giving the folder name.

```
METHOD put_folder.
  DATA: lv_folder TYPE string.
  DATA: lv_msg TYPE string.
  DATA: lv_xml TYPE string.
  DATA: lv_http_status TYPE i.
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

  TRY.
    Escape for considering special characters in folder name
    lv_folder = /rs3/cl_http=>escape_url( p_folder ).
    CONCATENATE lv_folder '/' INTO lv_folder.

    CREATE OBJECT lr_bucket
      EXPORTING
        i_bucket_name = p_bucket
        i_dbg          = p_dbg.

    CALL METHOD lr_bucket->put_object
      EXPORTING
        i_object_name      = lv_folder
        i_escape_url       = abap_false
      IMPORTING
        e_http_status      = lv_http_status
        e_response_content = lv_xml.

    IF lv_xml IS NOT INITIAL.
      /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
    ENDIF.

    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.
      CONCATENATE 'Folder ' lv_folder ' created successfully'
        INTO lv_msg RESPECTING BLANKS.
    ELSE.
      CONCATENATE 'Folder ' lv_folder ' could not be created'
        INTO lv_msg RESPECTING BLANKS.
    ENDIF.
    CONDENSE lv_msg.
    WRITE:/ lv_msg.

    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.
      lv_msg = lr_cx_aws_s3->get_text( ).
      WRITE:/ lv_msg.
  ENDTRY.
```

Any exception which may arise will be caught and shown the exception text.

delete_folder

The folder is escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method head_object is called, giving the folder name. If the folder exists the method delete_object is called, giving the folder name. HTTP 204 No content is returned on success.

```
METHOD delete_folder.|
  DATA: lv_folder TYPE string.
  DATA: lv_msg TYPE string.
  DATA: lv_xml TYPE string.
  DATA: lv_http_status TYPE i.
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.

  TRY.
*    Escape for considering special characters in folder name
    lv_folder = /rs3/cl_http=>escape_url( p_folder ).
    CONCATENATE lv_folder '/' INTO lv_folder.

    CREATE OBJECT lr_bucket
      EXPORTING
        i_bucket_name = p_bucket
        i_dbg          = p_dbg.

    CALL METHOD lr_bucket->head_object
      EXPORTING
        i_object_name = lv_folder
      IMPORTING
        e_http_status = lv_http_status.

    IF lv_http_status = /rs3/cl_http=>c_status_200_ok.
      CALL METHOD lr_bucket->delete_object
        EXPORTING
          i_object_name = lv_folder
        IMPORTING
          e_http_status = lv_http_status
          e_response_content = lv_xml.

      IF lv_xml IS NOT INITIAL.
        /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).
      ENDIF.

      IF lv_http_status = /rs3/cl_http=>c_status_204_no_content.
        CONCATENATE 'Folder ' lv_folder ' deleted successfully'
          INTO lv_msg RESPECTING BLANKS.
      ELSE.
        CONCATENATE 'Folder ' lv_folder ' could not be deleted'
          INTO lv_msg RESPECTING BLANKS.
      ENDIF.
      CONDENSE lv_msg.
      WRITE:/ lv_msg.
```

Any exception which may arise will be caught and shown the exception text.

head_folder

The folder is escaped to consider special characters.

The bucket object lr_bucket is instantiated giving the bucket name. After the method head_object is called, giving the folder name. HTTP status is returned.

```
METHOD head_folder.  
  DATA: lv_folder TYPE string.  
  DATA: lv_msg TYPE string.  
  DATA: lv_xml TYPE string.  
  DATA: lv_http_status TYPE i.  
  DATA: lt_response_headers TYPE tihttpnvp.          "#EC NEEDED  
  DATA: lr_bucket TYPE REF TO /rs3/cl_aws_s3_bucket.  
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.  
  
  TRY.  
*    Escape for considering special characters in folder name  
    lv_folder = /rs3/cl_http=>escape_url( p_folder ).  
    CONCATENATE lv_folder '/' INTO lv_folder.  
  
    CREATE OBJECT lr_bucket  
      EXPORTING  
        i_bucket_name = p_bucket  
        i_dbg          = p_dbg.  
  
    CALL METHOD lr_bucket->head_object  
      EXPORTING  
        i_object_name      = lv_folder  
      IMPORTING  
        e_http_status      = lv_http_status  
        e_response_headers = lt_response_headers.  
  
    lv_msg = /rs3/cl_http=>get_reason_by_status( lv_http_status ).  
    WRITE:/ lv_http_status, lv_msg.  
  
    IF lv_xml IS NOT INITIAL.  
      /rs3/cl_xml_utils=>show_xml_in_dialog( lv_xml ).  
    ENDIF.  
  
    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.  
      lv_msg = lr_cx_aws_s3->get_text( ).  
      WRITE:/ lv_msg.  
    ENDTRY.  
  ENDMETHOD.          "head_folder
```

Any exception which may arise will be caught and shown the exception text.

Program ZLNKEAWS_S3_DEMO_S3

Demo program ZLNKEAWS_S3_DEMO_S3 shows how to operate on S3 service. It lists the Buckets owned by the AWS account ID.

Selection screen:



Write headers parameter is to see the request and response headers.

Technical explanation

The only one operation on S3 is list buckets. It is implemented on static method list_buckets of local class lcl_demo_s3.

An object type ZLNKECL_AWS_S3 is instantiated, giving the IAM user.

After the method get_service is called. It returns an XML containing the list of the buckets the IAM has rights to access to.

```
CLASS lcl_demo_s3 IMPLEMENTATION.  
  
METHOD list_buckets.  
  DATA: lr_s3 TYPE REF TO /rs3/cl_aws_s3.  
  DATA: lv_response_content TYPE string.  
  DATA: lr_cx_aws_s3 TYPE REF TO /rs3/cx_aws_s3.  
  DATA: lv_exception_text TYPE string.  
  TRY.  
    CREATE OBJECT lr_s3  
      EXPORTING  
        i_user_name = p_iam  
        i_dbg       = p_dbg.  
  
    CALL METHOD lr_s3->get_service  
      IMPORTING  
        e_response_content = lv_response_content.  
  
    CALL METHOD /rs3/cl_xml_utils=>show_xml_in_dialog  
      EXPORTING  
        i_xml = lv_response_content.  
  
    CATCH /rs3/cx_aws_s3 INTO lr_cx_aws_s3.  
      lv_exception_text = lr_cx_aws_s3->get_text( ).  
      WRITE:/ lv_exception_text.  
    ENDTRY.  
  
  ENDMETHOD.                                "execute  
  
ENDCLASS.                                  "lcl_demo_bucket IMPLEMENTATION
```

Any exception which may arise will be caught and shown the exception text.

Conclusion

You have installed and configured S3 for SAP, which is ready to be used.

You have demo programs as a reference for your developments.

In case you are interested in our services, as well as on the Commercial edition of AWS S3 for SAP feel free to contact with contact@linkeit.com

If you need support you can mail us at support@linkeit.com with subject "S3 for SAP"

You can also contact with Jordi Escoda, the developer of S3 for SAP at jordi.escoda@linkeit.com

We encourage downloading, installing and using AWS S3 SDK for ABAP Community edition.

Feel free to share and contribute, you can ask us in case you need any additional feature to improve the product.

Enjoy using S3 for SAP!