ABAP SDK Implementation guide for Azure Active Directory

Author: Microsoft SAP Team

Version: 1.0

Contents

Microsoft

What is Azure Active Directory?

Azure Active Directory (Azure AD) provides an easy way for businesses to manage identity and access, both in the cloud and on-premises. Your users can use the same work or school account for single sign-on to any cloud and on-premises web application. Users can use their favourite devices, including iOS, Mac OS X, Android, and Windows. An Organization can protect sensitive data and applications both on-premises and in the cloud with integrated multi-factor authentication ensuring secure local and remote access. Azure AD extends your on-premises directories so that information workers can use a single organizational account to securely and consistently access their corporate resources. Azure AD also offers comprehensive reports, analytics, and self-service capabilities to reduce costs and enhance security. The Azure AD SLA ensures that your business always runs smoothly and can be scaled to enterprise levels.

For more details on Azure Active directory, visit Microsoft Azure Active Directory

Prerequisites

Make sure you have installed ABAP SDK for Azure in your SAP system. Refer document 'ABAP SDK for Azure – GitHub' for more details, Visit https://github.com/Microsoft/ABAP-SDK-for-Azure
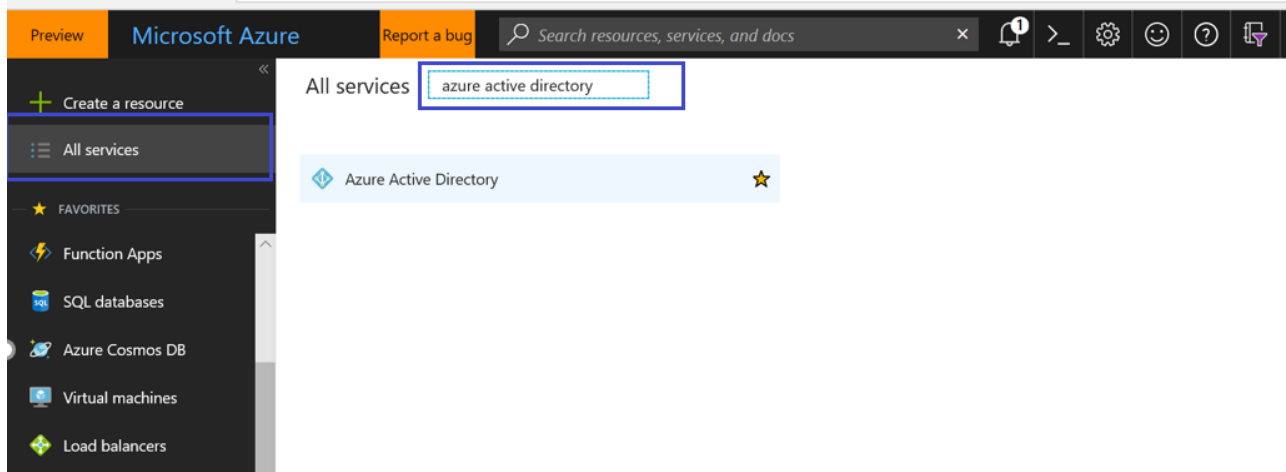
How to setup Azure Active Directory in Azure?

Login to Microsoft Azure portal.

Note: If you do not have an account already. please create a new Azure account. You can start free

Once you are logged into portal, go to all services and search for Azure Active Directory and Select

"Azure Active Directory" as shown below.

Create a new tenant for your organization in case it hasn't been created.

https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-access-create-new-tenant
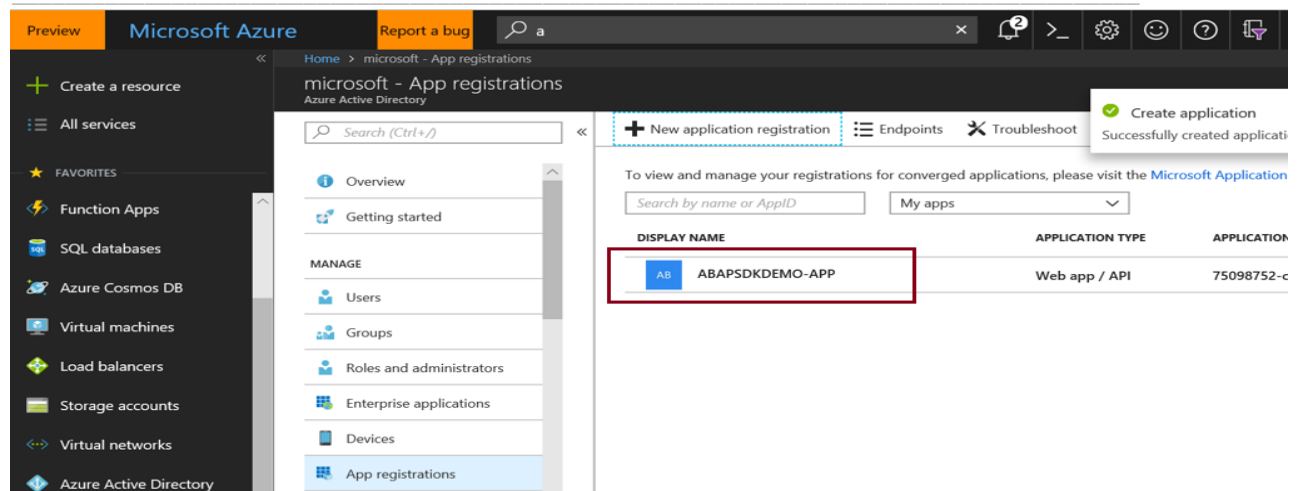
Click on 'App Registrations' on left side menu as shown below and click the button 'New application Registration'
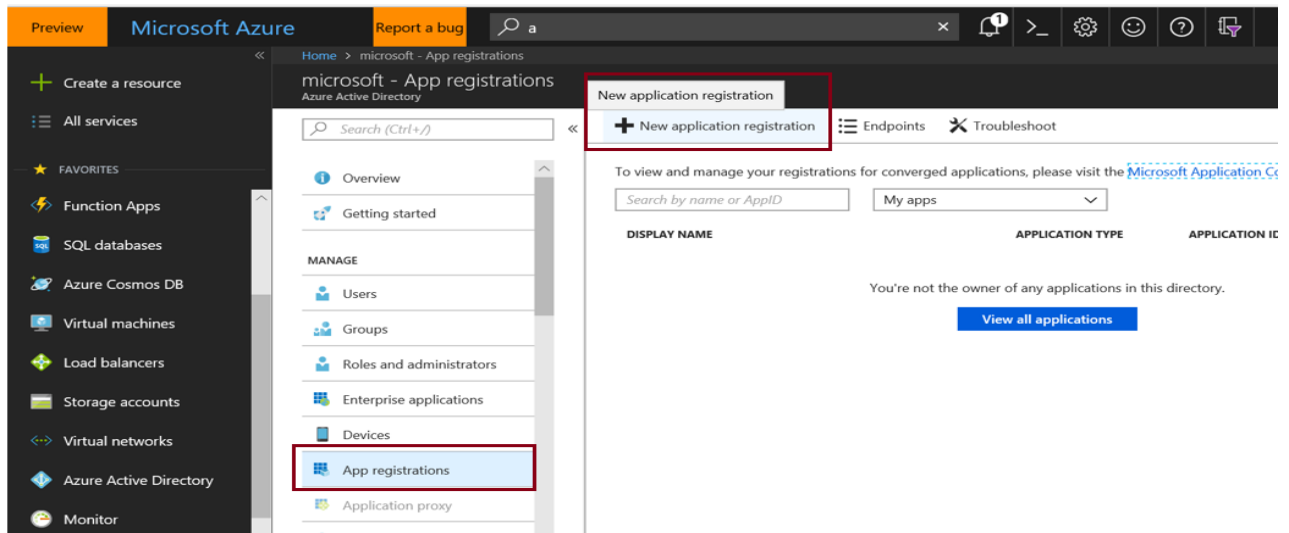
Specify details of your Application and press 'create' button.
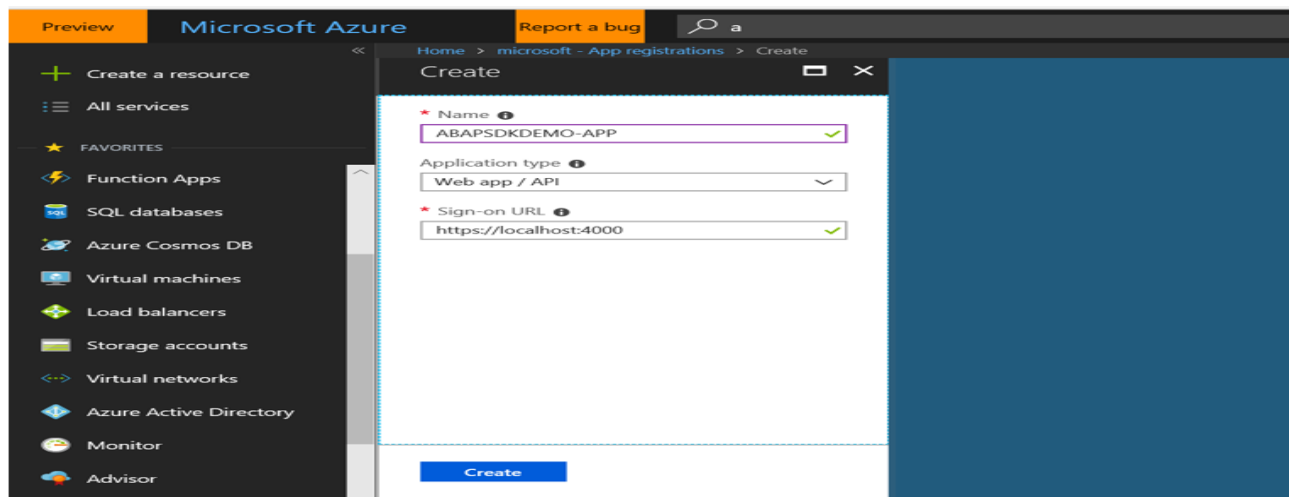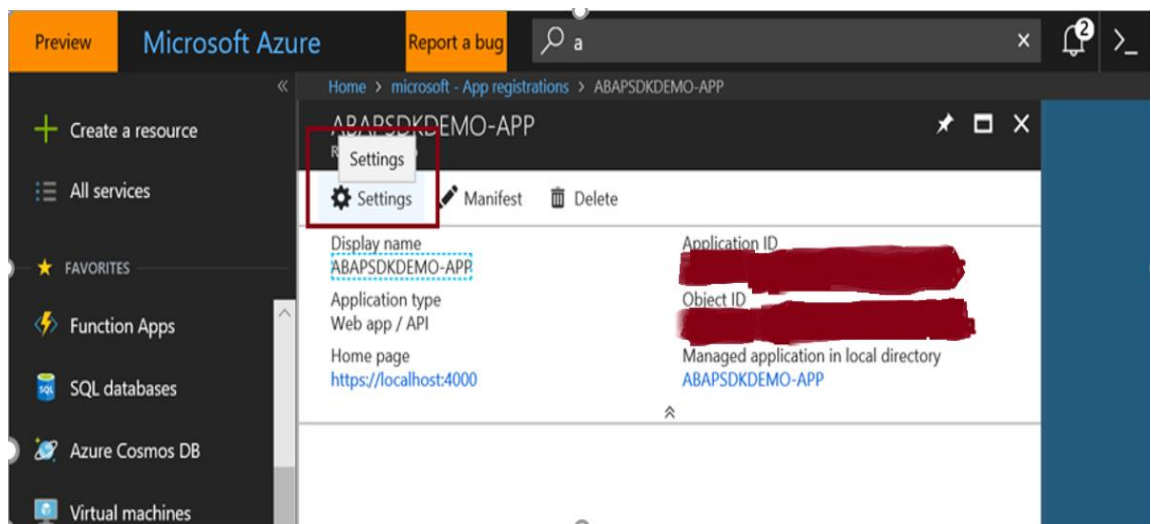


Application is created successfully.

Generate keys for your application

1. Once your application is created, go to your application by clicking on it.
   Copy the application id which will be required in the implementation of code in ABAP SDK. This application id is client Id.

   Click on 'Settings' in the above screen, go to the 'Required Permissions' under API Access as shown below.
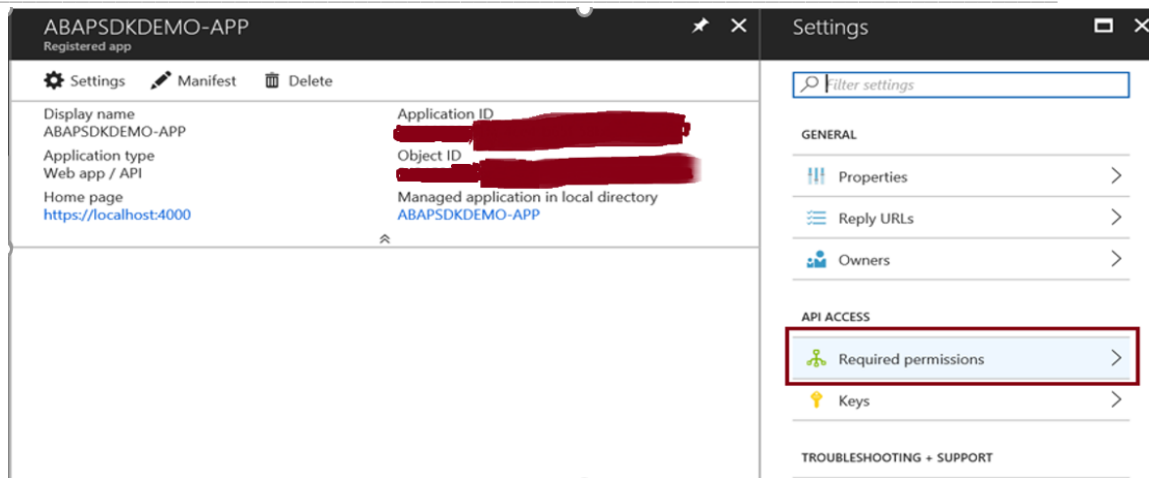
2. Then click on 'Add' button as shown in the below screen.
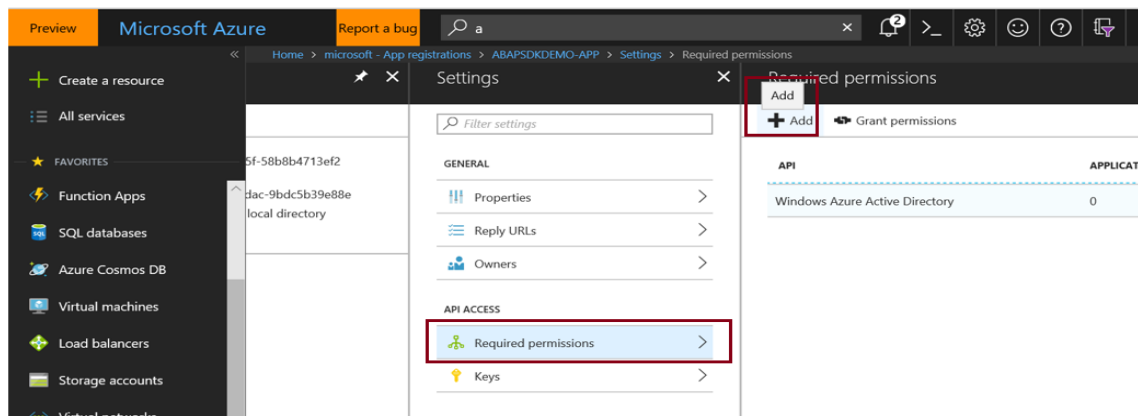
3. Under 'Add API access' section click on 'select an API 'and Search Azure Key Vault and Select the same as shown below.
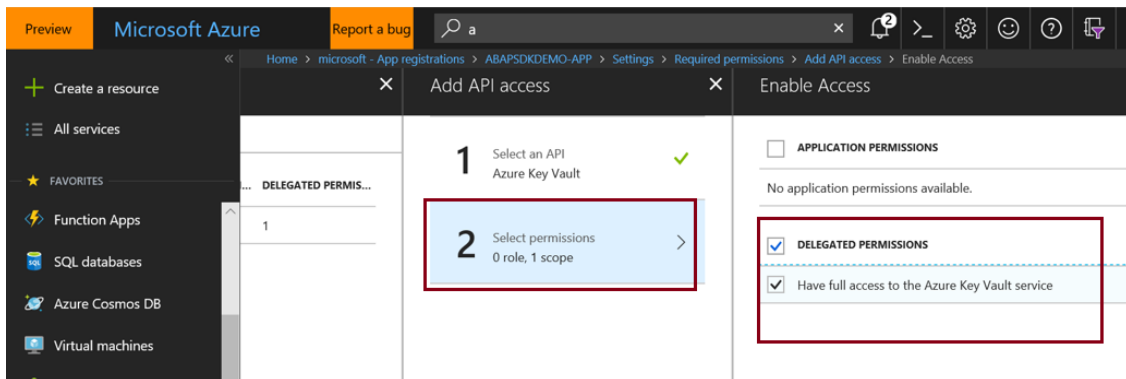
Note:
In this step, we have chosen Azure Key vault as an external application as an example.
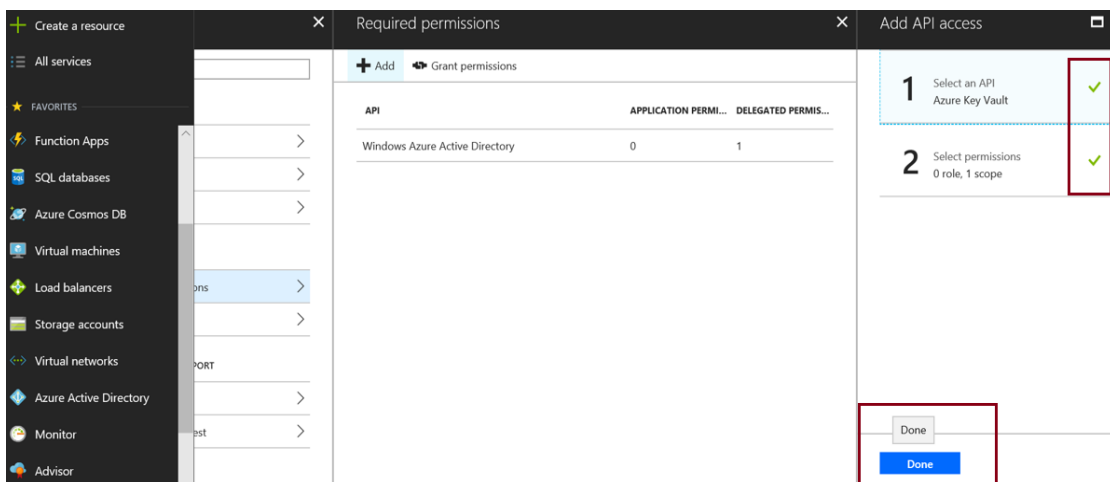In real world scenario, you need to choose your existing API which you want to access with AAD token for authentication.
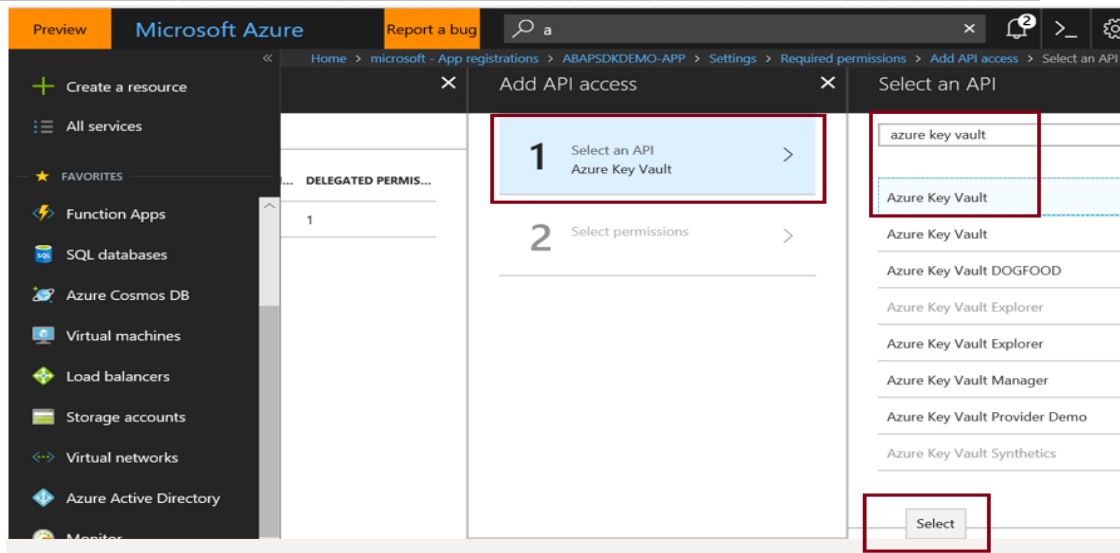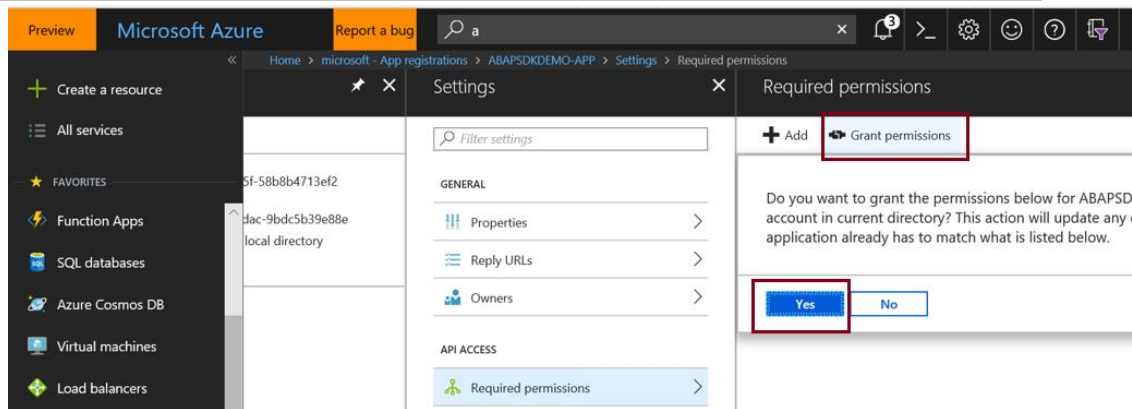
4. Under Add API access section, click on 'Select permissions' and enabled the checkbox for 'Delegated Permissions' as shown below.
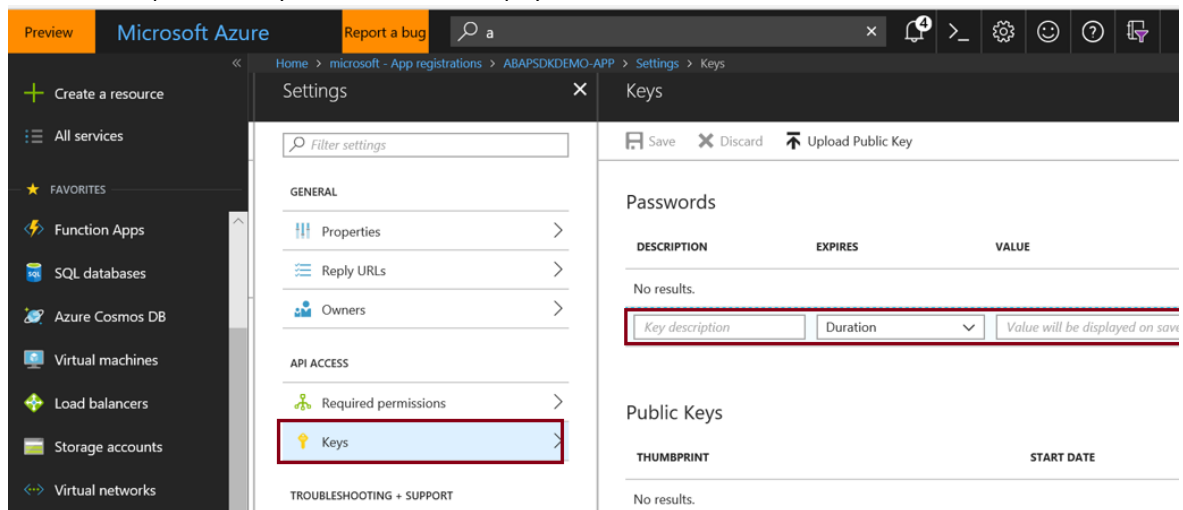

Then click on 'Done' button to complete the Required permissions activity.

5. Click on 'Grant permissions' and press 'Yes' button as shown below.

6. Click on 'Keys' section and provide key description and expiry Duration in the below screen. Under 'EXPIRES' dropdown list, you can select the expiry duration.

Here key description is provided as client_secret and it expires on 8/2/2019.
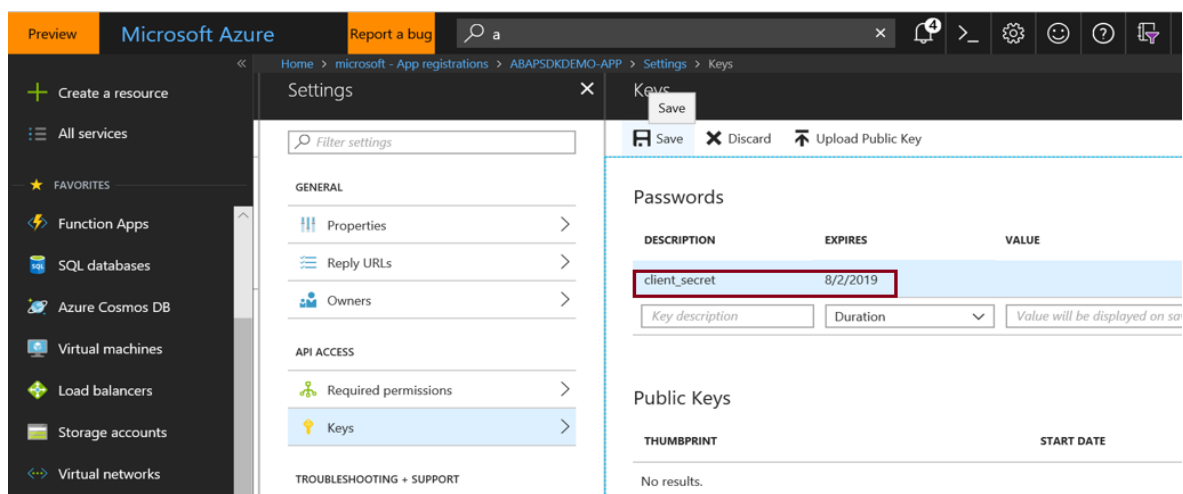
7. Click on 'Save' button to generate secret key. **Copy this key and it will be used in ABAP SDK implementation. Please remember you won't be able to retrieve this key once you leave the screen.**

Steps to use AAD authentication from SAP using ABAP SDK for Azure

Creation of RFC destination to Azure Active Directory
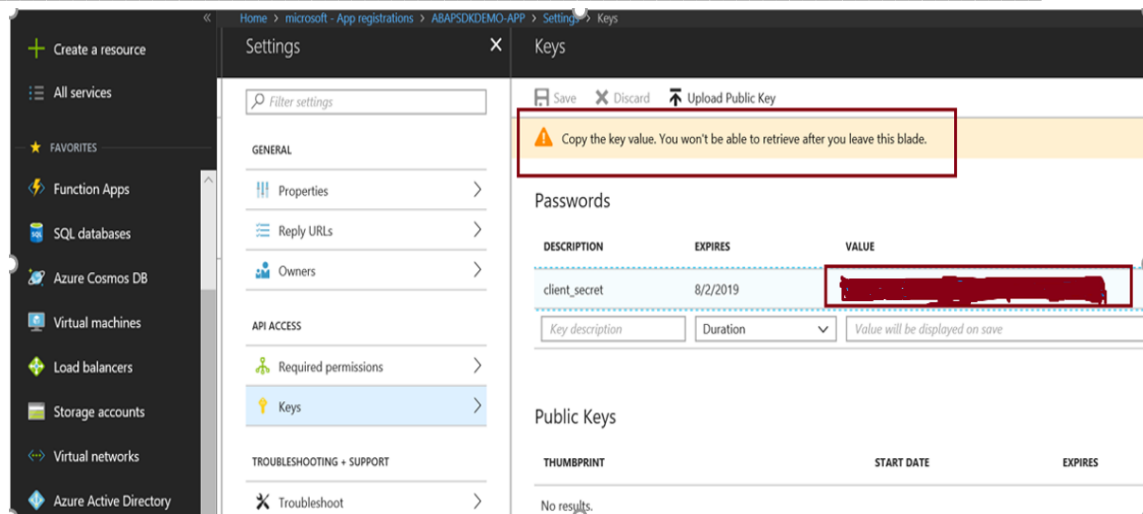
Go to transaction SM59 in your SAP system and create new RFC destination of type 'G'. Maintain your Azure

Active directory endpoint in the Target host and path prefix for authorization token as shown below.

Target host: **login.microsoftonline.com**

Port: **443**

Path Prefix: **/<Input Tenant ID>/oauth2/token**

For Tenant ID details and creating a new tenant id in Azure Active Directory, please refer this document section 'How to setup Azure Active Directory in Azure?'
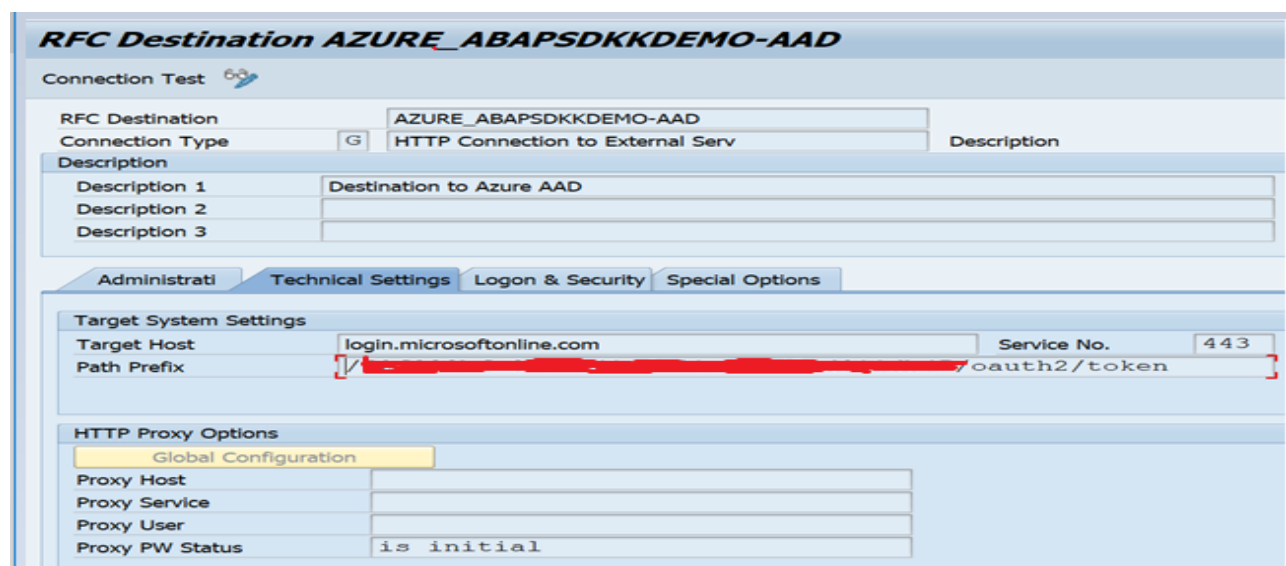
Now go to 'Logon & Security' tab and choose radio button SSL 'Active' and select SSL certificate 'DFAULT SSL Client (Standard)'.

Do a connection test to make sure it is working. RFC destination is working.

STRUST Setup

We need to import Microsoft's Certificate and import in STRUST for SSL handshake between SAP system and Azure Active Directory over HTTPS protocol. To download the certificate, in your browser, go to URL with the hostname and path prefix you used for creating RFC destination. https://login.microsoftonline.com/72f988bf-86f1-41af-91ab-2d7cd011db47/oauth2/token

Click on the Lock symbol you find next to refresh button in Chrome browser and select Certificate to view the certificate used for communication
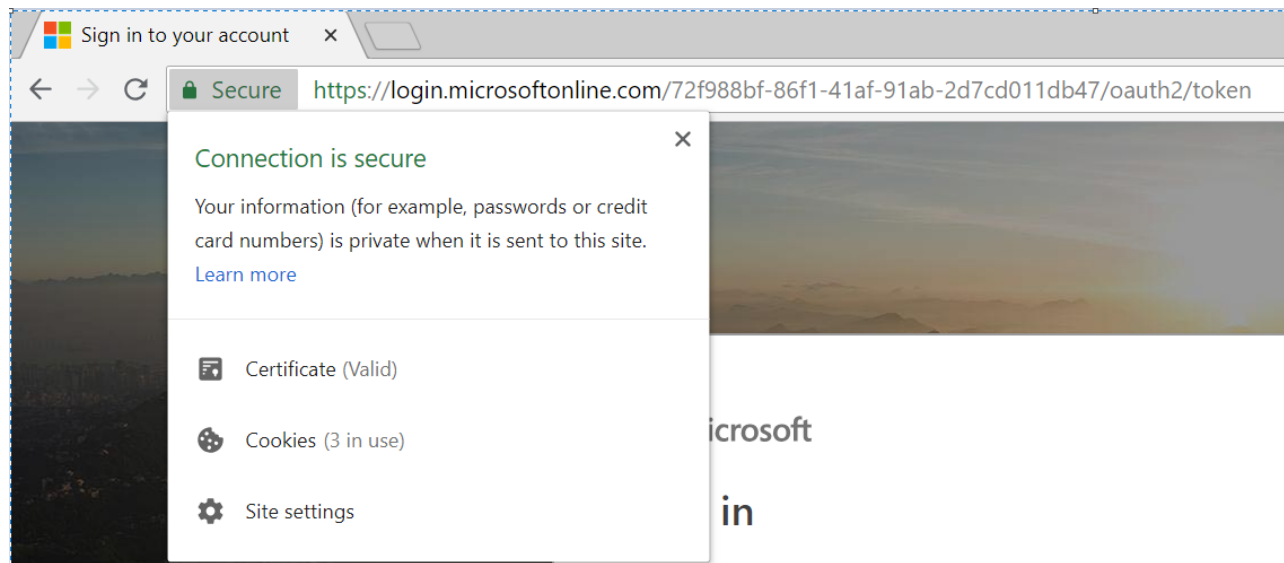
In the certificate, go to Details tab and Choose button 'Copy to File' to download the certificate to your local machine. Repeat the process and download all the certificate until root. In this case, you need to download two certificates.

1.   Microsoft IT TLS CA 4

2.   Baltimore Cyber Trust Root
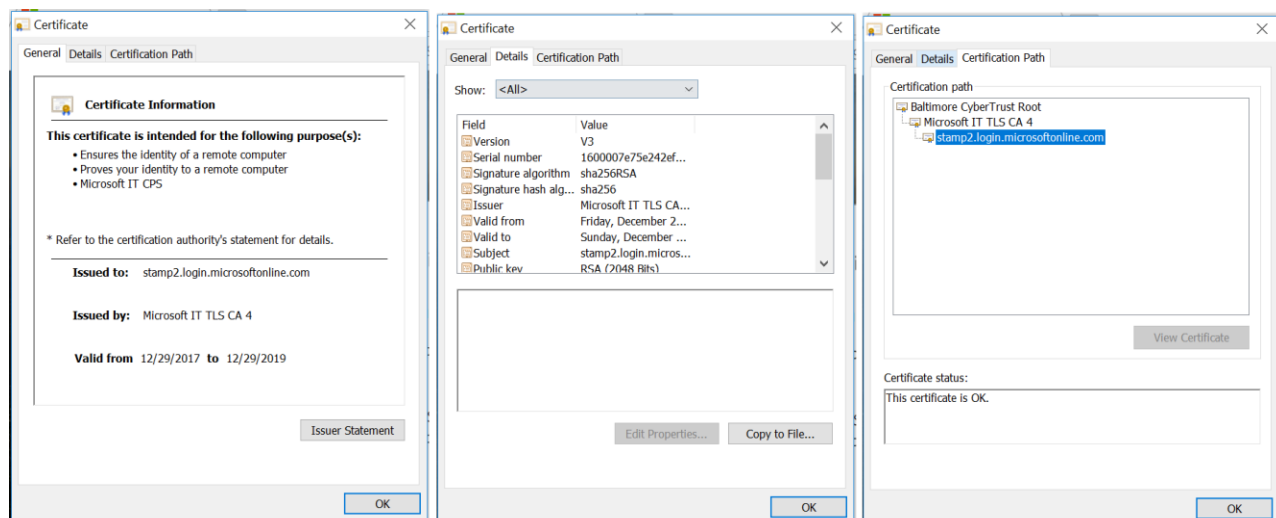
when all the certificates are downloaded, Go to STRUST transaction in your SAP system and Import all these certificates in DFAULT PSE.

Note: We are not going through the process of Importing certificates in STRUST in this document. It is straight forward, and your BASIS team can help you to do this activity.

Configuration

ABAP SDK has following main configuration tables and they need to be maintained. We will create a new Interface ID to establish connection between SAP system and target Azure Active Directory (AAD). A new Interface ID needs to be created for each AAD namespace.

ZREST_CONFIG – Master Table for Interface ID Maintenance. You must define a new Interface name and maintain the RFC destination that was created for target Event hub.

ZREST_CONF_MISC – This is an Interface Miscellaneous table which contains information on Alerts and re-processing of failed messages automatically.

ZADF_CONFIG – This is an Interface extension table. This stores data that is more specific to Azure Services like SAS keys, AAD secrets and processing Method.

ZREST_CONFIG

Create a new Interface ID like 'DEMO_AAD' and Maintain the RFC destination you created earlier.
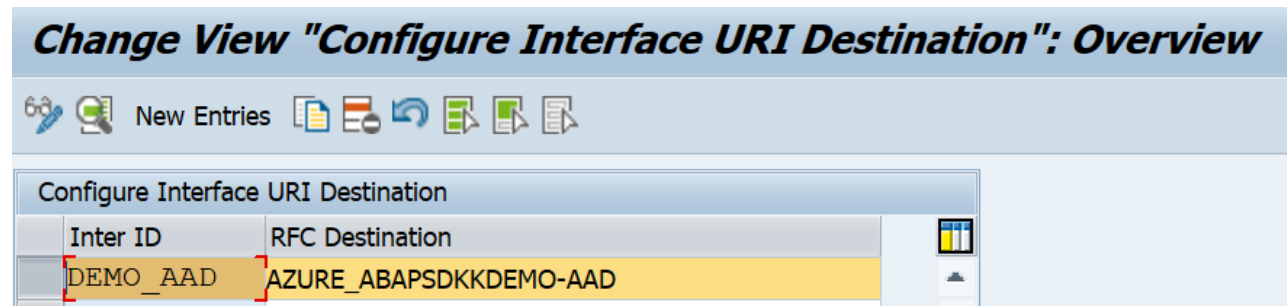
ZREST_CONF_MISC

Create an entry in table 'ZREST_CONF_MISC' for the above interface Id 'DEMO_AAD'.

Details of configuration:

- METHOD is 'POST'.
- MAX_RETRY is number of retry in case of service failure.
- EMAIL_ID is the email id for sending alerts.
- MAIL_BODY_TXT is Text Id to be maintained for the mail content.
- RETRY_METHOD is type of retrial (Regular '0' or exponential '1')

Table ZREST_CONF_MISC Display

MANDT	300
INTERFACE ID	DEMO_AAD
METHOD	POST
MAX RETRY	1
EMAIL ID	▪▪▪▪▪@microsoft.com
MAIL BODY TXT	ERROR WHILE SENDING MESSAGE TO AAD
RETRY METHOD	

ZADF_CONFIG

Create an entry in table 'ZADF_CONFIG' for the above interface Id 'DEMO_AAD'.

Details of configuration:

- INTERFACE_TYPE is 'Azure Active Directory'.
- SAS_KEY is the shared access key. This is the key which generated in AAD under section 'Generate keys for your application' Step 7 (refer Page 8). You need to change this key in this config table whenever key is changed in Azure.
- URI is left blank. This may be required for future versions.
- SERVICE_TYPE can be synchronous(S) or asynchronous(A)
- IS_TRY is a reprocessing flag, maintain as blank. it can be configured for reprocessing in case of failure of services.

Table ZADF_CONFIG Display

MANDT	300
INTERFACE ID	DEMO_AAD
INTERFACE TYPE	AAD
SAS KEY	*****
URI	
SERVICE TYPE	S
IS TRY	

DEMO Program

Please refer to DEMO program 'ZADF_DEMO_AZURE_AAD' to generate AAD token for AAD based authentication.

Please note that in the demo program, application id generated in step 1 of Generate keys for your application is used as client id.

```
Report                  ZADF_DEMO_AZURE_AAD              Active
    27                                  iv_business_identifier = gv_message_bid ).
    28    oref_aad ?=  oref.
    29
    30 □ TRY.
    31        CALL METHOD oref_aad->get_aad_token
    32          EXPORTING
    33            iv_client_id = '************' " Input client id as per implementation guide for AAD
    34            iv_resource  = 'https://vault.azure.net'  "Resource for Azure Key vault application
    35          IMPORTING
    36            ev_aad_token = gv_aad_token
    37            ev_response  = gv_response.
    38      CATCH zcx_interace_config_missing INTO cx_interface.
    39        gv_string = cx_interface->get_text( ).
    40        MESSAGE gv_string TYPE 'E'.
    41      CATCH zcx_http_client_failed INTO cx_http .
    42        gv_string = cx_http->get_text( ).
    43        MESSAGE gv_string TYPE 'E'.
```

ABAP SDK Monitor

We have provided an Interface Monitor (Transaction ZREST_UTIL), using this monitor you can view history of all the messages that were posted to Azure Services. In case you have a scheduled a background job to post messages to Azure, you can view the statuses of the messages in this monitor. This Monitor can be used for troubleshooting and re-processing of the message as well.

Go to transaction ZREST_UTIL and provide your Interface ID in the selection screen and execute to view all the messages

In this monitor, you can view the status of the HTTPs message and its headers, response, payload and so on. In case of errors, you can also re-process the message from this tool.

Auto re-processing of failed messages

For auto-processing of messages in case of failures, you must schedule a background job for program 'ZREST_SCHEDULER' as a pre-requisite.