SAP Analytics Cloud Embedded Edition

API Sample Scripts for Administration

User Guide

Document Version 0.7.2 – April 2022
Sample Scripts Versions 0.7
Document Classification: Public

THE BEST RUN

Contents

Introduction

Sample scripts have been developed for use with the SAP Analytics Cloud **Embedded Edition** API to ease its administration and remove any barriers to its adoption. These sample are freely available.

Although the sample scripts are provided on an 'as is' basis, these samples have been developed with 'best practices' in-mind.

The sample scripts have been developed in Postman using Java Script. Postman, which is freely available, is a commonly used tool for implementing API workflows. For those organisations that use Postman, the samples can be used immediately or with some minor modifications. For other organisations, Postman provides a good basis to demonstrate how requests are made to the API, and optionally export code snippets in various languages, such as Java, Python, C, PHP, Node.js and many others. This means development time can be reduced, in some case quite dramatically.

This document provides guidance on how to use these samples with Postman and is suitable for API developers as well as any SAP Analytics Cloud services administrator that wishes to complete administration tasks that can only be achieved with the API.

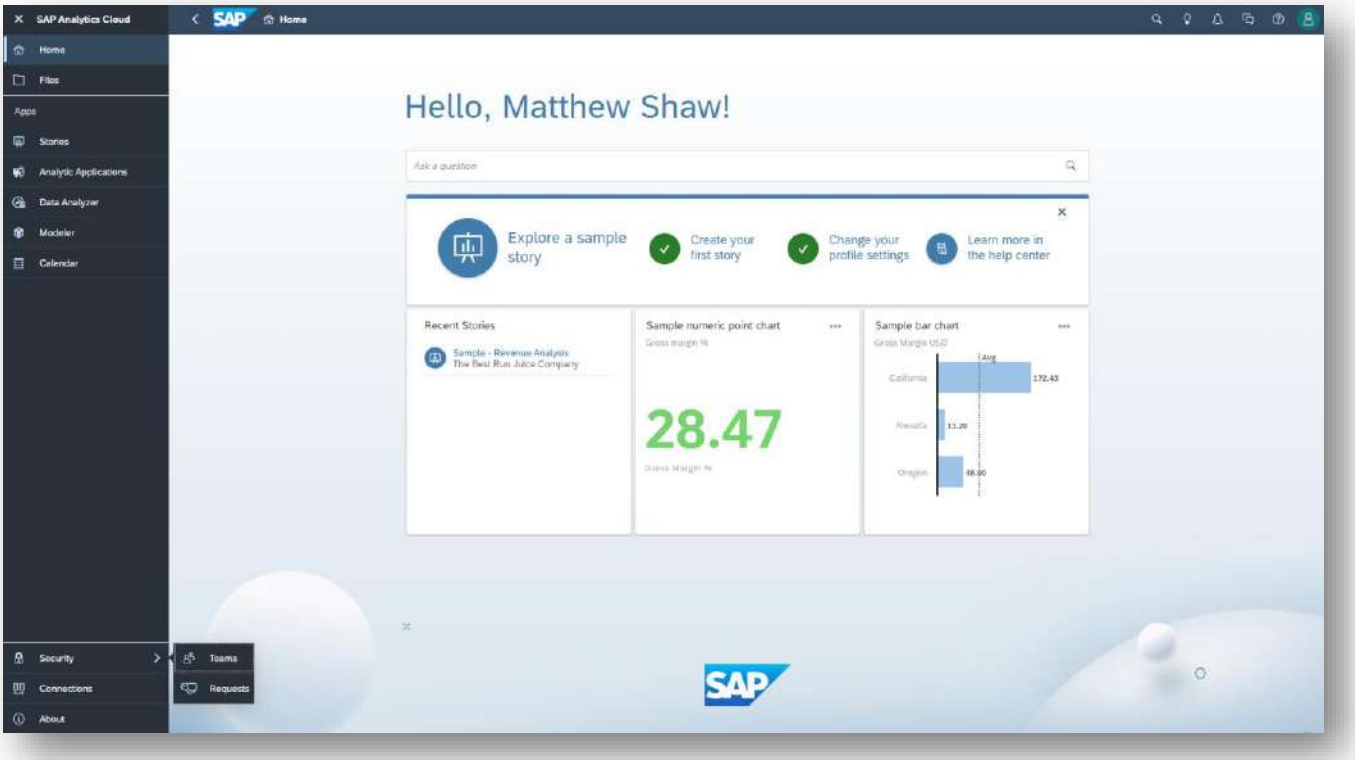
These samples provide administration tasks for ALL the endpoints of the Embedded Edition API. This means you will almost certainly find no need to develop or extend what is already provided here.

All the sample scripts manage the sessions' timeout and will automatically re-submit any requests following an error or when a new session needs to be established. It means all the session management and most aspects of error handling have been taken care of. It means you benefit from all the known and expected, but perhaps very rare, occurrences of errors from the API without the need to investigate, isolate the cause and resolve the problem yourself.

Once the OAuth Client has been added then the SCIM API becomes 'enabled' allowing users and teams to be created and managed via that API. You will need to use the SCIM API as this is the only means to add users into the Embedded Edition of SAP Analytics Cloud. For the SCIM API another comprehensive set of sample scripts are available. For more details, please visit https://blogs.sap.com/2021/05/28/sap-analytics-cloud-scim-api-best-practices-and-sample-scripts/ where the SCIM API can be used by both the Enterprise and the Embedded Editions of SAP Analytics Cloud. These SCIM API samples fully integrate with this Embedded Edition API Sample Scripts and are designed to be used in conjunction with each other.

It's important to understand that the APIs to manage the SAP Analytics Cloud Service (or tenant) is only applicable for the Embedded Edition. They are not applicable for the Enterprise Edition, unlike the SCIM API which is available for both editions.

Business Benefits and Overview



With the Embedded Edition of SAP Analytics Cloud, the user interface is simplified, and you cannot create new connections, nor perform any system administration tasks via the user interface. The screenshot above shows the interface for the most powerful user you can create. Notice the simplified menu. Almost all administration tasks must be performed via the API. To get up-to-speed as quickly as possible, these API Sample Scripts remove barriers to adopting the solution. They do this by removing the need to build or create any API scripts yourself. You don't need to try to understand how the API works, how sessions and errors should be managed or even how to write the various requests.

All the administration API endpoints are covered by these samples and there should be no need for you to edit or enhance them any further.

The business benefits are summarised in the blog that introduces these sample scripts in addition to the best practices for the 'Administration' API and the Embedded Edition overall. Please refer to https://blogs.sap.com/2022/04/07/sap-analytics-cloud-embedded-edition-best-practices-sample-scripts-for-administration/ The blog post also provides a means to ask the community and the author questions about these samples or related best practices.

Initial Setup with Postman

The initial setup requires the following steps to be completed:

A. Download and install Postman Desktop
B. Download the samples
C. Import and define the environment in Postman
D. Import sample collections into Postman
E. Run sample collection 'Embedded 701-Test Tenant Environment Setup' to test the environment has been setup correctly

The following sections provides step-by-step instructions on each of these steps.

Step A – download and install Postman

Download Postman from the Postman web site. You will need the client https://www.postman.com/downloads/. You may also require the Postman Desktop Agent, but that isn't essential and is only need for the web version of Postman with these scripts.

Postman can be run on a desktop client, or via the web using a desktop agent, but Postman also has a command line interface which is more suitable for enterprise deployments. This document covers only the client setup. For more details on how to run Postman command line please refer to the Postman web site.

Step B – download the samples

Download the samples from https://github.com/SAP-samples/analytics-cloud-scim-api-samples (direct zip download). These samples include 2 'sets' of samples:

1. A set for the SCIM API, so you can manage users and teams. These are stored in the '**SCIM**' folder. These are documented elsewhere.
2. A set for administration tasks for the Embedded Edition found in the '**Embedded**' folder. This user guide only covers the contents in this Embedded folder.

Within the **Embedded** folder, the samples consist of the following components:

1. The sample scripts themselves. These are shared as json files, each contains a Postman collection, and each collection is a sample script. They were exported as a collection v2.1 file. The first release of these samples contains 18 collections and thus there is a collection json file for each sample.
2. Environment. This is a .json file containing a Postman environment. The environment defines the username, password and server details the samples will need to use. A single environment will be used by multiple Postman collections. There is just 1 environment json files provided which act as a 'template' for you to complete for your SAP Analytics Cloud Embedded Edition Service(s).
3. Example data files. These are json and csv files that the Postman collections will use as data files. These data files drive the script input. For example, a script that creates Live Connections will read a data file containing the Live Connection details to be added. The example data files provide the structure or template of what each script is expecting and provides a good basis to amend for your own purposes. Most scripts have example data files, though not all. For all the scripts that can read data file an example data file is provided.

Step C – import and define the environments into Postman

Before you can use any of the sample scripts, you'll need to import the Postman environments and configure them appropriately.

A summary of the environment variable configuration is:

Variable		Description
Embedded_uua_url_FQDN	Mandatory	Fully qualified hostname of the uua url without the leading http://
Embedded_uua_clientid	Mandatory	The uua clientid
Embedded_uua_clientsecret	Mandatory	The uua clientsecret (typically ends with a =)
Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN	Mandatory	Fully qualified hostname of the endpoints sac_embedded_ edition_service_config. Importantly, it's the fully qualified hostname, not the full URL, i.e., without the leading http:// and without the trailing /api/v1/
Embedded_tenant_uuid	Mandatory	The tenant_uuid
SACplatform	Mandatory	The platform SAP Analytics Cloud is hosted on. The only supported value is "CF" for the Embedded Edition.
Embedded_OAuthClientName	Optional	The name of the OAuth Client, that should either be 'added' or 'used' depending upon the sample script that is used
ContentNamespace	Optional	The Content Namespace that should either be 'set' or 'used' depending upon the sample script that is used

In addition to the above variables, there are others that are needed but these are only used by the 'SCIM' API sample scripts. These variables are listed below and form part of the environment 'template' you import. These variables are not used for any of the 'Embedded Edition' samples. However, some of the sample scripts within this 'Embedded Edition' automatically update some or all the variables below. This then means the very same Postman Environment can be used for the 'SCIM' samples without having to manually update either these variables, or configure a new Postman Environment for use with the SCIM API Sample Scripts:

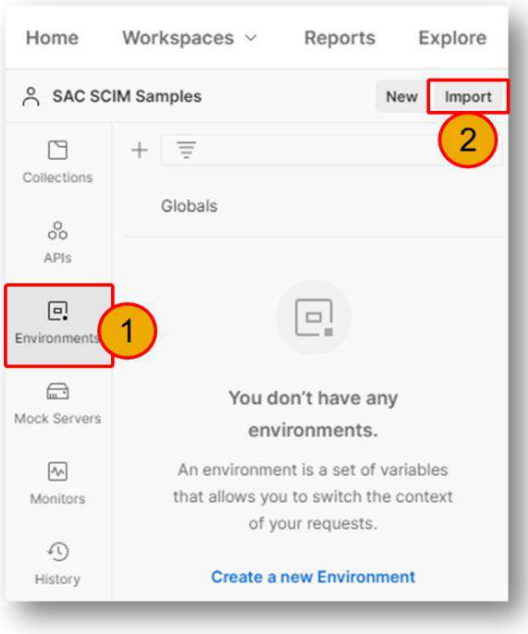
Variable		Description
SACserviceFQDN	Not used	The hostname, fully qualified, of the SAP Analytics Cloud Service
SACtokenFQDN	Not used	The hostname, fully qualified, of the OAuth Client Token Service. Importantly, it's the fully qualified hostname, not the full URL.
Username	Not used	Username of the OAuth client
Password	Not used	Password of the OAuth client without any carriage return at the end!

SAMLSSO	Not used	Defines the Authentication method defined for the SAP Analytics Cloud Service. If you are using the default Identify Provider that comes with SAP Analytics Cloud, then use a value of “default”. Otherwise, set this to “userid”, “email” or “custom” depending upon how you configured SAML Single-Sign-On.

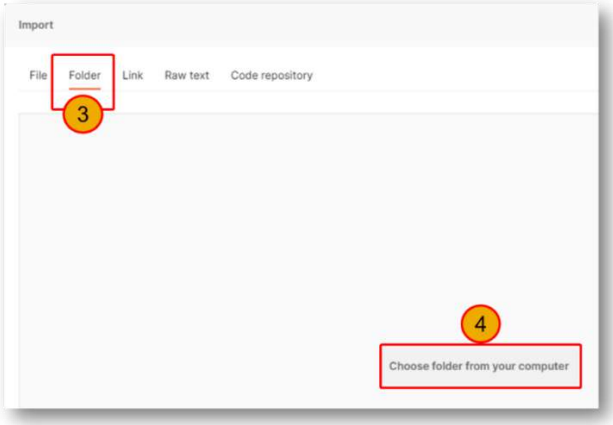
Individual step-by-step instructions are on how to set these variables correctly follows here:

Import environment ‘templates’

Open Postman client. If you’re new to Postman, you’ll appreciate to know a ‘workspace’ has been created for you. You can have multiple workspaces. You’ll work with just one workspace for this setup. Environments are stored inside a workspace.

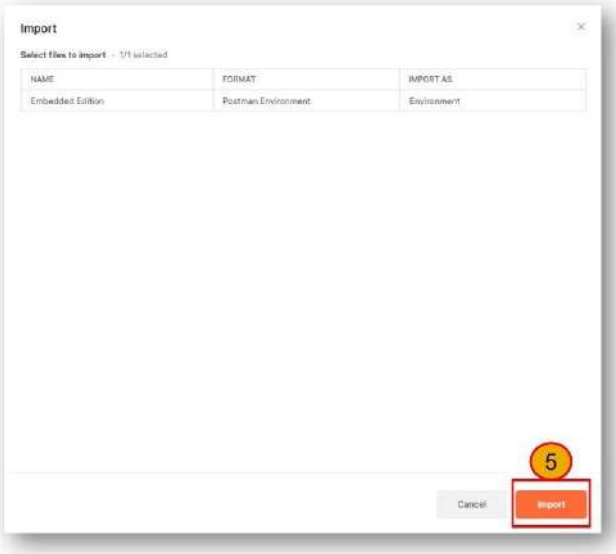


1. Select the ‘Environments’ tab
2. Select ‘Import’

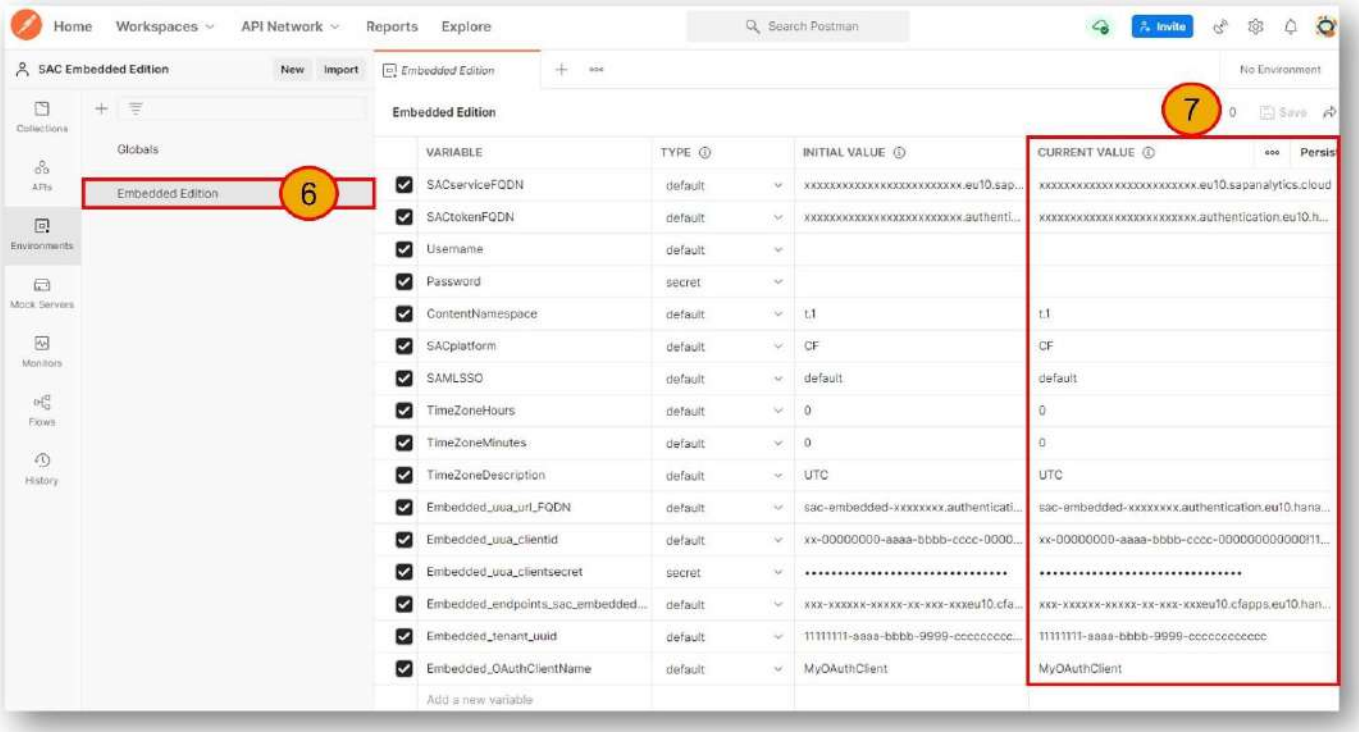


3. Select ‘Folder’

4. Select ‘Choose folder from your computer’. Then select the folder where you downloaded the environment json file earlier. You need to select the ‘Embedded\Embedded Environments’ folder and press ‘Select Folder’



5. Select Import. This will import the 2 environment json files into your Postman workspace. Each act as a template for you to edit for your SAP Analytics Cloud Service.



6. Ensure the environment called ‘Embedded Edition’ is selected
    a. Optionally feel free to rename it appropriately.
    b. If you have more than one SAP Analytics Cloud Embedded Edition Service, duplicate this environment, one for each SAP Analytics Cloud Embedded Edition Service and name them appropriately. For each of these environments, you’ll need to repeat the following steps, one for each SAP Analytics Cloud Embedded Edition Service.

7. We need to define this environment for your SAP Analytics Cloud Embedded Edition Service(s).
    a. The environment consists of numerous variables, such as 'Embedded_uua_clientid', 'Embedded_uua_clientsecret' and many others.
    b. We need to update the '**Current Value**' for many of these variables. For the moment, ignore the 'Initial Value'
    c. The next steps will guide you through where to obtain each of these values
    d. There's nothing to do in this step, but just remember that the following steps will require you to update the values shown in box 7.

Pre-requisites

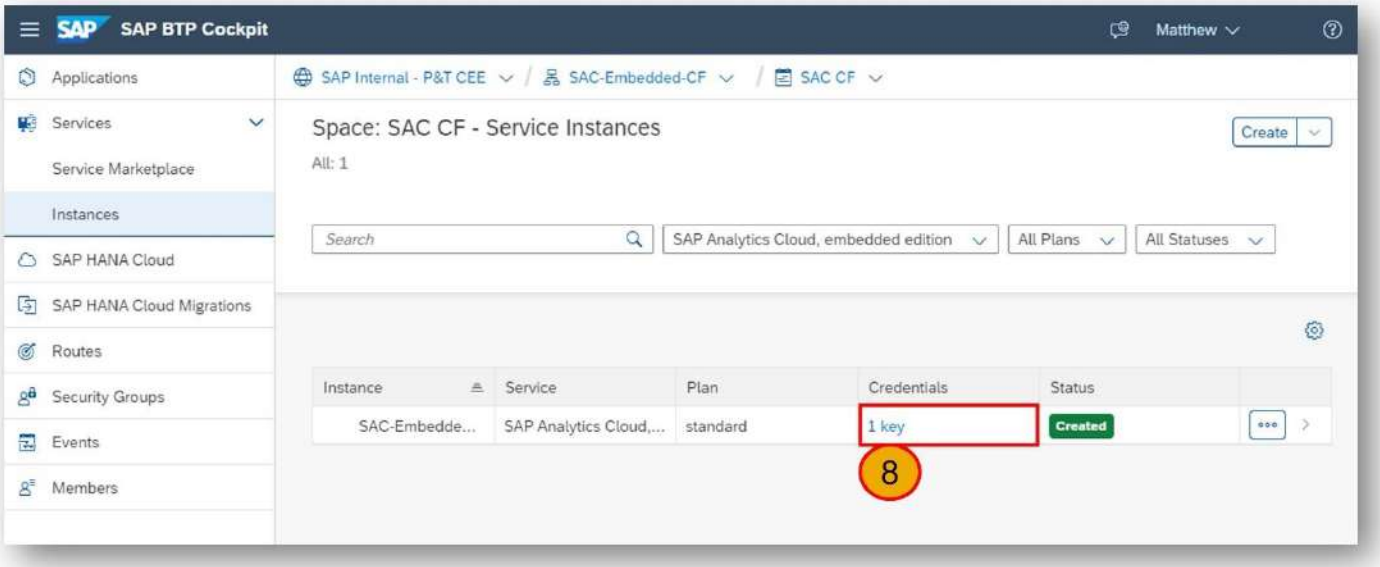
This user guide assumes you have already performed the following:

1. You have created a subaccount within your Global Account cockpit on the SAP Business Technology Platform.
2. You have granted that subaccount an Entitlement Assignment with a Service Plan for SAP Analytics Cloud Embedded Edition.
3. You have created a Space with a Quota Plan assigned to it.
4. You have created a new instance of 'SAP Analytics Cloud Embedded Edition' and you have then created a Service Key on that instance.

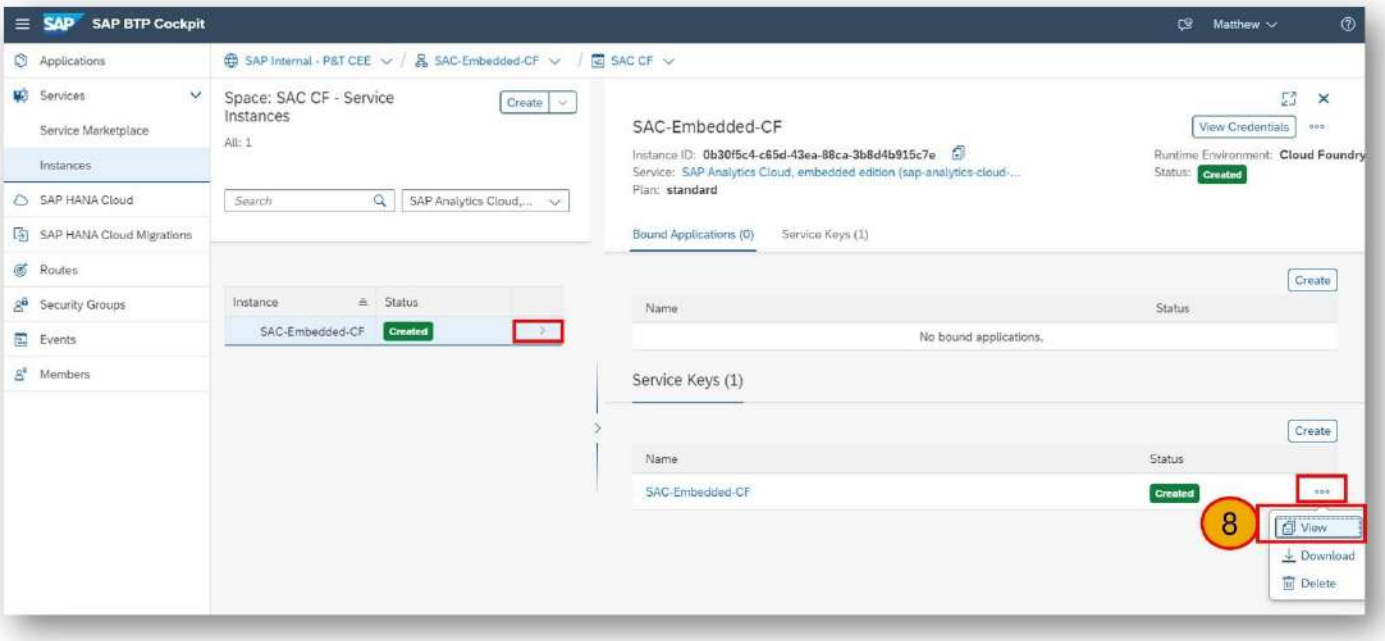
	For step-by-step instructions to complete the pre-requisites, please refer to this blog https://blogs.sap.com/2021/08/24/getting-started-with-sap-analytics-cloud-embedded-edition-btp-service.-part-i/ You should stop at the end of part 1 of this blog, ignoring parts 2, 3, 4 and 5, since these sample scripts provide a more comprehensive solution that was developed after the blog was written.
	Official documentation SAP Analytics Cloud, Embedded Edition: Getting Started Guide includes details of how to create an instance of SAP Analytics Cloud Embedded Edition and how to create a Service Key

View Service Key

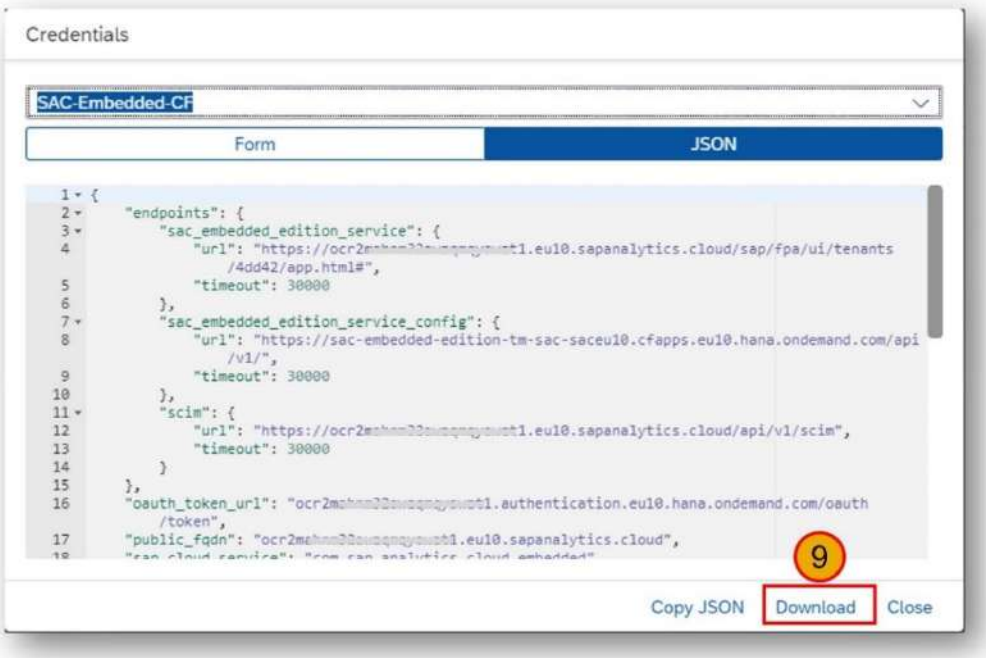
Once you have completed the above tasks, you will be able to access your service key. The service key will provide us with values we need to store inside the Postman 'template' Environment you just imported. The service key is shown here:



You can also access the service key by clicking on the '>' and '…' as shown below:



8. View the key, by either of the methods mentioned above. The result should show a dialogue box like this:

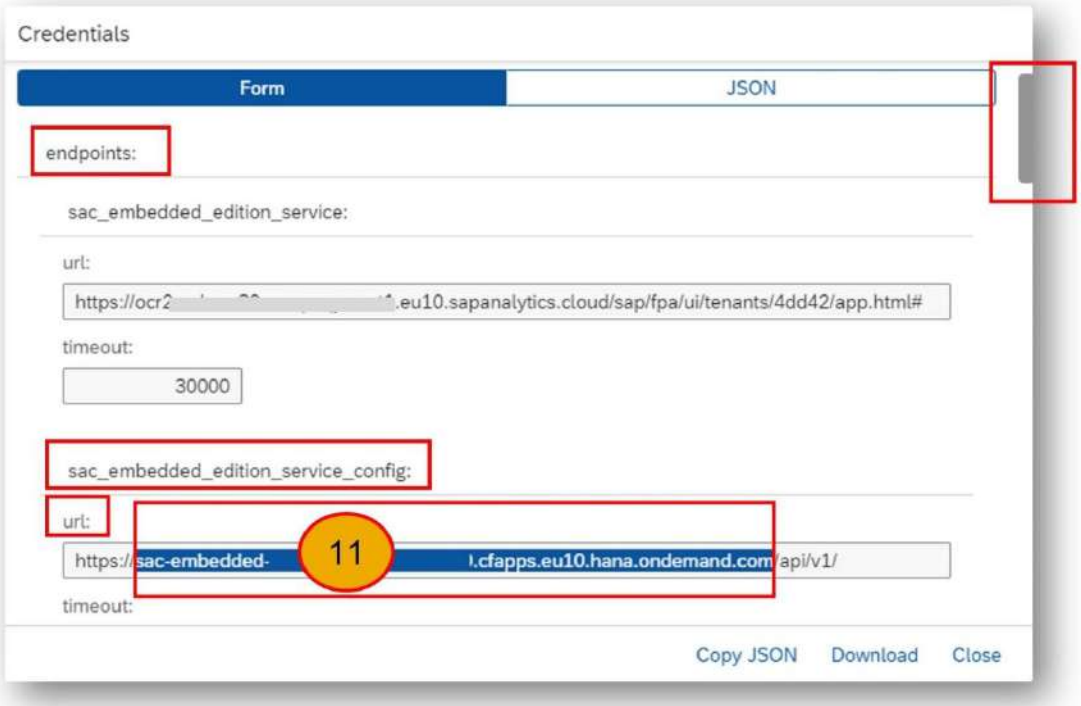


9. The following steps require copying values from within this service key into your Postman environment as discussed in Step 7. You may optionally download the service key.

Set Postman variables

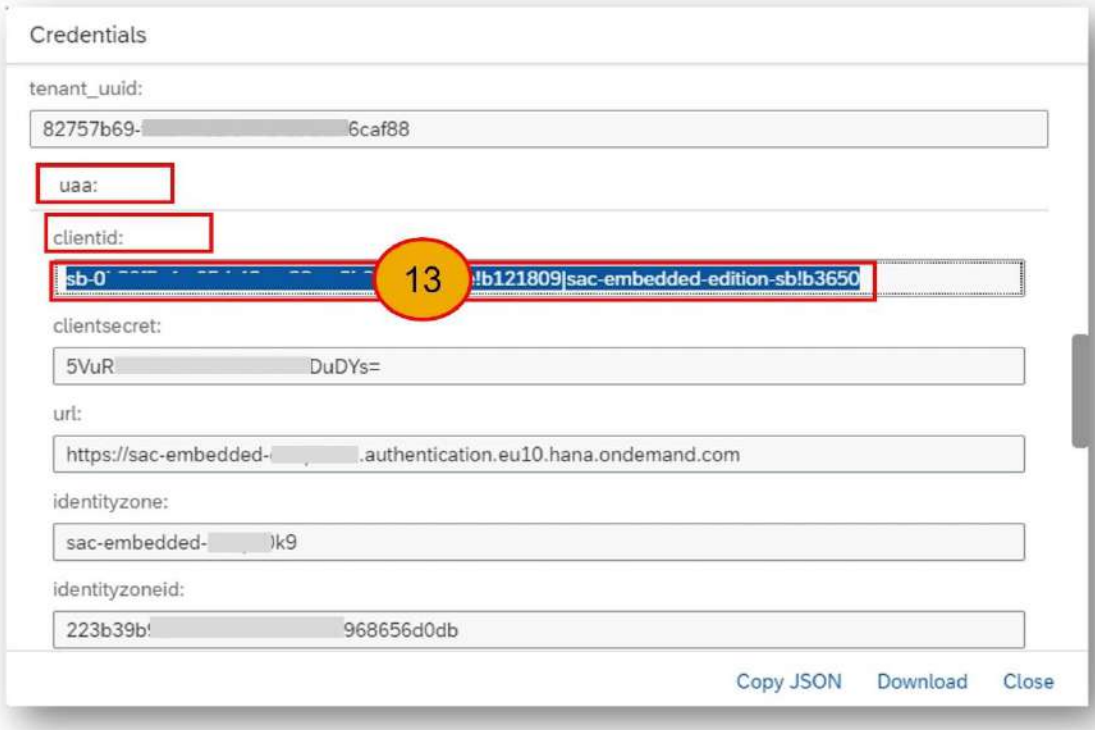
We will now copy the appropriate values from the service key into the Postman Environment:

10. Select the 'Form' option (since it makes it a little easier to follow instructions that reading the raw JSON!)

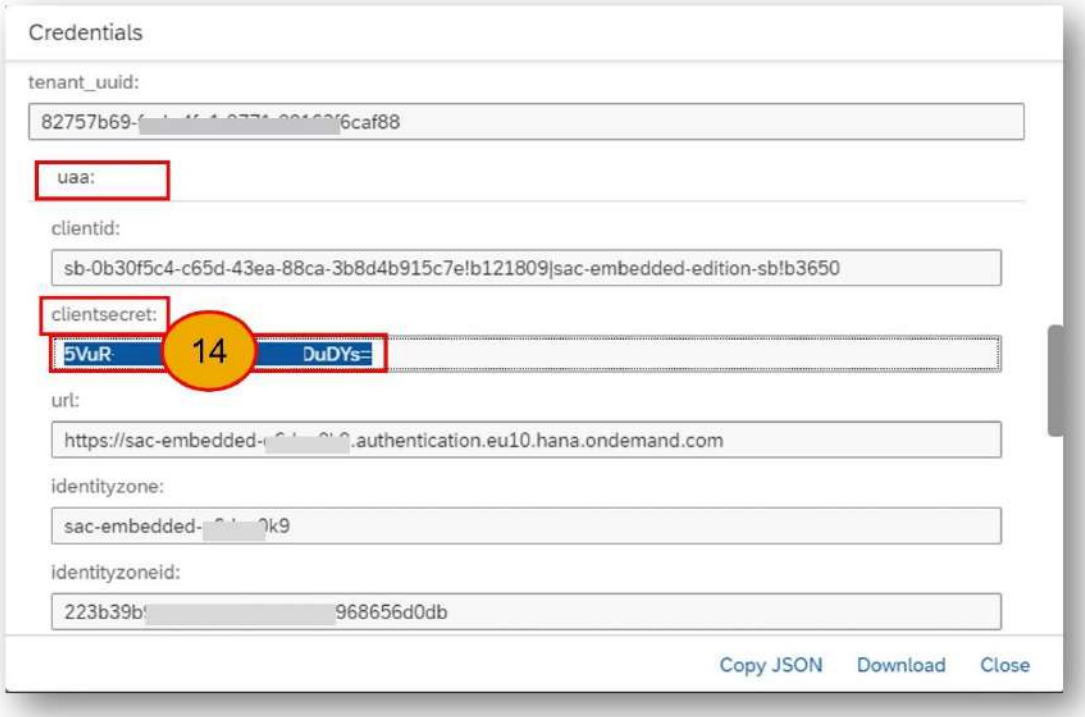


11. For the '**endpoints**' '**sac_embedded_edition_service_config**' '**url**' we need to copy just the fully qualified domain name (FQDN) that is shown, i.e., ignore the leading https:// and ignore the trailing /api/v1/. Paste this value into the Postman Current Value for the variable '**Embedded_endpoints_sac_embedded_edition_service_config_FQDN**' (see step 7 for a screenshot). It's very easy to copy the wrong value as only just above is another parameter with <u>almost an identical name</u>. So be extra careful and ensure you copy the value from the parameter that ends "**_config**"

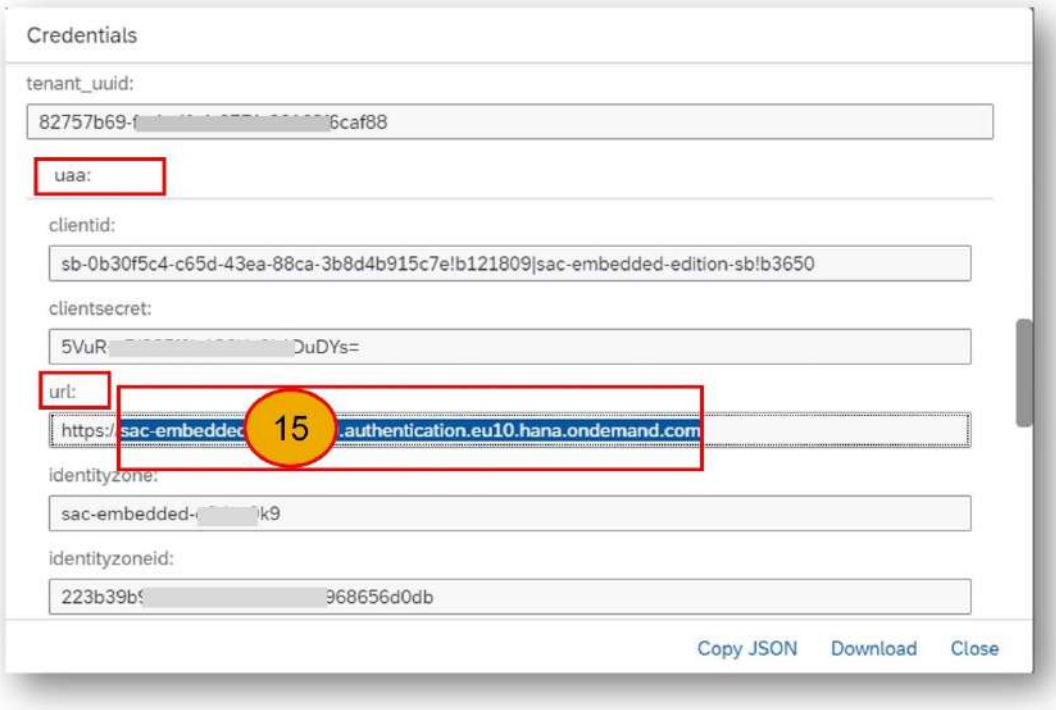
12. Scroll down a little, and copy the value for '**tenant_uuid**' into your Postman Current Value for the variable '**Embedded_tenant_uuid**' (see step 7 for a screenshot)



13. Copy the value for '**uaa**' '**clientid'** into your Postman Current Value for the variable '**Embedded_uua_clientid**' (see step 7 for a screenshot)

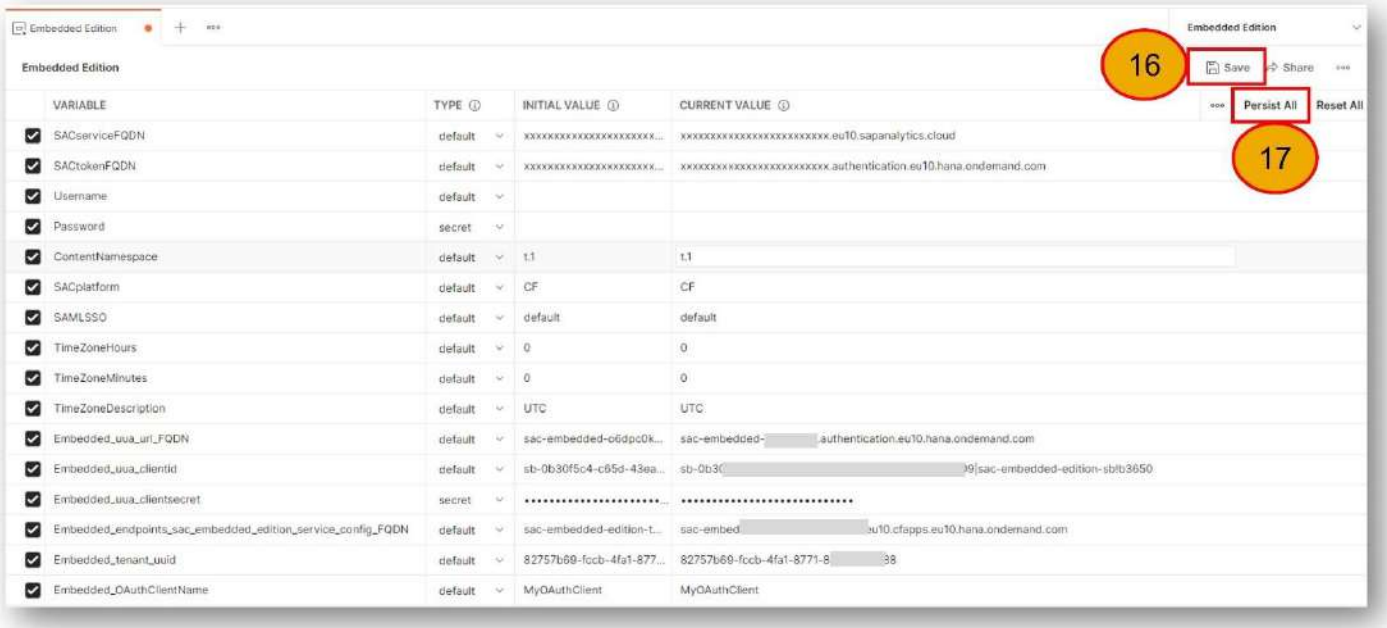


14. Copy the value for '**uaa**' '**clientsecret'** into your Postman Current Value for the variable '**Embedded_uua_clientsecret**' (see step 7 for a screenshot)



15. For the '**uaa**' '**url'** we need to copy just the fully qualified domain name (FQDN) that is shown, i.e., ignore the leading https://. Paste this value into the Postman Current Value for the variable '**Embedded_uua_url_FQDN**' (see step 7 for a screenshot)

Your Postman environment should now look something like this:

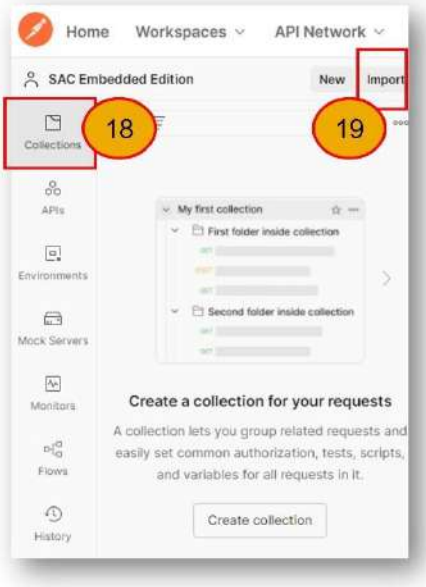


16. Press Save.
17. Optionally press ‘Persist All’. This will copy the values form the ‘Current Value’ into the ‘Initial Value’ column. We will only work with ‘Current Values’. ‘Initial Values’ are used when the environment is shared with others.

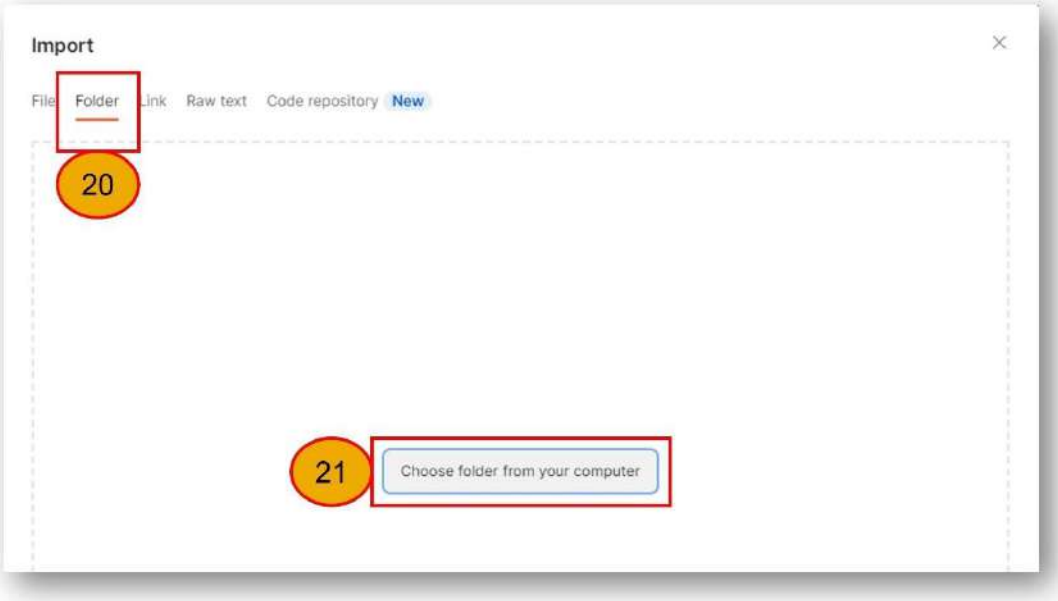
At this point you will have completed the initial setup of your Postman Environment. We now need to import the actual sample scripts into Postman.

Step D – import sample collections in Postman

Now the environment has been setup we can import the sample scripts, each is called a collection in Postman. We can then run a test collection to check the environment we've setup is valid.

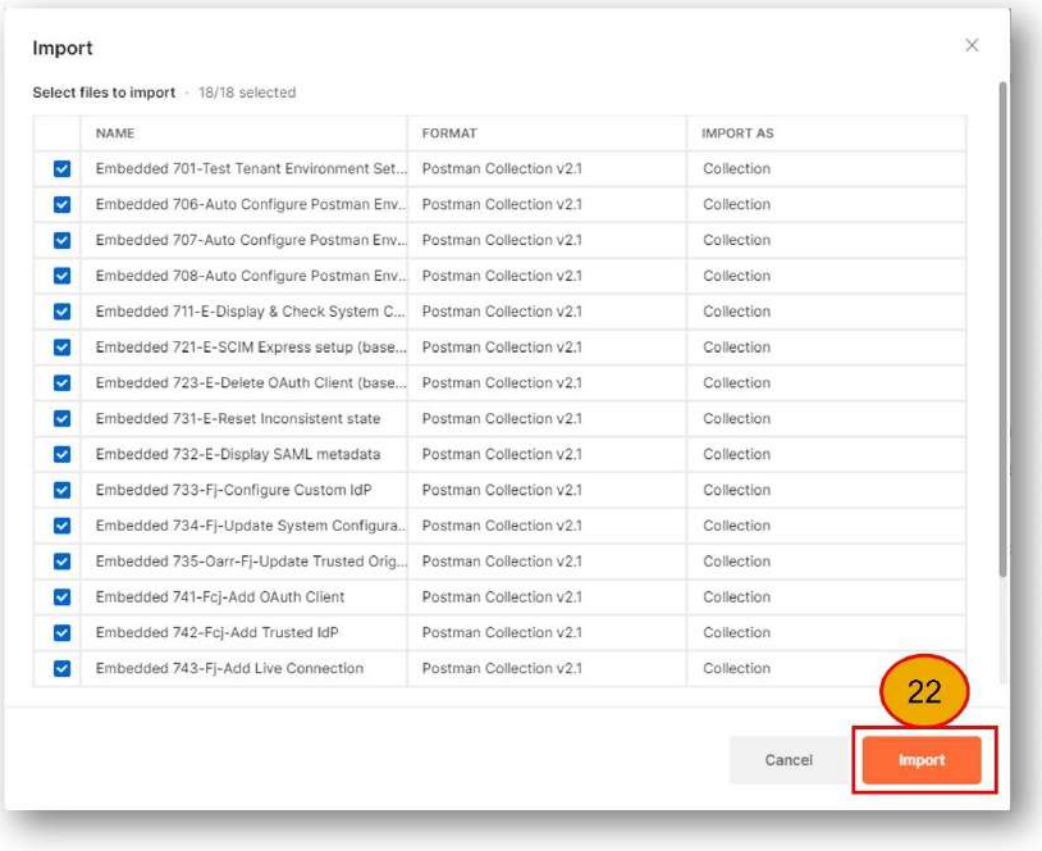


18. In Postman select the 'Collections' tab
19. Click 'Import'



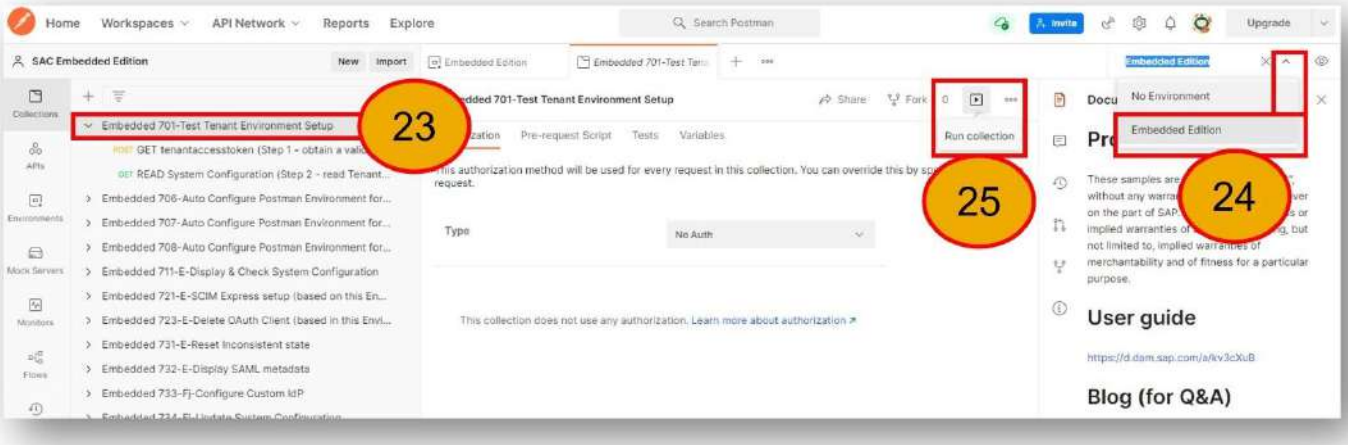
20. Select 'Folder'

21. Select ‘Choose folder from your computer’ and then select the folder where you downloaded the sample scripts to (Embedded\Embedded Samples) and press ‘Select Folder’

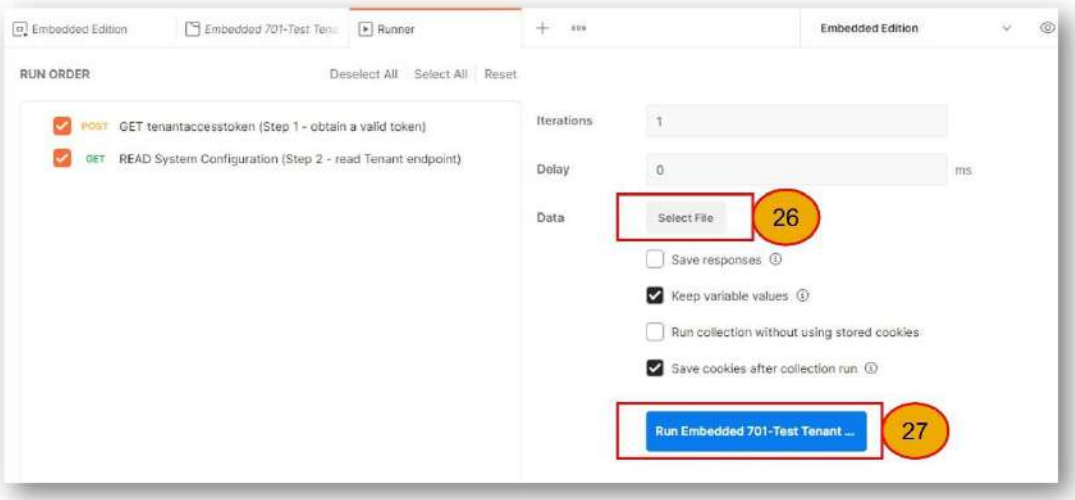


22. Select ‘Import’. This will import all the sample scripts (collections) into your Postman Workspace.

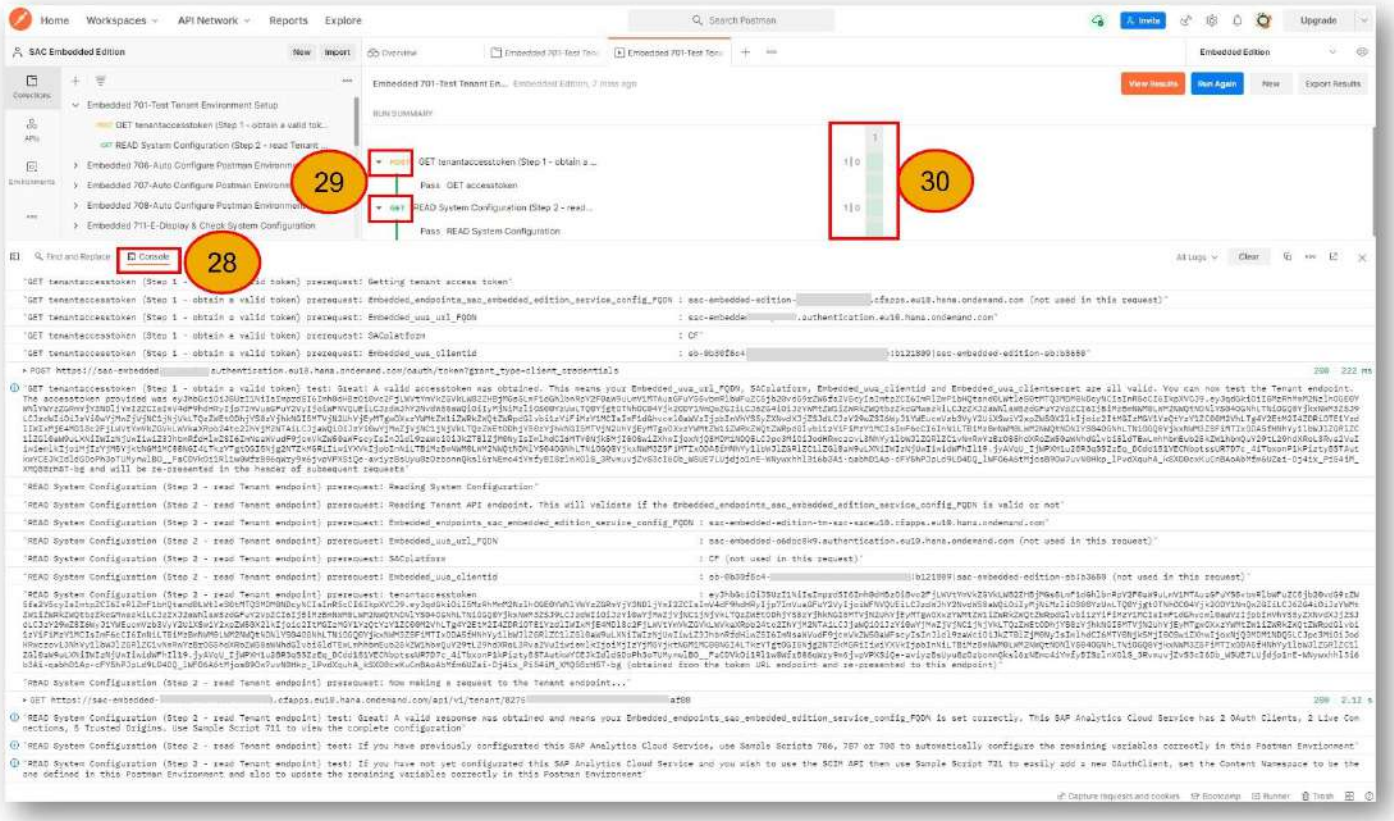
Step E – Run sample collection 701 to test the environment has been setup correctly



23. Select the collection 'Embedded 701-Test Tenant Environment Setup'. A new window will open for this collection. This collection will perform a very simple test against the API. The collection is quite harmless, and no updates will be performed.
24. Select the environment which was defined in step C. (We need to select the environment <u>before</u> we press run since changing the environment after we press run has no effect)
25. Press Run (having previously selected the correct environment)



26. A new window, called a 'Runner' will open. For this collection there is no data file, however this is where you'd select a data file if the collection required it. These data files were introduced in step B 'example data files'. Many, though not all collections require a data file. It's important not to miss this step if a data file is required. However, for this collection, no data file is needed, so we don't need to select one.
27. Press Run.



28. Open the Console and enlarge the console window.
    a. The console shows important information.
    b. For this collection, the script is particularly verbose and provides a great deal of information to help you determine where any problem may be.
29. Open the 'Run Summary' for each of the requests.
30. You should observe 'pass' tests. If you do, then everything is well, and you've successfully configured the environment and run a simple test. The console will tell you how OAuth Clients, Live Connections and Trusted Origins are in your SAP Analytics Cloud Service. If either of these tests fail, inspect the console carefully for clues to where you went wrong.

If you have successfully configured the Postman Environment the console will show:

```
READ System Configuration (Step 2 - read Tenant endpoint) test: Great! A valid response was
obtained and means your Embedded_endpoints_sac_embedded_edition_service_config_FQDN is set
correctly. This SAP Analytics Cloud Service has 0 OAuth Clients, 0 Live Connections, 5 Trusted
Origins. Use Sample Script 711 to view the complete configuration

READ System Configuration (Step 2 - read Tenant endpoint) test: If you have previously configurated
this SAP Analytics Cloud Service, use Sample Scripts 706, 707 or 708 to automatically configure the
remaining variables correctly in this Postman Environment

READ System Configuration (Step 2 - read Tenant endpoint) test: If you have not yet configurated
this SAP Analytics Cloud Service and you wish to use the SCIM API then use Sample Script 721 to
easily add a new OAuthClient, set the Content Namespace to be the one defined in this Postman
Environment and also to update the remaining variables correctly in this Postman Environment
```

Introduction to sample scripts (collections)

The samples that are included are summaries here:

Test

- Embedded 701-Test Tenant Environment Setup

You have already used 701 to test you have made the initial Postman Environment setup correctly in the step-by-step instructions.

Auto Configure Postman Environment

- Embedded 706-Auto Configure Postman Environment for SCIM (step-by-step instructions)
- Embedded 707-Auto Configure Postman Environment for Modelling
- Embedded 708-Auto Configure Postman Environment for Story Listing

These samples are needed when connecting to an instance of SAP Analytics Cloud Embedded Edition that has already been setup with an OAuth Client. If you work for SAP and connecting to a customer environment, then this will also be applicable. The auto configuration scripts will identify the right OAuth Client, for that use-case, and further configure the Postman Environment so the SCIM API sample scripts work without the need for further manual configuration. There are currently no sample scripts for Modelling or Story Listing.

Step-by-step instructions for 706 are available in a following section

Display & Check System Configuration

- Embedded 711-E-Display & Check System Configuration

This sample does nothing, except print to the console the entire configuration of this instance of SAP Analytics Cloud Embedded Edition. This is particularly helpful for troubleshooting and documenting the current configuration. There's no need to run 706, 707 or 708 to display the system configuration. This sample doesn't require a data file and is completely harmless, no updates are made to the system or to the Postman Environment. It's worth running this sample to see the current configuration. Remember the output is shown in the Postman Console.

The console log will also identify what aspects of the current Postman Environment are set correctly, nor not, for the current configuration of your SAP Analytics Cloud Embedded Edition instance. This aspect is particularly helpful to avoid other issues.

Express Setup

- Embedded 721-E-SCIM Express setup (based on this Environment) (step-by-step instructions)
- Embedded 723-E-Delete OAuth Client (based on this Environment)

These sample configure this instance of SAP Analytics Cloud Embedded Edition as defined in the Postman Environment. They operate without the need for any data files. Instead, it uses the Postman Environment variables to configure it. This is ideal for first time setup as it combines multiple tasks into a single script that doesn't require any additional configuration or data files to drive it.

Step-by-step instructions for 721 are available in a following section

General Administration

- Embedded 731-E-Reset Inconsistent state
- Embedded 732-E-Display SAML metadata (step-by-step instructions)
- Embedded 733-Fj-Configure Custom IdP (step-by-step instructions)
- Embedded 734-Fj-Update System Configuration
- Embedded 735-Oarr-Fj-Update Trusted Origins
- Embedded 741-Fcj-Add OAuth Client
- Embedded 742-Fcj-Add Trusted IdP
- Embedded 743-Fj-Add Live Connection
- Embedded 751-Fcj-Delete OAuth Client
- Embedded 752-Fcj-Delete Trusted IdP
- Embedded 753-Fj-Delete Live Connection

Most of these samples speak for themselves. All the samples are fully documented later in this user guide. Many though not all, of these 'general administration' scripts require a data file ('-F' in the name). Where a data file is needed, sample data files are provided making it easy to use as a template or just use as provided.

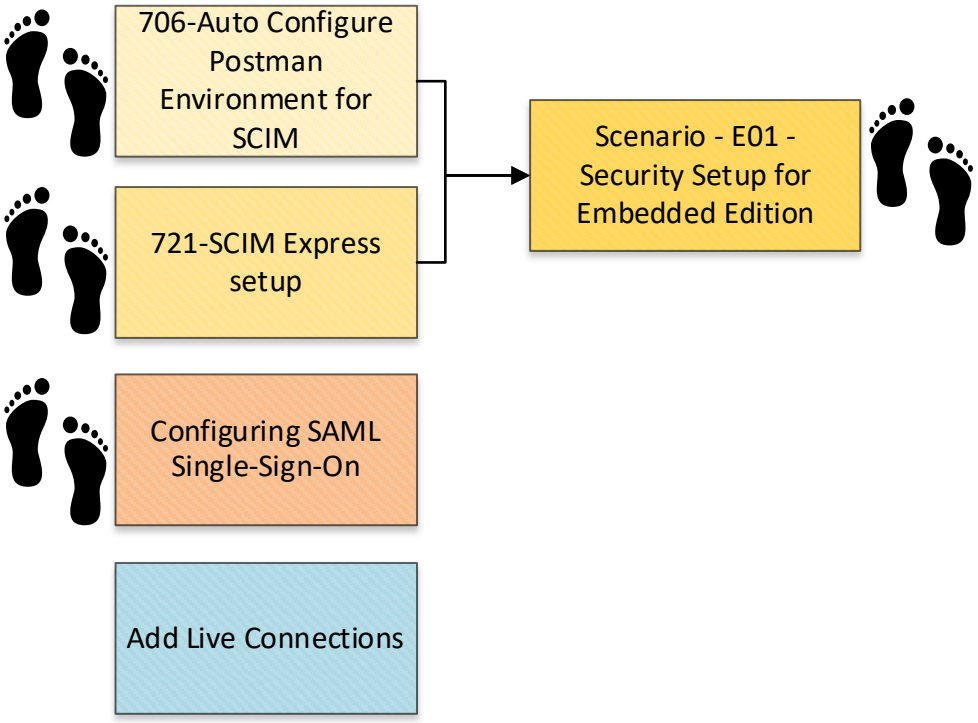
A summary and notes for some of these samples where it's not entirely obvious what they do:

- 731-E-Reset Inconsistent
  - If the tenant has an 'inconsistent' status, then this script will 'reset' it.
- 733-Fj-Configure Custom IdP
  - Configuring the Custom IdP requires exchanging metadata files. For the Embedded Edition the file cannot be uploaded, instead it must be 'json encoded' and included in an API request. This user guide provides comprehensive instructions for how to 'json encode' the metadata. It just requires a few 'global find and replace' commands on the text. See the detailed step-by-step instructions and the script documentation for details.
- 734-Fj-Update System Configuration
  - Enables you to update the various system parameters such as NR_PARALLEL_SESSION_FOR_BW. These parameters are typically the ones you would expect to see in the User Interface of the 'Enterprise Edition' via the menu-System- Administration-System Configuration.
- 735-Oarr-Fj-Update Trusted Origins
  - Enables you to update the list of trusted origin URLs. These are the ones you would expect to see in the User Interface of the 'Enterprise Edition' via the menu-System-Administration-App Integration.

Step-by-step instructions for 732 and 733 to setup SAML SSO are available in a following section

Official SAP documentation for the Embedded Edition 'Endpoints'

Configuration Order



The order of which you configure your SAP Analytics Cloud Embedded Edition is very flexible. Step-by-step instructions are provided for many of the initial setup procedures as shown by the feet icons above.

Configuring SAML SSO is optional, and this can be performed either before or after users have been created. However, in general, some issues can be avoided if users are created after SAML SSO has been setup.

The security setup of users, teams and roles involves several steps. These steps have been predefined for you in a 'Scenario'. 'Scenarios' is a very simple concept which comprises of nothing more than sets of pre-configured sample data files. Each scenario addresses a single use-case by combining different sample scripts (Postman collections) together in a particular order.

It means most of the thinking has been done for you. All you need to do is tweak the data files for your own needs, such as user ids, email addresses etc.

The 'Scenario – E01 – Security Setup for Embedded Edition' will setup your Embedded Edition with 5 teams (Admins, Users, Content_Admins, Content_Viewers, Content_Editors) being members of their respective roles and users being added to some of those teams. The prerequisite for the Scenario E01 is either sample scripts 706 or 721.

The only means to create users is via the SCIM API.

User assignment to teams, can be managed via the SAML SSO attribute mapping if you configure SAML SSO with your own custom Identity Provider.

The follow section provides step-by-step instructions for those highlighted above. Other tasks, such as Adding a Live Connection requires a simple modification to the provided sample data file before being run by the appropriate sample script.

Step-by-Step instructions

For a selected number of sample scripts, step-by-step instructions are provided.

Please remember that each sample script is fully documented in this user guide and sample data files are also provided where applicable. If a data file is needed, then very simple modifications are likely to suffice your need.

Auto Configure Postman Environment

If you are connecting to an instance of SAP Analytics Cloud Embedded Edition that has already been setup with an OAuth client, then there will be additional Postman Environment variables that need to be defined if you are using the SCIM samples. Different variables need to be defined depending upon your use-case, since each use-case can use a different OAuth client configuration, we need to ensure we use the right OAuth client if there are multiple. Thankfully there are sample scripts (Collections) that do this for you and so you don't need to do anything but run a Collection.

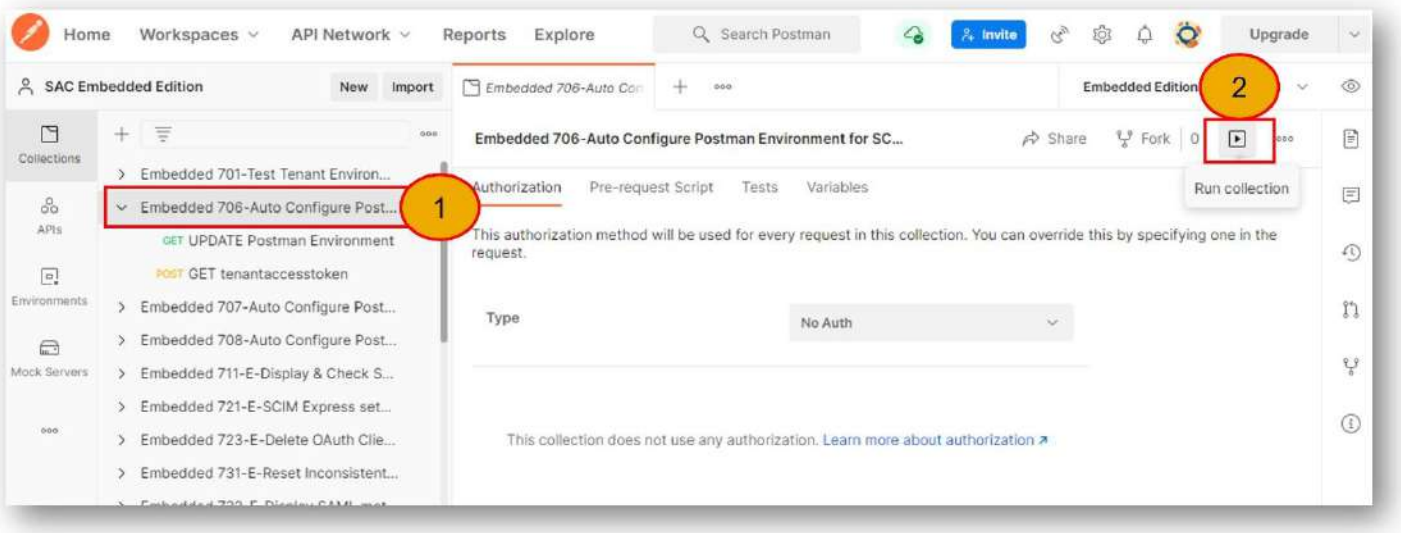
Run the sample script (Collection) that is relevant for you:

Use case	Sample Script (Collection) to run
User and Team provisioning & administration using the SCIM API	Embedded 706-Auto Configure Postman Environment for SCIM
Modelling API usage	Embedded 707-Auto Configure Postman Environment for Modelling
Story Listing API usage	Embedded 708-Auto Configure Postman Environment for Story Listing

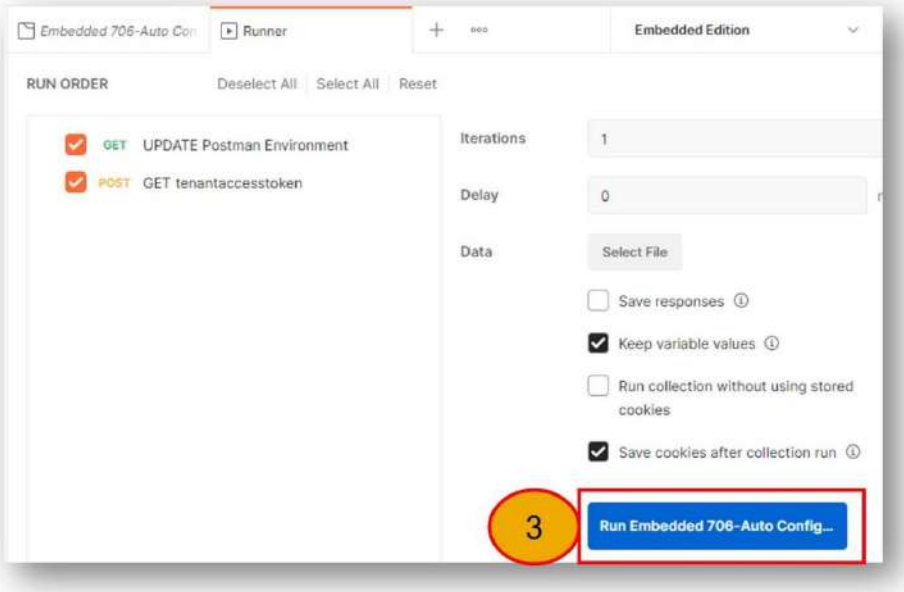
Each collection will read which OAuth clients have already been created and will use the one that is relevant for that use-case. For example the SCIM use-case will identify the name of the OAuth client that has a role PROFILE:sap.epm:SCIM_Public_API and the use that configuration to set other Postman Environment variables.

If you are unsure, use 706 since there are currently no sample scripts for Modelling or Story Listing. There are only sample scripts for the SCIM API at the moment, in addition to these sample scripts that help with the administration of your Embedded Edition.

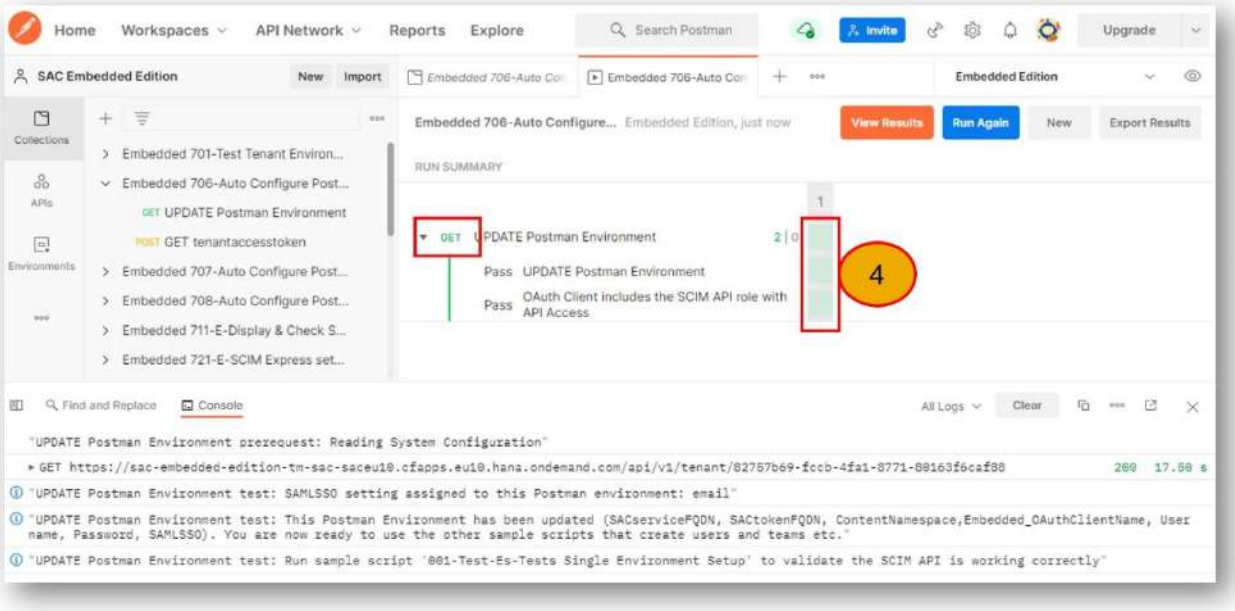
None of these sample scripts (706, 707, 708) require a data file and all are quite harmless. They don't make any changes to the configuration of the SAP Analytics Cloud Embedded Edition you are connected to. The step-by-step instructions are:



1. Select the Collection (in this example its "Embedded 706-Auto Configure Postman Environment for SCIM")
2. Press Run



3. (We don't need to select a data file for these 'auto configure' Collections) so just press Run



4. Open each of the tests and check that each test shows a 'Pass'

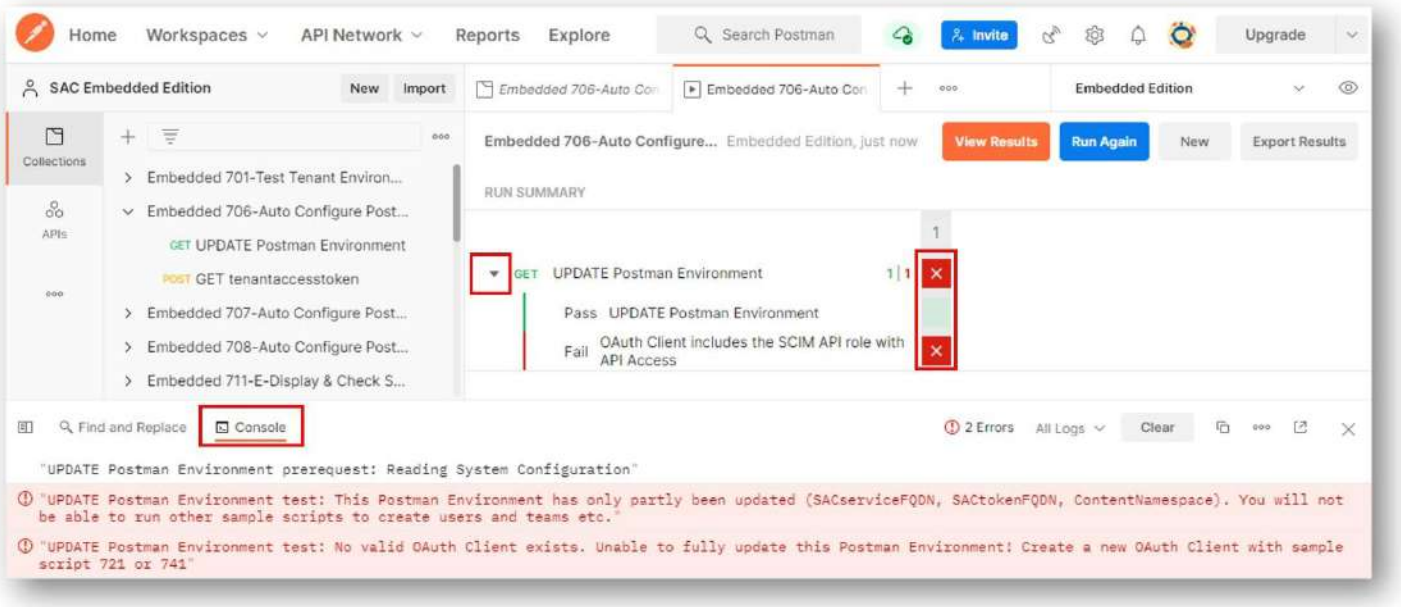
If successful, the console will show the following for script 706:

```
UPDATE Postman Environment prerequest: Reading System Configuration
UPDATE Postman Environment test: SAMLSSO setting assigned to this Postman environment: default
UPDATE Postman Environment test: This Postman Environment has been updated (SACserviceFQDN, SACtokenFQDN,
ContentNamespace, Embedded_OAuthClientName, Username, Password, SAMLSSO). You are now ready to use the other
sample scripts that create users and teams etc.
UPDATE Postman Environment test: Run sample script '001-Test-Es-Tests Single Environment Setup' to validate
the SCIM API is working correctly
```

Running the script will automatically configure the following variables within the currently selected Postman Environment: `SACserviceFQDN, SACtokenFQDN, ContentNamespace, Embedded_OAuthClientName, Username, Password, SAMLSSO`

Most of these variables are used by the SCIM API sample scripts that are documented in the "SAP Analytics Cloud User and Team Provisioning SCIM API Sample Scripts User Guide" (and this includes the script '001-Test-Es-Tests Single Environment Setup' mentioned in the console log). It means that much of the configuration for the SCIM API sample scripts has now been completed for you. This then means you must skip steps A, B and most of C in that document. Instead go directly to step 24 (within Step C) to configure the TimeZone variables.

Should the 706, 707 or 708 Collections fail they will look like this:



The console will show this for 706:

```
UPDATE Postman Environment prerequest: Reading System Configuration

UPDATE Postman Environment test: This Postman Environment has only partly been updated (SACserviceFQDN,
SACtokenFQDN, ContentNamespace). You will not be able to run other sample scripts to create users and teams
etc.

UPDATE Postman Environment test: No valid OAuth Client exists. Unable to fully update this Postman
Environment! Create a new OAuth Client with sample script 721 or 741
```

It means an OAuth client for the use-case you've chosen hasn't yet been created and you will need to create one with scripts 721 or 741. Script 721 is used in the next section 'Express Setup - First time setting up...'

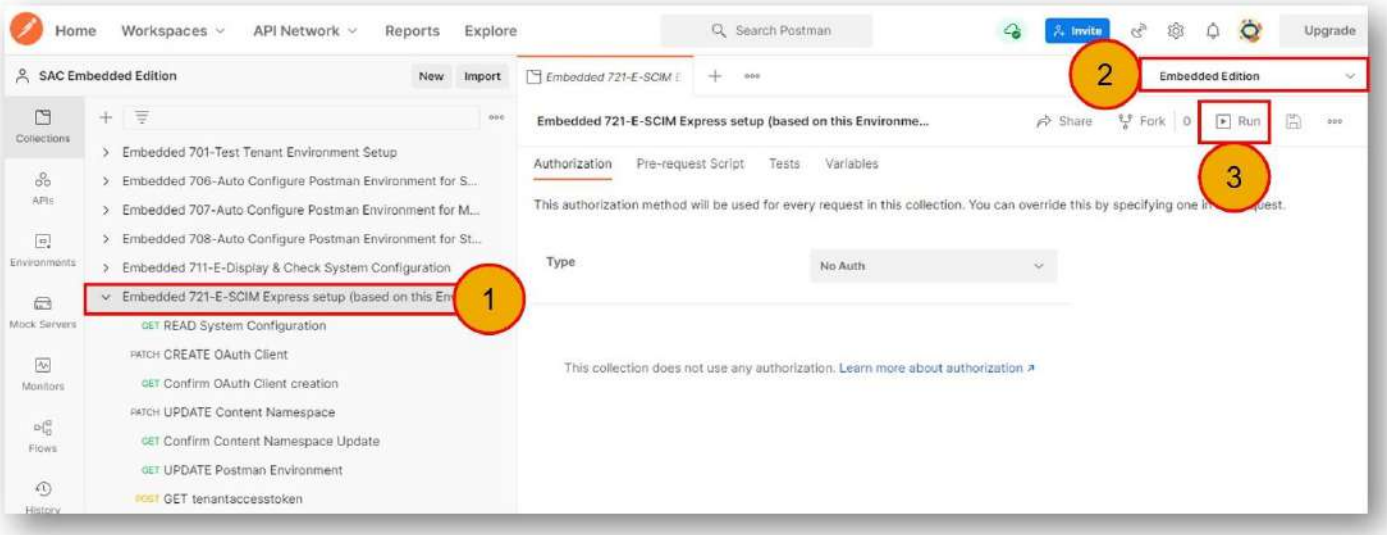
Express Setup - First time setting up this instance of SAP Analytics Cloud Embedded Edition

Configuring SAP Analytics Cloud Embedded Edition requires several tasks to be completed. To ease the setup and merge a few tasks into just 1, a single collection called “Embedded 721-E-SCIM Express setup (based on this Environment)” is available. This will do the following for you:

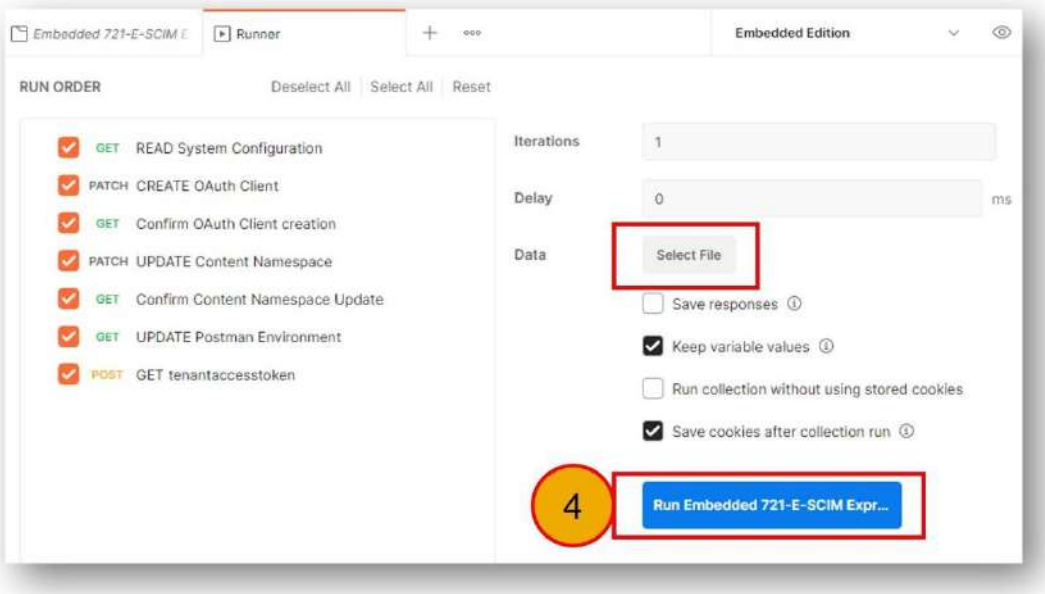
1. Create an OAuth Client for you.
    a. The OAuth client will have all the necessary permissions for API access and the role (PROFILE:sap.epm:SCIM_Public_API) that is needed to create and manage users and teams. The OAuth client added will not enable you for Modelling or Story access. (If you need an OAuth Client for these other reasons, then please refer to sample script 741 and the provided sample data file that will do this for you. Then use script 707 or 708 to update the Postman Environment automatically)
    b. The name of the OAuth client will be the one already defined in your Postman Environment variable called ‘Embedded_OAuthClientName’ and has value ‘**MyOAuthClient**’. Feel free to change this name before running sample script 721
2. Update the Content Namespace of this Embedded Edition to the name as defined in the Postman Environment.
    a. Changing the Content Namespace isn’t necessary, but it is considered best practice to keep the Content Namespace the same and consistent across the landscape. For more details, please refer to this blog. For this reason, the Content Namespace is updated as part of this ‘express setup’.
    b. If you want to keep the existing Content Namespace that SAP assigned to your Embedded Edition, then use sample script 711 to display your current setting for this, look for ‘Content Namespace’ at the end of the console display. Copy this value into your Postman Environment variable ‘ContentNamespace’. Otherwise, your Embedded Edition will be updated to whatever the current value in your Postman Environment variable ‘ContentNamespace’ is. The default is ‘**t.1**’
3. Update other Postman Environment variables so the SCIM API Sample Scripts work without any further need to configure them (unlike when using the SAP Analytics Cloud Enterprise Edition).

It will do this all this without a data file and so there’s no need to specify or configure one.

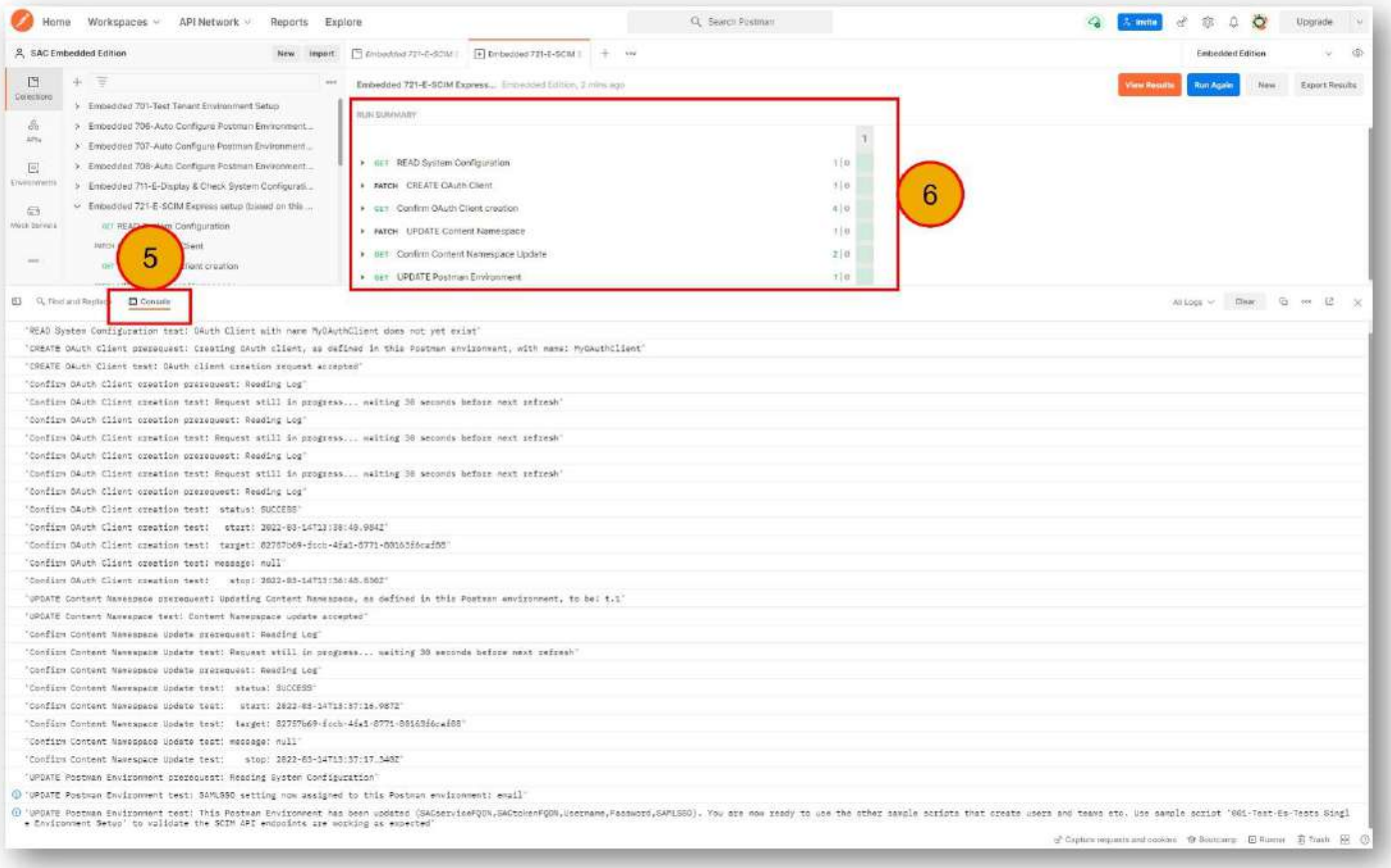
Here are the step-by-step instructions:



1. Select the collection “Embedded 721-E-SCIM Express setup (based on this Environment)”
2. Ensure the correct Environment is selected before step 3
3. Press Run



4. This Collection does NOT need a data file so there's no need to select one (unlike a number of others), just press Run



5. Open the Console and observe the progress of the script.
    a. It will take a between 2 and 5 minutes to run
    b. It first creates a new OAuth Client for SCIM use only
    c. The Content Namespace is then updated to be ‘t.1’ (assuming you’ve not change the Postman Environment variable ‘ContentNamespace’)
    d. Finally, the Postman Environment is updated so the SCIM API samples will work.
    e. The console will show:

```
SAMLSSO setting now assigned to this Postman environment: default
UPDATE Postman Environment test: This Postman Environment has been updated
(SACserviceFQDN,SACtokenFQDN,Username,Password,SAMLSSO). You are now ready to use the
other sample scripts that create users and teams etc. Use sample script '001-Test-Es-
Tests Single Environment Setup' to validate the SCIM API endpoints are working as
expected
```

6. The run summary should show each test passes.

Your next tasks, which can be performed in any order, are likely to be:

1. Create users and teams
    a. Which will then mean you can use the user interface to login. The URL will be the hostname that has been automatically set for you in your Postman Environment variable **SACserviceFQDN**
    b. The step-by-step instructions for this are provided in this guide the next section ‘Security Setup’
2. Add a new HANA live connection
3. Define a custom IDP

Security Setup (Scenario – E01)

The Security Setup Scenario uses the SCIM API, and this means you must either have successful run one of these sample scripts described above:

1. 'Embedded 721-E-SCIM Express setup (based on this Environment)'
2. or 'Embedded 706-Auto Configure Postman Environment for SCIM'

You must also have setup the SCIM API sample scripts following the "SAP Analytics Cloud User and Team Provisioning SCIM API Sample Scripts User Guide". Because you've already run the sample scripts 721 or 706 much of the configuration for the SCIM API sample scripts has now been completed for you automatically. This then means you must skip steps A, B and most of C in that document. **Instead go directly to step 24** (within Step C) to configure the TimeZone variables. Once you have successfully completed step 39 of that setup, you can proceed here.

Scenario	E01 - Security Setup for Embedded Edition
Purpose	Setup the security for the SAP Analytics Cloud Embedded Edition
Description	Creates an initial 'setup' user in SAP Analytics Cloud Embedded Edition for you to login to the user interface. This will allow you to create a number of teams manually to avoid the need to create team folders. (Team folders may be problematic) Then roles can be assigned to the teams to conform to best practice. The initial 'setup' user is then updated so the previously assigned role is assigned to that user via the teams, rather than directly to the user. Regular users can then be created and assigned to the teams inheriting the roles via those teams.

You will have already downloaded the data files for this scenario. They will be in the folder 'Embedded\Scenarios\Scenario E01 - Security Setup for Embedded Edition'

Enable manually created teams to be accessed via the API

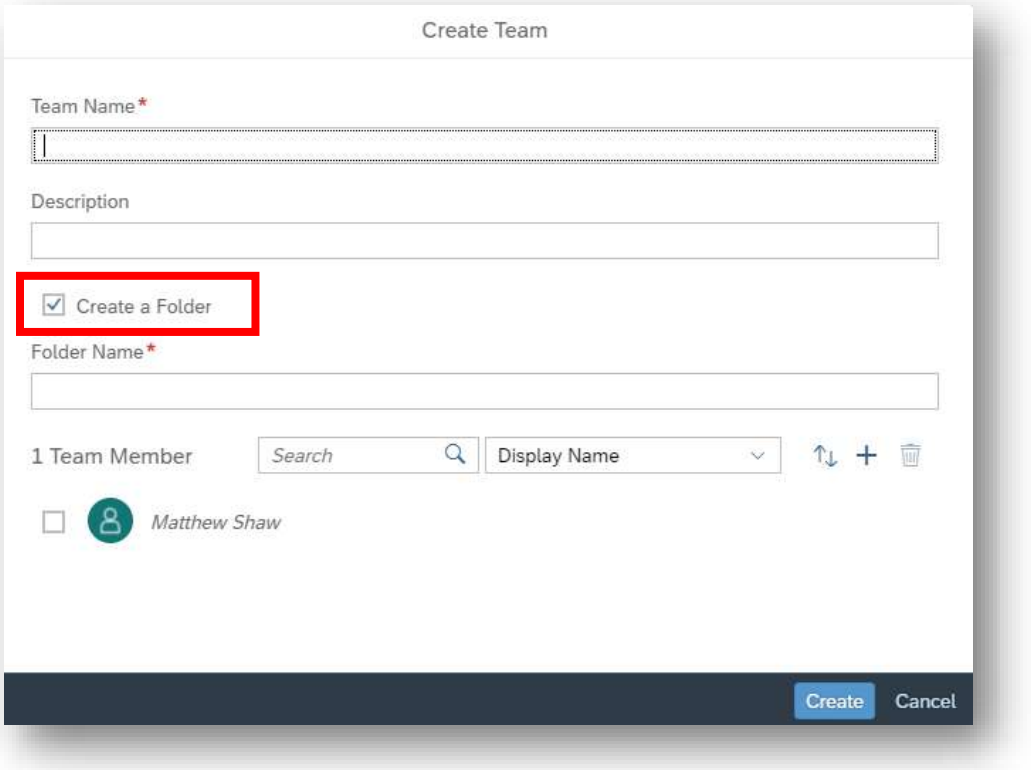
To follow best practices, it is advised not to assign roles directly to users. Instead, it is advised to add users into a team and assign the team to roles. It means the users inherit right role rights through the team membership. Teams are also a very good means to manage security and rights assignment since they ease administration overheads by allowing inheritance to work so users within a team inherit appropriate rights etc. Assigning roles to teams will also enable SAML attribute mapping of users to teams since with the Embedded Edition SAML attribute mapping to roles directly is not possible.

However, when a team is created an optional 'team folder' can be created with it. These 'team folders' for the embedded edition cannot be accessed, nor can they be managed in anyway and this includes deleting them. It means such team folders have no role within the embedded edition. This isn't an issue in general, since it is best practice not to use such folders but instead create public folders and assign rights appropriately to these public folders.

When creating teams via the API, the team folder is always created as the API was developed before the feature that a team folder was optional. This could cause an issue, if you decide to delete the team, leaving the team folder in place

even though you can't access it! It could be a problem because you won't be able to create a new team of the same name since the team creation will fail. The team create will fail because you cannot have 2 team folders of the same name (the old team folder and the new one).

In turn, it means the best practice would be to only create teams manually and NOT via the API. When creating teams manually always create the team WITHOUT the team folder. You should de-select the option as shown:



When a team is created manually via the user interface, that team cannot be managed via the API. The fix is quite straightforward, simply log a Support Incident with SAP Product Support and ask for the business toggle **IMPLEMENT_WORKAROUND_FOR_SCIM_GROUPS** to be turned on. Mention SAP KBA 2857395 and the SAP Analytics Cloud Service URL in the incident. This will then mean all teams created via the user interface can be used and managed via the API. For more details please refer to this blog.

There is no harm in having teams with folders, it's just that it will prevent you, after you've deleted a team, from re-creating it with the same name. If you are happy with this, then you may skip steps 1, 2 and 3 and proceed directly to step 4.

Step 1: 132 Create setup user with roles

- Sample data file: **Step 01 - 132 Create setup User with roles.json**
- Edit the data file and replace the personal entries to match your own. Ensure the email is correct. The data file contains:

```
[
  {
    "file_userid": "MATTHEW",
    "file_isconcurrent": false,
    "file_givenname": "Matthew",
    "file_familyname": "Shaw",
    "file_displayname": "Matthew Shaw",
    "file_email": "matthew@sap.com",
    "file_managerid": "",
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Admin"}],
    "file_JSON_teams": [],
      "file_roles_action": "add",
      "file_teams_action": "keep"
  }
]
```

- Run the collection "132-U-CU-CUlemr-Oarrk-Fcj-Es-SAML Create/update users (roles, no teams)" with this data file

At the end of step 1, you will be able to login to the user interface of SAP Analytics Cloud. Use the URL that has been set for you in your Postman Environment variable called **SACserviceFQDN**.

If you are using the default Identity Provider that comes with SAP Analytics Cloud, on the login page, press the '**Forgot Password?**' button for a reset password email to be sent to you. Once your password is reset, you can login. You will have the role 'Embedded BI Admin'.

Step 2: Enable business toggle IMPLEMENT_WORKAROUND_FOR_SCIM_GROUPS

- Log a Support Incident with SAP Product Support and ask for the business toggle IMPLEMENT_WORKAROUND_FOR_SCIM_GROUPS to be turned on. Mention SAP KBA 2857395 and the SAP Analytics Cloud Service URL in the incident.

Step 3: Create teams manually

Once SAP Product Support have confirmed the business toggle IMPLEMENT_WORKAROUND_FOR_SCIM_GROUPS is ON, then and only then create 5 teams but importantly without creating the team folder when you create the team.

You should deselect the 'Create a Folder' option.

Create 5 teams with the names:

- Admins
- Users
- Content_Admins
- Content_Viewers
- Content_Editors

There is no need to specify the team description as this will be set in the next step, any descriptions provided will be lost.

The team names are used in the next steps, so if you give them different names you must change their names in the subsequent steps too.

Step 4: Update team descriptions

- Sample data file: **Step 04 - 501 Update teams description.json**
- Run the collection “501-T-UC-Ud-Fcj-Es-Update create team” with this data file

Once the script is run, the descriptions for the 5 teams, you just created manually in step 3, will be updated.

If you skipped step 1, 2 and 3 then this step, with the same sample data file and collection, will instead create 5 new teams with the correct description. Each team will also have a hidden team folder that you won’t be able to delete.

The data file contains:

```
[
  {
    "file_team": "Admins",
    "file_displayname": "Users with the role Embedded_BI_Admin"
  },
  {
    "file_team": "Users",
    "file_displayname": "Users with the role Embedded_BI_User"
  },
  {
    "file_team": "Content_Admins",
    "file_displayname": "Users with the role Embedded_BI_Content_Admin"
  },
  {
    "file_team": "Content_Viewers",
    "file_displayname": "Users with the role Embedded_BI_Content_Viewer"
  },
  {
    "file_team": "Content_Editors",
    "file_displayname": "Users with the role Embedded_BI_Content_Editor"
  }
]
```

Step 5: 612 Add roles to teams

- Sample data file: **Step 05 - 612 Add roles to Teams.json**
- Run the collection "612-T-Uc-Uur-Oarrk-Fj-Es users/roles actions on Teams" with this data file

The data file specifies the Roles that each team will be added to:

Team name	Role team will be a member of
Admins	Embedded_BI_Admin
Users	Embedded_BI_User
Content_Admins	Embedded_BI_Content_Admin
Content_Viewers	Embedded_BI_Content_Viewer
Content_Editors	Embedded_BI_Content_Editor

The data file contains:

```
[
  {
    "file_team": "Admins",
    "file_users_action": "keep",
    "file_roles_action": "add",
    "file_JSON_users": [],
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Admin"}]
  },
    {
    "file_team": "Users",
    "file_users_action": "keep",
    "file_roles_action": "add",
    "file_JSON_users": [],
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_User"}]
  },
    {
    "file_team": "Content_Admins",
    "file_users_action": "keep",
    "file_roles_action": "add",
    "file_JSON_users": [],
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Content_Admin"}]
  },
    {
    "file_team": "Content_Viewers",
    "file_users_action": "keep",
    "file_roles_action": "add",
    "file_JSON_users": [],
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Content_Viewer"}]
  },
    {
    "file_team": "Content_Editors",
    "file_users_action": "keep",
    "file_roles_action": "add",
    "file_JSON_users": [],
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Content_Editor"}]
  }
]
```

Once the script 612 has been run with this data file, each team will be a member of a role as specified above. It means that when a user is a member of the team, they will inherit the respective role. Users can be in more than 1 team, and indeed a team can also be a member of multiple roles.

Step 6: 233 Remove roles from user and add teams

- Sample data file: **Step 06 - 233 Remove roles from user and Add teams.json**
- You will need to update the data file, shown below, to match your own details and in particular the email address
- Run the collection "233-U-UC-UClemrt-Oarrk-Fj-Es-SAML Update/create users (with roles & teams)" with this data file

The data file contains the following:

```
[
  {
    "file_userid": "MATTHEW",
    "file_isconcurrent": false,
    "file_givenname": "Matthew",
    "file_familyname": "Shaw",
    "file_displayname": "Matthew Shaw",
    "file_email": "matthew@sap.com",
    "file_managerid": "",
    "file_JSON_roles": [{"value":"PROFILE:sap.epm:Embedded_BI_Admin"}],
    "file_JSON_teams": [{"value":"Admins"}],
      "file_roles_action": "remove",
      "file_teams_action": "add"
  }
]
```

Once the script 233 has been run with this data file, the initial 'setup' user that was created in step 1 of this scenario, will:

- have the roles that where directly assigned removed
- be assigned as a member of the 'Admins' teams. This team is already members of their respective role and so there is not net difference with the right this 'setup' user has assigned. It's just we are now following best practices by assigning roles to teams, rather than directly to users.

If you skipped steps 1, 2 and 3, then:

- This step 6 will create a new user with 'Admin' rights. You will need to update the data file, shown above, to match your own details and in particular the email address
- Once you can run sample collection 233 with your data file, you will be able to login to the user interface of SAP Analytics Cloud. Use the URL that has been set for you in your Postman Environment variable called **SACserviceFQDN**.
- If you are using the default Identity Provider that comes with SAP Analytics Cloud, on the login page, press the '**Forgot Password?**' button for a reset password email to be sent to you. Once your password is reset, you can login. You will have the role 'Embedded BI Admin'.

Step 7: 123 or 133 Create regular users with teams

- Sample data file: **Step 07 - 123 or 133 Create regular Users with teams.json**
- Edit the data file, shown below, with your own users, their names and email addresses etc. The sample adds the users to the 'Users' team, but you may wish to add them to other teams. For example, adjust the entry `file_JSON_teams` to include the 'Admins' team instead of the Users for your administrator users. Equally use the other teams for Content Admins, Creators, Editors as you see fit.
- Run one of these collections with this data file:
  - 123-U-CU-CUlemrt-Oarrk-Fcj-Es-Create/update users (roles & teams)
  - 133-U-CU-CUlemrt-Oarrk-Fcj-Es-SAML Create/update users (roles & teams)
    If you are unsure, use 133.
- Once the script 123 or 133 ha been run with this data file, these users will be created and added to the team as mentioned.

The data file contains the following:

```
[
  {
    "file_userid": "MATTHEW1",
    "file_isconcurrent": false,
    "file_givenname": "Matthew",
    "file_familyname": "Shaw 1",
    "file_displayname": "Matthew Shaw 1",
    "file_email": "Matthew+1@sap.com",
    "file_managerid": "",
    "file_JSON_roles": [],
    "file_JSON_teams": [{"value":"Users"}],
      "file_roles_action": "keep",
      "file_teams_action": "add"
  },
{
    "file_userid": "MATTHEW2",
    "file_isconcurrent": false,
    "file_givenname": "Matthew",
    "file_familyname": "Shaw 2",
    "file_displayname": "Matthew Shaw 2",
    "file_email": "Matthew+2@sap.com",
    "file_managerid": "",
    "file_JSON_roles": [],
    "file_JSON_teams": [{"value":"Users"}],
      "file_roles_action": "keep",
      "file_teams_action": "add"
  }
]
```

The only means to add users is via the SCIM API.

User assignment to teams, can be managed via the SAML SSO attribute mapping.

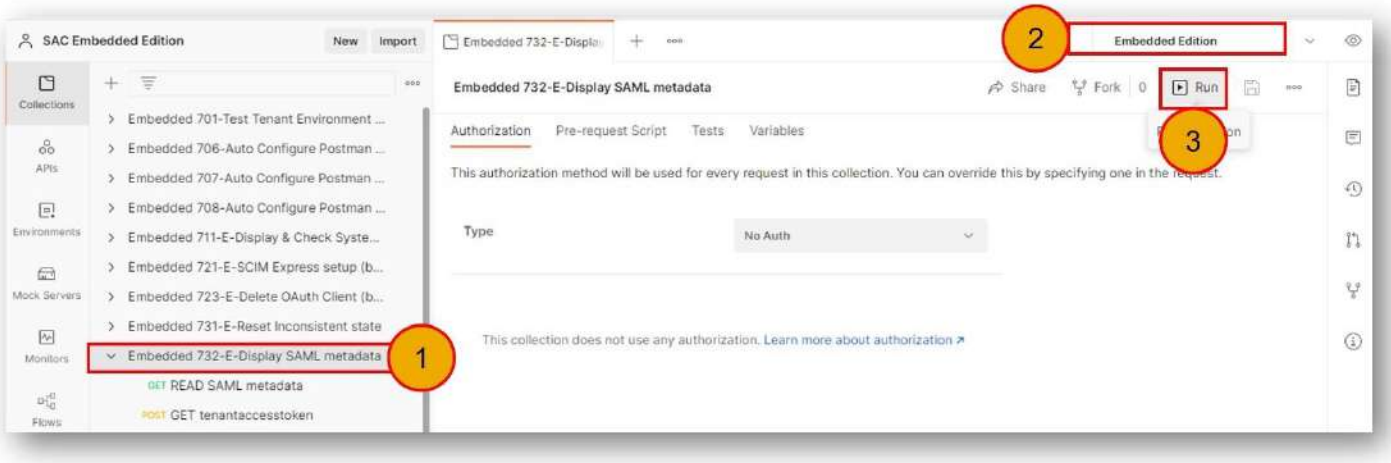
Configuring SAML Single-Sign-On

The configuration of the SAML SSO can be performed either before or after any users have been created in SAP Analytics Cloud. It means the 'Express setup' and the Scenario 'E01 - Security Setup for Embedded Edition' are not prerequisites.

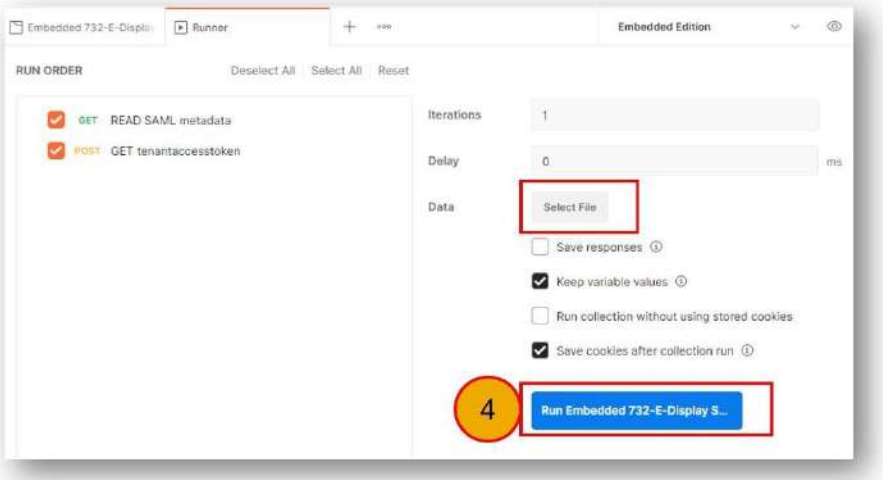
To enable the trust relationship between the SAP Analytics Cloud Embedded Edition and your own Custom Identity Provider an exchange of metadata files is required:

- download the meta data file from SAP Analytics Cloud – steps 1 to 6 shown below
- upload this meta data file (from SAC) to your custom Identity Provider – step 7
- download the meta data file from your custom Identity Provider – step 8
- json encode the meta data file – steps 9 to 13
- upload this meta data file (from your IdP) to SAP Analytics Cloud Embedded Edition – steps 14 to 19

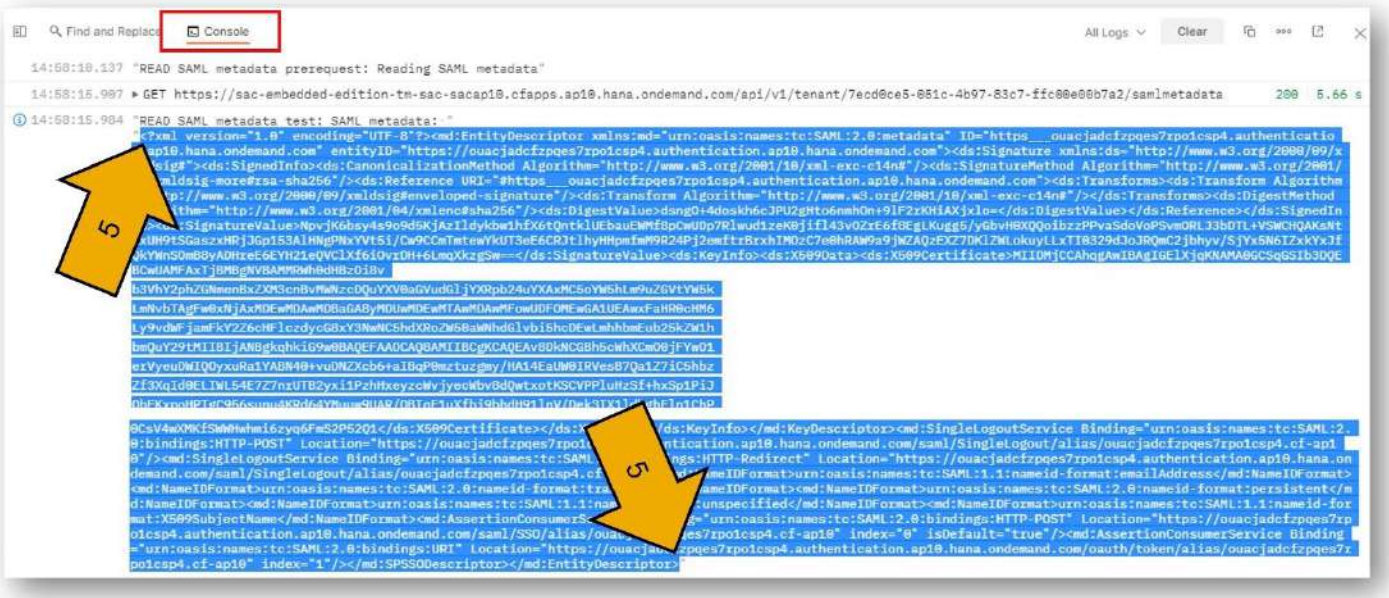
The step-by-step instructions are:



1. Select the sample Collection 'Embedded 732-E-Display SAML metadata'
2. Ensure the environment is selected correctly
3. Press Run



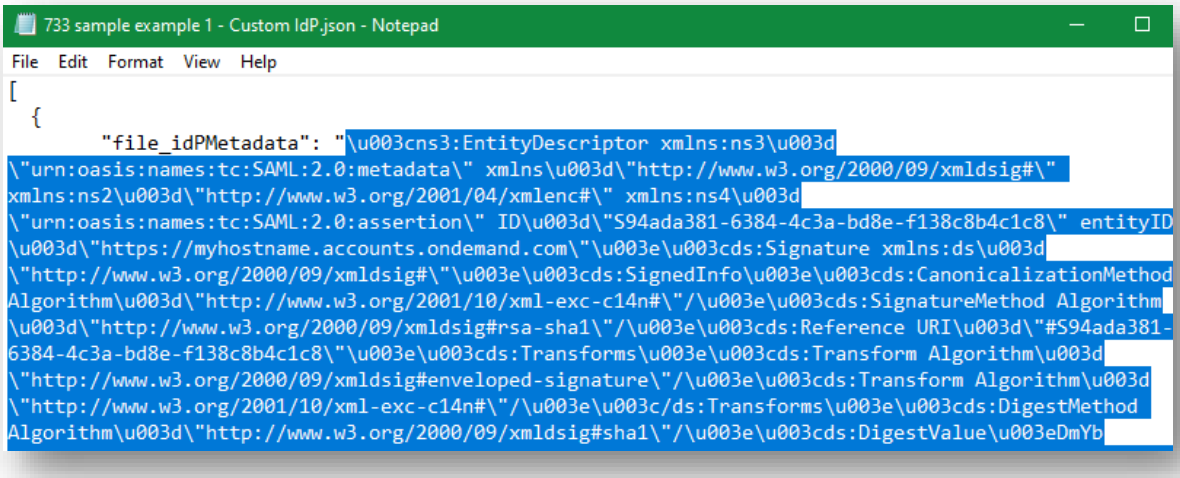
4. No data file is needed, so press Run



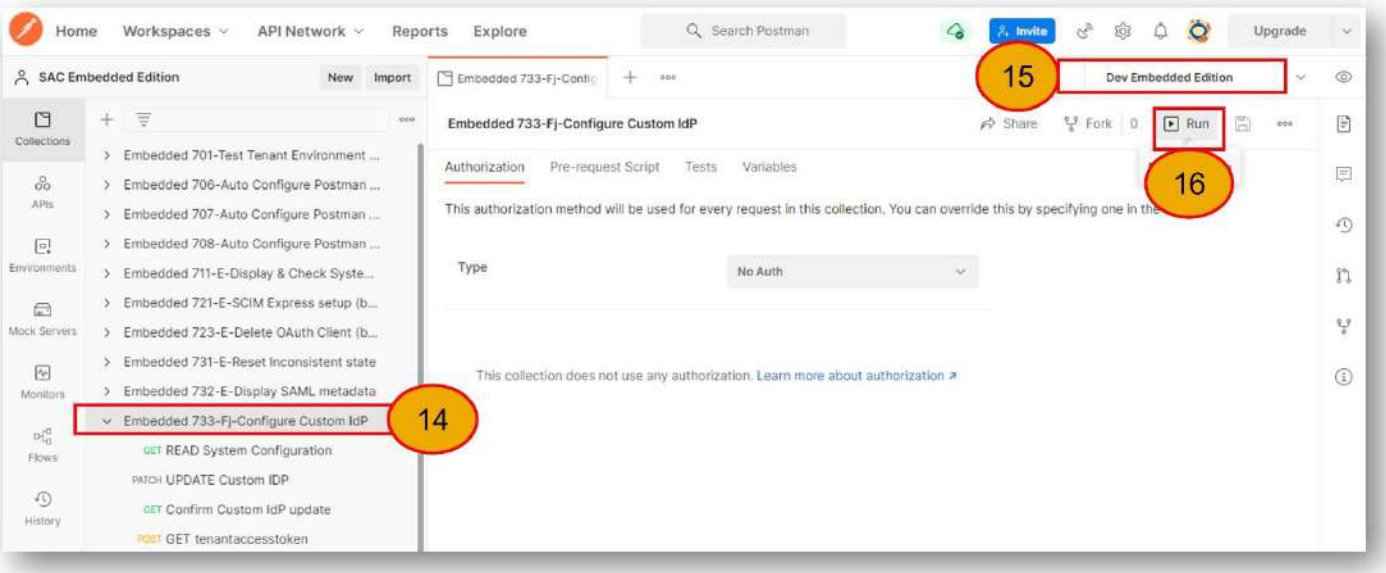
5. Open the console and select the entire contents of the output shown and copy it to the clipboard. Be careful not to select the leading " and trailing ".
6. Create a new empty file called 'SAC metadata.xml' and paste the entire contents of what you just copied into your clipboard, into this new empty file. Save the file.
7. Upload this 'SAC metadata.xml' file to your Identity Provider and save any changes needed there.
8. Download the SAML 2.0 metadata.xml configuration file from your Custom Identity Provider and save the file locally, call the filename 'My IdP metadata.xml' (or similar)
9. Open the 'My IdP metadata.xml' file you downloaded in step 8 with a very basic text editor (avoid sophisticated text editors that can understand regular expressions, or match only on whole words, you want to use the very basic editor possible). Perform 4 global find and replace commands on the entire contents of the file and in the order shown. This will 'json encode' the file. If you don't do this properly or carefully, then Postman won't be able to read the entry correctly.

Find all	& Replace with
"	\"
=	\u003d
<	\u003c
>	\u003e

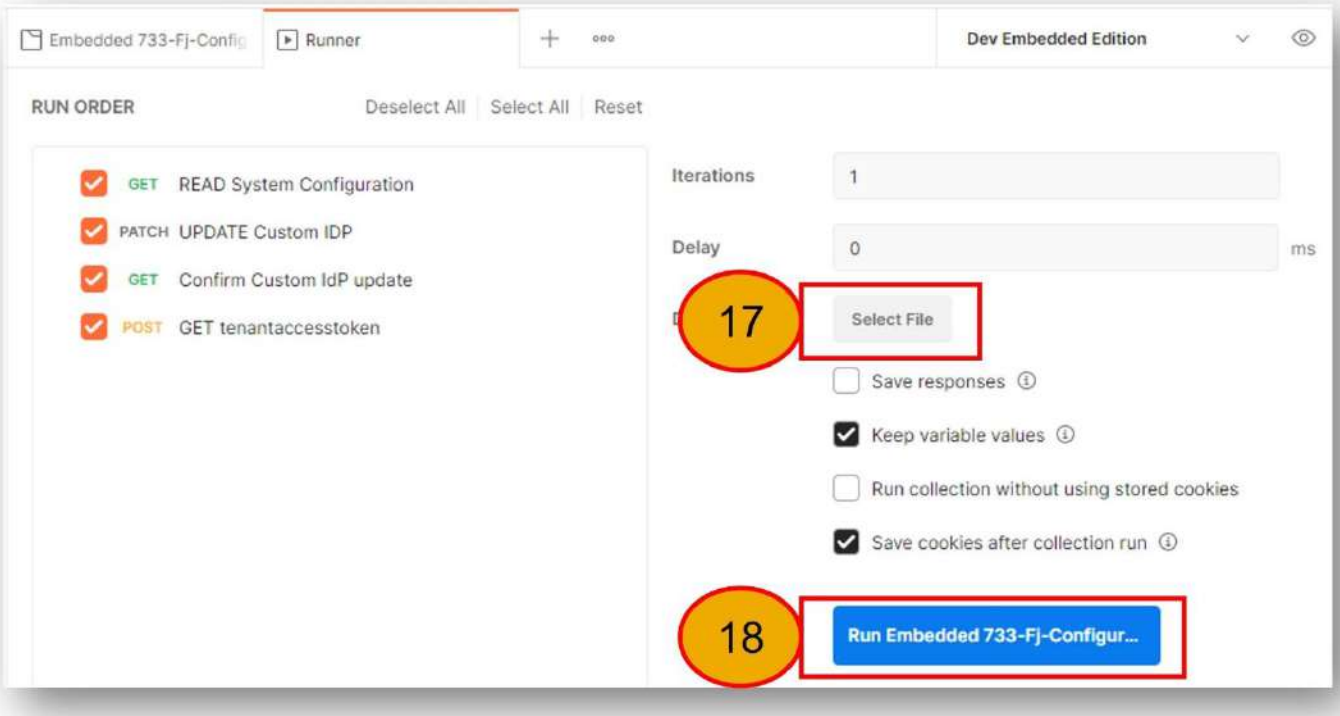
10. Open the sample data file '`733 sample example 1 - custom idp.json`' and edit this file in your text editor



11. For the value of the json variable **file_idPMetadata**, insert/paste the result of step 9 where you performed 4 global find and replace commands on the 'My IdP metadata.xml' file. You have now successfully 'json encoded' this entry. It means Postman will be able to read a json file containing a json file!
12. For the value of the json variable **file_nameIdColumn** enter values of **userid**, **email** or **custom** to reflect the property you are using for the Subject ID.
13. Save the file you edited in step 11 and 12.



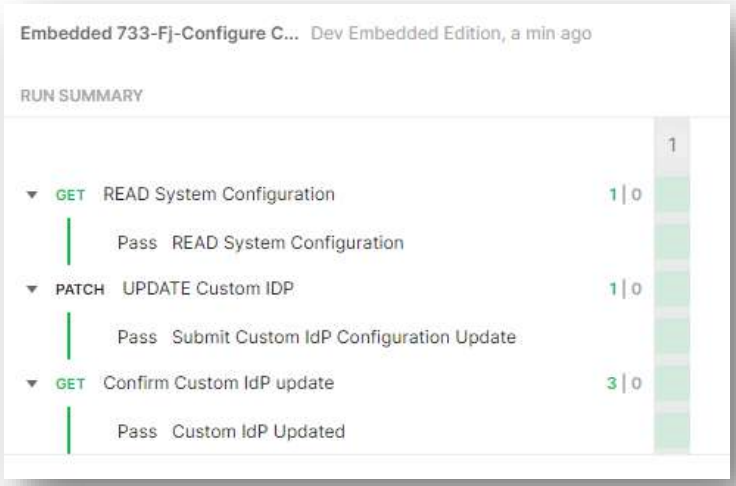
14. Select the sample Collection 'Embedded 733-Fj-Configure Custom IdP'
15. Check the environment is selected correctly
16. Press Run



17. Select the file you saved in step 12 (‘*733 sample example 1 - custom idp.json*’)
    a. You may optionally select the ‘Preview’ button that appears once you’ve successfully selected the file. It should show what Postman sees inside the file. You can see its un-encoded the json file:

    b. If there is an error, then your json encoding was not correct. Start again from step 8 again and be sure to use a very basic text editor. Take your time, the steps are detailed and complex.
18. Press Run
    a. It will take a few minutes to complete. Open the console to show the progress of the script.
19. Once successful you should observe the following:

a. The test results all pass:



b. The console shows success, similar to:

```
READ System Configuration prerequest: Reading System Configuration
Current Settings for Custom IdP: Enabled: false    Entity ID : undefined    Name
ID column : undefined
READ System Configuration test: Proceeding to request Custom IdP update
UPDATE Custom IDP prerequest: Updating Custom IdP
UPDATE Custom IDP test: Custom IdP Configuration Update request accepted
Confirm Custom IdP update prerequest: Reading Log
Confirm Custom IdP update test: Request still in progress... waiting 30 seconds
before next refresh
Confirm Custom IdP update prerequest: Reading Log
Confirm Custom IdP update test: Request still in progress... waiting 30 seconds
before next refresh
Confirm Custom IdP update prerequest: Reading Log
Confirm Custom IdP update test:  status: SUCCESS
Confirm Custom IdP update test:   start: 2022-03-17T16:42:28.074Z
Confirm Custom IdP update test:  target: 7ecd0ce5-051c-4b97-83c7-ffc00e00b7a2
Confirm Custom IdP update test:    stop: 2022-03-17T16:42:50.323Z
Confirm Custom IdP update test: message: null
Confirm Custom IdP update test: Custom IdP update was successful
Confirm Custom IdP update test: Now a Custom IdP has been enabled, it is not
possible to revert back to the default setting and remove the Custom IdP. It is
only possible to make alterations to the existing Custom IdP configuration
Confirm Custom IdP update test: SAMLSSO setting now assigned to this Postman
environment: email
```

The Postman Environment variable ‘SAMLSSO’ will be updated automatically to match the option you specified in the data file for the variable file_nameIdColumn back in step 12. This means the SCIM API sample scripts will adjust their workflow accordingly and there’s no need to update this variable yourself.

You have now successfully configured SAML SSO for SAP Analytics Cloud Embedded Edition.

If you have created a user in SAP Analytics Cloud Embedded Edition, you should now be able to login to the user interface authenticate against your Custom Identity Provider. Open a new browser tab and navigate to the URL that has been set for you in your Postman Environment variable called SACserviceFQDN (assuming you’ve already run sample script 706 or 721)

The NameID property, returned from your Custom Identity Provider, will need to match the userName property in SAP Analytics Cloud for a user to logon successfully. This userName property is different depending upon the **nameIdColumn** you specified back in step 12:

- 'userid' then the SAP Analytics Cloud user id will be the SAML mapping (userName) value.
- 'email' then the SAP Analytics Cloud email will be the SAML mapping (userName) value.
    - You may need to update any existing users with the correct email so the match works. Sample scripts 20x, 409, 419 and 429 will do this for you.
- 'custom' then the SAP Analytics Cloud SAML mapping (userName) property can be update independently of all other user properties.
    - You may need to update any existing users you have already created. Sample scripts 409, 419 and 429 will do this for you. For new users you should use sample scripts 13x and 23x and specify the saml_mapping property in the data file.

Please refer to the SCIM API Sample Script documentation for more details about those sample scripts.

Official SAP documentation for Enabling a Custom SAML Identity Provider

Scenarios

Scenarios is a very simple concept which comprises of nothing more than sets of pre-configured sample data files. Each scenario addresses a single use-case by combining different sample scripts (Postman collections) together in a particular order.

It means most of the thinking has been done for you. All you need to do is tweak the data files for your own needs, such as the email address, user ids etc.

The summary of the scenarios is as follows:

Scenario	Description / use-case
E01 - Security Setup for Embedded Edition	Creates 5 teams, each is assigned to 5 different roles. Users are then added to these teams

The Scenario E01 is fully documented in the step-by-step instructions above.

Collection behaviour and design

A common design has been taken to all the sample collections and its worth understanding these aspects to appreciate their behaviour.

Postman tests

The collections have been designed with Postman tests, for example 'OAuth Client Created' tests will pass if the OAuth Client was successfully created. The sample 'Embedded 711-E-Display & Check System Configuration' has a significant number of tests to quickly highlight what is and what's configured. Not all tests necessarily need to pass in this instance but do in all other instances.

General handing of errors

As mentioned in the 'Postman tests' above, the scripts have been designed to handle different responses from the API, for example many of the request's return a 'log' that needs to be read until the task has completed. The scripts repeatedly read that 'log' until the task completes. Many of the samples perform various tests before making a request to the API so to ensure success. Any errors are then displayed in the console, so it's worth always looking at the console as the scripts are run. The scripts are also designed to manage other aspects including various types of errors:

- Session timeouts. Each accesstoken has a maximum lifetime. All the collections capture and manage the expiry of the accesstoken. When it expires, a new accesstoken will be obtained and the original request will be re-submitted. It means, just because the sessions times-out mid operation, it has no effect on the net result. The console log will report a session timeout and you'll see from the console a new session will be established.
- Any un-expected error will typically result in 2 re-attempts of the same request. This is helpful when there's an occasional 'wobble' somewhere and there's nothing wrong with the request being sent, but something else went wrong somewhere.
- Endless loops are avoided. There are two main endpoints, the Tenant API itself and the token endpoint. Each endpoint is somewhat independent of each other, meaning one could be responsive and the other not. One may be non-responsive for all manner of reasons. Just because one endpoint is returning un-expected results could trigger a call to the other endpoint, for example to obtain a new accesstoken. The fetch of a new accesstoken is likely to be valid but that may not resolve the error on Tenant API endpoint. In such conditions, this 'loop' is detected, and the script will exit that loop and move onto the next user, or team. The console log will report such errors including when a 'loop' of errors has been detected. Endless loops are very rare.

Sessions, tokens and sharing of tokens across collections

There are two tokens that are managed by the samples:

1. **accesstoken**: this token to obtained from the token endpoint and is re-presented in the authorisation header of the SCIM endpoint.

In all cases, the accesstoken values are stored in the Postman environment which means if you run multiple collections, these tokens will be re-used across them. This will save time as there's often no need to re-establish a session if the existing one can be re-used. It means, if you run a collection and it suddenly times-out, even if you're only just started it a few moments earlier. This will happen when you previously run another collection earlier that day, but perhaps forgot the session would still be valid. In all cases, you don't need to worry about the session timeout of this token, the samples manage their life cycle and recover when they expire, regardless of when or where it occurs in the script.

Command line interface

Postman provides a command line interface that will also these scripts to run without the need for the user interface. In general, this command line option should be used as it provides improved stability and potentially performance improvements too. For more details, please visit the Postman web site and its associated community pages.

SAP HANA Academy provide a step-by-step tutorial for the command line option
https://youtu.be/UfmauS2xagk

Miscellaneous

When things go wrong

There are some best practices when things appear to be misbehaving! Typically, and in almost every case, it will be a client-side issue.

It's recommended you do the following:

1. Check you're using the right data file for the collection (very common mistake!)
2. Check you've selected the correct environment before you start the Postman runner (very common mistake!)
3. Inspect the console log for clues to what is wrong
4. Restart Postman
5. Use the option in Postman User Interface Help-Clear Cache and Reload
6. Check you're on the latest version of Postman
7. Run the 701 collection to validate the SAP Analytics Cloud endpoints are responding correctly
8. Check your data file is valid
    - simplify your data file to a single entry
    - revert to the sample data files
9. Go back to step 1!

If you believe the issue is with SAP, then please refer to Getting Support

Important official references

Official documentation for the Embedded Edition

Feature Scope for Embedded Edition

SAP Analytics Cloud, Embedded Edition: Getting Started Guide

Non-script, non-API errors

Postman collections can fail due to errors caused neither by the SAP Analytics Cloud API, nor the collection script code or its design. Rather these are caused by either client or network issues.

An example includes:

- 'Error: socket hang up'.
    - This can be observed in the Postman console. Often the API call will have been successful. Sometimes though, Postman stops processing at this point.

Under these conditions, Postman recommend use of the web-based version of Postman over the client desktop application. Postman are likely to recommend the command-line version over the web-based version.

Bugs

These samples span multiple technologies, and it may not be obvious which technology is the cause of your puzzlement! Do take your time to isolate where the issue is. There are multiple channels of support and assistance:

- Postman support and its associated community forum. These channels should be used when you believe there's an issue with Postman, for example if Postman has stability issues, or misbehaves in some way.
- SAP Product Support should be used when you believe there is a defect with the SAP Analytics Cloud API itself. SAP Product Support would likely wish to see the issue reproduced with a single request, or minimal number of requests. Remember, SAP Product Support don't support these sample scripts, they are provided 'as is'.
- The SAP community https://community.sap.com/ is likely to be helpful with progressing general questions or observations of the API when you're not convinced the issue is a defect.
- Finally, the author of these scripts, whilst is unlikely to help with individual issues, would be pleased to hear of any bugs. The total number of lines of code across all the samples amounts to over 10 thousand. So, there's most likely a few bugs somewhere! When you find a bug, please report it to the author and share it for others to benefit. As these samples are provided 'as is'' there is no official support or channel by which bugs will be resolved. The source code is open for a good reason.

Provided ‘as is’

These samples are being provided “AS IS”, without any warranty obligations whatsoever on the part of SAP. SAP makes no express or implied warranties of any type, including, but not limited to, implied warranties of merchantability and of fitness for a particular purpose.

About and contact

For related wiki content, PPT download, feedback, questions & answers please visit
https://blogs.sap.com/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Sample scripts created by Matthew Shaw, SAP.
https://people.sap.com/matthew.shaw/
https://twitter.com/MattShaw_On_BI

Data protection

Many data files you create for these scripts to operate may contain personal data. Please ensure you follow your organisation guidelines regarding processing and handling of this sensitive data to ensure compliance with the law.

Collection documentation

Each sample script is documented with the following properties, and these are described here:

Sample	The name of the sample collection.
Basic description	A high-level description of what the use-case the sample provides
Ideal when	Description of when the sample collection would be suitable
Not suitable when	Description of when the sample collection is not suitable
Notes	Handy notes that typically compare one sample with another or share personate information or advice.
Data file syntax	Defines the column names or the properties of the csv or json file. The notation “F: .csv and .json” means that both csv and json files can be used, where as “F: .json” means only a json file can be used. Csv files can only be used when the data file doesn’t contain a ‘,’ as part of the data or when an array of values is used. For all samples, example csv and json files are provided as a template for you to copy to help ensure you define the data file correctly.
Environment	Specifies the variables that need to be set for the sample to work
How the script works	Describes at a high level how the sample works and the logic behind the sample.
Sample Data Files	For a few of the collections, the sample data files require additional explanation.
Related Scenarios	Related Scenarios will be listed if applicable
Step-by-step instructions	Reference to where step-by-step instructions can be found

Embedded 701-Test Tenant Environment Setup

Sample	Embedded 701-Test Tenant Environment Setup
Basic description	Tests the environment is set correctly for scripts that work
Ideal when	You'd like to validate the Postman Environment is set correctly or not
Notes	It's a harmless script, it will only read from the API, no updates are performed.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform,
How the script works	It reads variable values from the environment and makes two APIs requests based upon their settings. The first API request obtains an access token. The access token is stored as another environmental variable and is represented as part of the header is the subsequent request. The second API request is to the API endpoint and reads the system configuration. The request also returns the total number of OAuth Clients, Live Connections and Trusted Origins and this is output to the console. During both calls, the console log is particularly verbose and writes out very helpful feedback to what variables are set to what values and what the response was from the API for each of the two calls. There are 2 tests in this Postman Collection: 1. GET accesstoken 2. READ System Configuration
Step-by-step instructions	Step-by-step instructions are available for this sample script in the 'Initial Setup with Postman' section of this document under 'Step E'

Embedded 706-Auto Configure Postman Environment for SCIM

Sample	Embedded 706-Auto Configure Postman Environment for **SCIM**
Basic description	Configures your Postman Environment to work with the SCIM API
Ideal when	An OAuth client has already been defined in the SAP Analytics Cloud Service for SCIM use, but you've yet to define the correct Postman Environment variables to use it. Sample 701 must have passed its tests for this to work.
Notes	It's a harmless script, it will only read from the API, no updates are performed.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName (optional) And updates the Postman Environment variables: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName
How the script works	Makes a single request to the /api/v1/tenant/{{Embedded_tenant_uuid}} endpoint Firstly, it looks to see if the OAuth client defined in the tenant matches the one named in the Postman Environment variable Embedded_OAuthClientName. If it matches the name, it then checks this OAuth client has the role PROFILE:sap.epm:SCIM_Public_API and API Access is enabled. If it does, it 'selects' this OAuth client. Otherwise, it will select the first OAuth Client it finds that has the role PROFILE:sap.epm:SCIM_Public_API and API Access is enabled. Once the OAuth client has been 'selected', it will update all the Postman Environment variables for the SCIM API to work: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName There are 2 tests in this Postman Collection: 1. UPDATE Postman Environment o This will pass if the API call is successful 2. OAuth Client includes the SCIM API role with API Access o This will pass if a valid OAuth Client has been found
Step-by-step instructions	Step-by-step instructions are available for this sample script in the 'Step-by-step instructions – Auto Configure Postman Environment' section of this document.

Embedded 707-Auto Configure Postman Environment for Modelling

Sample	Embedded 707-Auto Configure Postman Environment for **Modelling**
Basic description	Configures your Postman Environment to work with the Modeling API
Ideal when	An OAuth client has already been defined in the SAP Analytics Cloud Service for Modelling, but you've yet to define the correct Postman Environment variables to use it. Sample 701 must have passed its tests for this to work.
Notes	It's a harmless script, it will only read from the API, no updates are performed.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName (optional) And updates the Postman Environment variables: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName
How the script works	Makes a single request to the /api/v1/tenant/{{Embedded_tenant_uuid}} endpoint Firstly, it looks to see if the OAuth client defined in the tenant matches the one named in the Postman Environment variable Embedded_OAuthClientName. If it matches the name, it then checks this OAuth client has the role PROFILE:sap.epm:Modeling_Public_API and API Access is enabled. If it does, it 'selects' this OAuth client. Otherwise, it will select the first OAuth Client it finds that has the role PROFILE:sap.epm:Modeling_Public_API and API Access is enabled. Once the OAuth client has been 'selected', it will update all the Postman Environment variables for the SCIM API to work: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName There are 2 tests in this Postman Collection: 1. UPDATE Postman Environment o This will pass if the API call is successful 2. OAuth Client includes the Modeling API role with API Access o This will pass if a valid OAuth Client has been found
Step-by-step instructions	There are no step-by-step instructions are available, though sample script 706 which is very similar does and they are in the 'Step-by-step instructions – Auto Configure Postman Environment' section of this document.

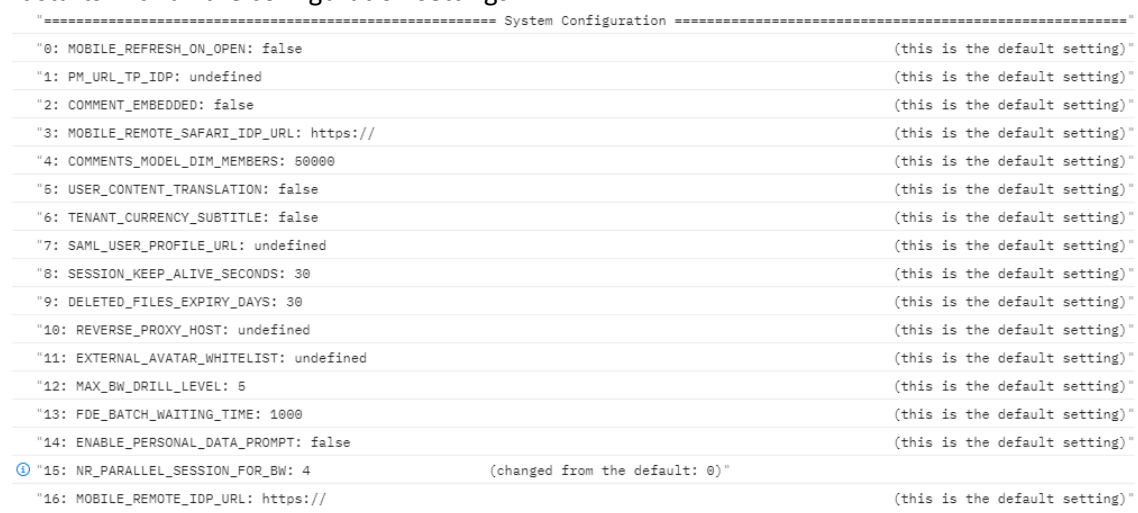
Embedded 708-Auto Configure Postman Environment for Story Listing

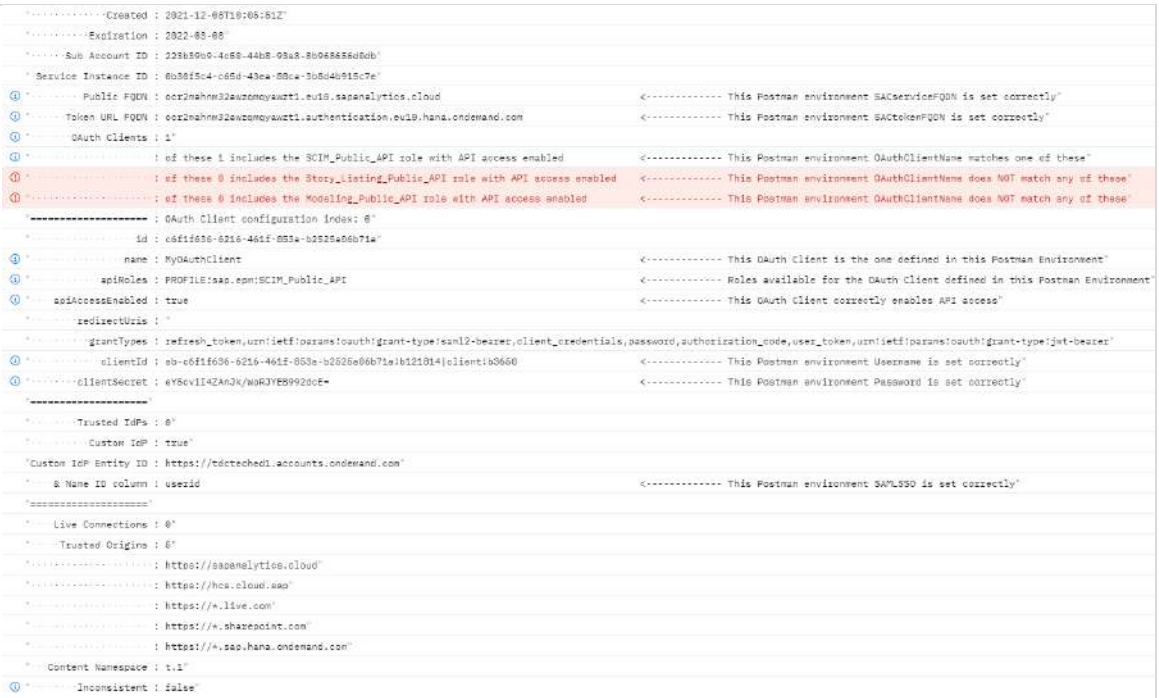
Sample	Embedded 708-Auto Configure Postman Environment for **Story Listing**
Basic description	Configures your Postman Environment to work with the Story Listing API
Ideal when	An OAuth client has already been defined in the SAP Analytics Cloud Service for Story Listing, but you've yet to define the correct Postman Environment variables to use it. Sample 701 must have passed its tests for this to work.
Notes	It's a harmless script, it will only read from the API, no updates are performed.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName (optional) And updates the Postman Environment variables: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName

How the script works	Makes a single request to the /api/v1/tenant/{{Embedded_tenant_uuid}} endpoint
	
	Firstly, it looks to see if the OAuth client defined in the tenant matches the one named in the Postman Environment variable Embedded_OAuthClientName. If it matches the name, it then checks this OAuth client has the role PROFILE:sap.epm:Story_Listing_Public_API and API Access is enabled. If it does, it 'selects' this OAuth client.
	
	Otherwise, it will select the first OAuth Client it finds that has the role PROFILE:sap.epm:Story_Listing_Public_API and API Access is enabled.
	
	Once the OAuth client has been 'selected', it will update all the Postman Environment variables for the SCIM API to work: SACserviceFQDN, SACtokenFQDN, Username, Password, SAMLSSO, ContentNamespace, Embedded_OAuthClientName
	
	There are 2 tests in this Postman Collection:
	1. UPDATE Postman Environment
	  ○ This will pass if the API call is successful
	2. OAuth Client includes the Modeling API role with API Access
	  ○ This will pass if a valid OAuth Client has been found

Step-by-step instructions	There are no step-by-step instructions are available, though sample script 706 which is very similar does and they are in the 'Step-by-step instructions – Auto Configure Postman Environment' section of this document.

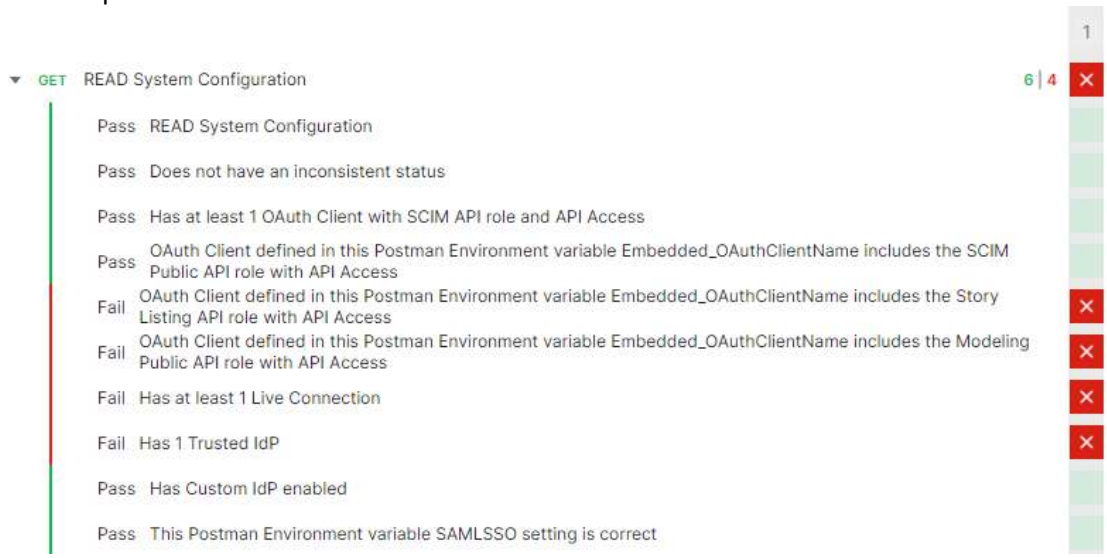
Embedded 711-E-Display & Check System Configuration

Sample	Embedded 711-E-Display & Check System Configuration
Basic description	Reads the entire configuration and prints that out via the console log.
Ideal when	• You'd like to understand how the system has been configured • You'd like to document the current configuration • Any form of troubleshooting is required as a great number of 'tests' will quickly indicate the progress of any system configuration
Not suitable when	It's always suitable!
Notes	It's a harmless script, it will only read from the API, no updates are performed.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName (optional)
How the script works	Makes a single request to the /api/v1/tenant/{{Embedded_tenant_uuid}} endpoint It will print out the entire configuration to the console log. It starts with all the configuration settings:  There are approx. 52 in total. Setting will be shown if they match of differ from the default settings as shown in the image above for entry 15 where the setting is 4 which was changed from the default 0. The layout makes it easy to spot what setting have been altered from the default. It then prints out the summary:



For the parameters that are defined in the Postman Environment, it identifies if the variables are set correctly or not. In the example above, 2 errors are shown because the 'selected' OAuth client does not support Story Listing and Modeling. This isn't a problem if you're not interested in either of course. (The 'selected' OAuth client is the one named in the Postman Environment variable Embedded_OAuthClientName)

An example of the Postman tests:



There are lots of tests in this Postman Collection, but not all necessarily need to pass. For example, above shows tests failing for the Story and Modelling, but this is only a problem if you want to use the API for those purposes. And the 'Has 1 Trusted IdP' is only needed when Trusted IdPs are required, and this is when OAuth to OAuth trust is required for certain live connections to work.

The tests are:

1. READ System Configuration
2. Does not have an inconsistent status

	3. Has at least 1 OAuth Client with SCIM API role and API Access 4. OAuth Client defined in this Postman Environment variable Embedded_OAuthClientName includes the SCIM Public API role with API Access 5. OAuth Client defined in this Postman Environment variable Embedded_OAuthClientName includes the Story Listing API role with API Access 6. OAuth Client defined in this Postman Environment variable Embedded_OAuthClientName includes the Modeling Public API role with API Access 7. Has at least 1 Live Connection 8. Has at least 1 Trusted IdP 9. Has Custom IdP enabled 10. This Postman Environment variable SAMLSSO setting is correct
Step-by-step instructions	There is no need to select a data file, just select the correct Postman Environment and run the sample. Open the console to display the scripts output.

Embedded 721-E-SCIM Express setup (based on this Environment)

Sample	Embedded 721-E-SCIM Express setup (based on this Environment)
Basic description	Creates a new OAuth client for SCIM purposes, Changes the Namespace of the SAC Service and Updates the Postman Environment variables for the SCIM API Sample Scripts to work, without the need to manually update these variables.
Ideal when	• You have a brand-new SAC Embedded Edition tenant, and you'd like to setup the environment as quickly as you can.
Not suitable when	• An OAuth client has already been created. (Use 711 to see which OAuth clients have already been created. Use 706, 707 or 708 to automatically configure your Postman Environment to use an existing OAuth Client) • You are unsure or have a doubt about updating the Content Namespace.
Notes	It uses the value held in the Postman Environment variable 'ContentNamespace' to update the Namespace of the tenant. It uses the value held in the Postman Environment variable 'Embedded_OAuthClientName' as the name of a new OAuth Client it will create with SCIM API role access. Takes about 3 or 4 minutes for the script to complete.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName, ContentNamespace
How the script works	A request is made to read the system configuration which determines which subsequent steps are needed or not: Create OAuth Client: A new OAuth client with the name defined in the Postman Environment variable 'Embedded_OAuthClientName' will be created if there is no existing OAuthClient of the same name. The new OAuthClient will be given the SCIM API role access. Update Content Namespace: The Content Namespace will be updated to match the one defined in the Postman Environment variable 'ContentNamespace'. This means, if you'd like to keep the existing ContentNamespace the tenant already has and still run this sample, then you should update this variable to match the existing value for the Content Namespace of your SAP Analytics Cloud Embedded Edition. Sample script 711 will display this value to the console log. Update Postman Environment: The variables SACserviceFQDN, SACtokenFQDN, Username, Password and SAMLSSO will all be updated saving you from having to update these yourself. It means the SCIM API Sample Scripts can be used immediately. For both the 'Create OAuth Client' and 'Update Content Namespace' a PATCH API request is made to 'request the update'. This API request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'request to update': 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur. OAuthClient creation can take five minutes to complete, and the Namespace typically completes within 30 seconds.

Step-by-step instructions	Step-by-step instructions are available for this sample script in the 'Step-by-step instructions – Express Setup – First time setting up…' section of this document.

Embedded 723-E-Delete OAuth Client (based on this Environment)

Sample	Embedded 723-E-Delete OAuth Client (based on this Environment)
Basic description	Deletes the OAuth client with the name as specified in the Postman Environment variable 'Embedded_OAuthClientName'.
Ideal when	• You'd like to delete the OAuthClient without having to specify the name of OAuthClient in the data file (like you do with sample script 751), you'd rather just specify the name in the Postman Environment variable.
Not suitable when	• You have multiple OAuthClients to delete, as its easier to use sample script 751 and provide a data file with the names of all OAuthClients you'd like to delete
Notes	It uses the value held in the Postman Environment variable 'Embedded_OAuthClientName' as the name of the OAuth Client it will delete. Takes about 2 ½ minutes for the script to complete.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform, Embedded_OAuthClientName
How the script works	A request is made to read the system configuration which determines which subsequent steps are needed or not: Delete OAuth Client: The OAuth client with the name defined in the Postman Environment variable 'Embedded_OAuthClientName' will be deleted if an existing OAuthClient of the same name already exists. For the 'Delete OAuth Client' a PATCH API request is made to 'request the deletion'. This API request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'request the deletion: 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur. OAuthClient deletion can take three minutes to complete.
Step-by-step instructions	There are currently no step-by-step instructions for this sample.

Embedded 731-E-Reset Inconsistent state

Sample	Embedded 731-E-Reset Inconsistent state
Basic description	Resets the 'inconsistent state' of the SAP Analytics Cloud Embedded Edition should it become inconsistent.
Ideal when	• The SAP Analytics Cloud Embedded Edition is in an inconsistent state, and you need to 'reset' it to continue configuring it. • You have not previously used these sample scripts, since using these sample scripts will almost certainly ensure the SAP Analytics Cloud Embedded Edition tenant remains perfectly consistent! The samples are written to ensure the requests made are perfectly formed and validated.
Not suitable when	• The SAP Analytics Cloud Embedded Edition is not inconsistent.
Notes	Its harmless to run this sample even when the status is not inconsistent, since it will not make any changes. Changes are only made when the status of 'inconsistent' is true.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform
How the script works	A request is made to read the system configuration which determines if the subsequent steps are needed or not: Reset Tenant: If the tenant's 'inconsistent' status is true, then a request is made to reset it. Otherwise, the script will abort. For the 'Reset Tenant' a PATCH API request is made to 'reset the inconsistent status'. This API request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'reset tenant': 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.
Step-by-step instructions	There are currently no step-by-step instructions for this sample.

Embedded 732-E-Display SAML metadata

Sample	Embedded 732-E-Display SAML metadata
Basic description	Returns the SAML metadata from SAP Analytics Cloud Embedded Edition
Ideal when	You need to setup SAML SSO and establish a trust relationship with SAP Analytics Cloud Embedded Edition and your own custom Identity Provider.
Not suitable when	You are using the default Identify Provider that comes with SAP Analytics Cloud Embedded Edition.
Notes	Its harmless to run this sample, it only returns the SAML metadata. No updates are performed. For more information on this request please refer to the official documentation.
Data file syntax	There are no data file requirements
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform
How the script works	A request is made to read the SAML metadata which is then displayed in the Postman Console log.
Step-by-step instructions	Step-by-step instructions are available for this sample script in the 'Step-by-step instructions – Configuring SAML Single-Sign-On' section of this document.

Embedded 733-Fj-Configure Custom IdP

<table>
<tr><td>Sample</td><td>Embedded 733-Fj-Configure Custom IdP</td></tr>
<tr><td>Basic description</td><td>Configures Customer Identity Provider</td></tr>
<tr><td>Ideal when</td><td>• You need to setup SAML SSO and establish a trust relationship with SAP Analytics Cloud Embedded Edition and your own custom Identity Provider.</td></tr>
<tr><td>Not suitable when</td><td>• You are using the default Identify Provider that comes with SAP Analytics Cloud Embedded Edition.</td></tr>
<tr><td>Notes</td><td>Running this script will change the authentication method of SAP Analytics Cloud Embedded Edition. Once the authentication method has changed from the default, it cannot be changed back. The authentication method can only be updated with new metadata or to specify a different nameIdColumn.
The idPMetadata must be ‘JSON encoded’ (explained in the step-by-step instructions)</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table><tr><th>Field</th><th>Type</th><th>Description</th></tr><tr><td>file_idPMetadata</td><td>string</td><td>Metadata from the Custom Identity Provider that has already been JSON-encoded.</td></tr><tr><td>file_nameIdColumn</td><td>string</td><td>Specifies the name id column to map the user to. It must be either: userid, email or custom</td></tr></table>
The fields used in the data file match the attributes used in the request to the API. For example ‘file_idPMetadata’ matches the attribute ‘idPMetadata’. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration.
A validation is then performed to check that the SAP Analytics Cloud Embedded Edition is not in an inconsistent status and that the current meta data or the nameId column is different from that as specified in the data file (i.e. an update is necessary). A check is also made that the nameId column specified is supported.

If the validation passes, then a PATCH request is made to update the custom identity provider configuration. In this PATCH request, the metadata is further ‘JSON-encoded’ so the entry can be successfully sent to the SAP API.

The ‘Configure IdP’ PATCH request will return a ‘log id’. The ‘log id’ is then used by another request the return the status of that first ‘Configure IdP’ request: ‘IN PROGRESS, FAILED, SUCCESS’ etc. The log request is repeated, every 30 seconds, until the status is anything other than ‘IN PROGRESS’, when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Updating the IdP Configuration typically takes about a minute.

Once successful, the Postman Environment variable ‘SAMLSSO’ is updated to the new setting, thus ensuring the SCIM API sample scripts work as expected.</td></tr>
</table>

Step-by-step instructions	Step-by-step instructions are available for this sample script in the ‘Step-by-step instructions – Configuring SAML Single-Sign-On’ section of this document. These instructions include the process of ‘JSON encoding’ the metadata file.
Sample Data Files	733 sample example 1 - custom idp.json This sample data file shows the idPMetadata after it has been ‘JSON encoded’

Embedded 734-Fj-Update System Configuration

Sample	Embedded 734-Fj-Update System Configuration
Basic description	Updates System Configurations
Ideal when	You need to update a system configuration
Not suitable when	n/a
Notes	This is ideal for making updates to the system configuration. Use sample script 711 to display the full system configuration
Data file syntax	F: .json

Field	Type	Description

`file_SystemConfig`	JSON array of paired name and value entries	JSON array of paired "name" and "value" entries. One pair for each configuration, for example
		`[{"name": "MOBILE_REFRESH_ON_OPEN","value": "false"}]`
		There needs to have at least 1 pair, but multiple can be specified, for example:
		`[{"name": "MOBILE_REFRESH_ON_OPEN","value": "false"}, {"name": "COMMENT_EMBEDDED", "value": "false"}]`

The fields used in the data file match the attributes used in the request to the API. For example, 'file_SystemConfig' matches the attribute 'SetTenantSystemConfig'. For more information on these attributes please refer to the official documentation.

This table shows all the possible configuration names and their default settings:

Configuration	Default Value
MOBILE_REFRESH_ON_OPEN	false
PM_URL_TP_IDP	
COMMENT_EMBEDDED	false
MOBILE_REMOTE_SAFARI_IDP_URL	https://
COMMENTS_MODEL_DIM_MEMBERS	50000
USER_CONTENT_TRANSLATION	false
TENANT_CURRENCY_SUBTITLE	false
SAML_USER_PROFILE_URL	
SESSION_KEEP_ALIVE_SECONDS	
DELETED_FILES_EXPIRY_DAYS	30
REVERSE_PROXY_HOST	
EXTERNAL_AVATAR_WHITELIST	
MAX_BW_DRILL_LEVEL	5
FDE_BATCH_WAITING_TIME	1000
ENABLE_PERSONAL_DATA_PROMPT	false
NR_PARALLEL_SESSION_FOR_BW	0
MOBILE_REMOTE_IDP_URL	https://
ENABLE_ON_PREMISE_FILE_EXPORT	false
TENANT_METRIC_NO_DATA_FORMAT	
ALLOW_SCHEDULE_PUBLICATION	true
AR_SESSION_TIMEOUT_V2	3600
MOBILE_DEFAULT_FILTER	0
DEFAULT_APP	0
COMMENTS_PER_MODEL_LIMIT	3000
MOBILE_REMOTE_SAFARI_SAML	false
TENANT_NO_DATA_FORMAT	

BW_RESPECT_VIZ_DEFAULTING	false
TENANT_CURRENCY_FORMAT	
BROWSER_CACHE_STORAGE_TIME	8
EXPORT_PACKAGE_SIZE	50000
DISABLE_MOBILE_APP_PASSWORD	false
ENABLE_ON_PREMISE_FILE	false
ALLOW_SHARING_TO_ALL_USERS	true
PREDICTIVE_BI_FORECAST_REMOTE	false
DISABLE_MOBILE_CACHING_IOS	false
ENABLE_EXPORT_IMPORT_JOB	false
COULD_DEL_DISCUSSION	true
REMOVE_STORY_URL_FROM_APPENDIX	false
TRACE_LEVEL	4
CUSTOMIZE_COMMUNITY_URL	
X509_ISSUER_NAME	CN=SSO_CA, O=SAP-AG, C=DE
GEO_LIVE_SYNONYM_SUPPORT	false
ALLOW_PUBLICATION_BURSTING	false
ALLOW_NON_SAC	true
MOBILE_DEFAULT_TAB	false
TENANT_SHOW_CURRENCY_AS	
MOBILE_REMOTE_TOKEN	HEADER_KEY_1=<<token>>
TENANT_SCALE_FORMAT	
CHART_PROGRESSIVE_RENDERING	false
ALLOW_ACN_COPY_CONTENT	false
ALLOW_ACN_PACKAGE_SHARING_OEM	false

There is no need to specify all the configuration name pairs, you only need to specify the configuration that needs to be updated.

Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform
How the script works	The data file is read. For any entries that have an empty string value for the 'value' a warning error is displayed to the console saying the value will be ignored and it will not form part of the system configuration update. This is because the API cannot have null values for the 'value'. In turn, this means that once a parameter has been set, it is not possible to 'unset' it. The list of system configurations that have a 'null' value by default is: 1. PM_URL_TP_IDP 2. SAML_USER_PROFILE_URL 3. REVERSE_PROXY_HOST 4. EXTERNAL_AVATAR_WHITELIST 5. TENANT_METRIC_NO_DATA_FORMAT 6. TENANT_NO_DATA_FORMAT 7. TENANT_CURRENCY_FORMAT 8. CUSTOMIZE_COMMUNITY_URL 9. TENANT_SHOW_CURRENCY_AS 10. TENANT_SCALE_FORMAT It means that once any of the above has been set, it's not possible to unset them. The 'System Configuration' PATCH request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'System Configuration' request: 'IN

	PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur. Configuration updates will typically complete within 30 seconds.
Step-by-step instructions	There are currently no step-by-step instructions available for this sample script.
Sample Data Files	There are 5 sample data files. The first contains all the settings with their default values. The others are a subset, again all with the default settings, but designed for you to easily update the 'value' 734 sample example 1 - System Configuration - default settings.json 734 sample example 2 - System Configuration - content namespace.json 734 sample example 3 - System Configuration - mobile settings.json 734 sample example 4 - System Configuration - common.json 734 sample example 5 - System Configuration - BW related.json

Embedded 735-Oarr-Fj-Update Trusted Origins

<table>
<tr><td>Sample</td><td>Embedded 735-Oarr-Fj-Update Trusted Origins</td></tr>
<tr><td>Basic description</td><td>Adds, removes to replaces the list of trusted origins</td></tr>
<tr><td>Ideal when</td><td>You are embedding SAP Analytics Cloud Embedded Edition within another web application such as including it in an ‘iframe’</td></tr>
<tr><td>Not suitable when</td><td>n/a</td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of trusted origins. You can then use this sample script to either add, remove or replace those origins listed.</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table><tr><th>Field</th><th>Type</th><th>Description</th></tr><tr><td><code>file_action</code></td><td>string</td><td>The action to perform on the list of trusted origins already registered. This can be ‘add’, ‘remove’ or ‘replace’</td></tr><tr><td><code>file_trustedOrigins</code></td><td>array</td><td>An array of trusted origins. For example ["https://sapanalytics.cloud", "https://hcs.cloud.sap", "https://*.live.com", "https://*.sharepoint.com", "https://*.online.office.de"]</td></tr></table>
For more information on trusted origins please refer to the official documentation and a special section on embedding iframes</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of trusted origins.

Checks are then made to compare the existing list of trusted origins and the desired settings as specified in the data file.

Updating Operations (arr)
The <code>file_action</code> can be “<code>add</code>”, “<code>remove</code>” or “<code>replace</code>”:
<code>add</code>: will add the origins to the list of origins.
<code>remove</code>: will remove the origins from the list of origins.
<code>replace</code>: will replace (set) the list of origins (i.e. remove and add accordingly).

Once the comparison is complete and assuming an update is required, then a PATCH request to made to update the list of trusted origins.

The ‘Update Trusted Origins’ PATCH request will return a ‘log id’. The ‘log id’ is then used by another request the return the status of that first ‘Update Trusted Origins’ request: ‘IN PROGRESS, FAILED, SUCCESS’ etc. The log request is repeated, every 30 seconds, until the status is anything other than ‘IN PROGRESS’, when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Updating the IdP Configuration typically takes less than 30 seconds.</td></tr>
<tr><td>Step-by-step instructions</td><td>There are currently no step-by-step instructions are available for this sample script.</td></tr>
<tr><td>Sample Data Files</td><td>There are 3 sample data files:</td></tr>
</table>

	735 sample example 1 - demo trusted origins.json 735 sample example 2 - add trusted origins.json 735 sample example 3 - replace trusted origins.json The last example contains the likely default set of trusted origins.

Embedded 741-Fcj-Add OAuth Client

<table>
<tr><td>Sample</td><td>Embedded 741-Fcj-Add OAuth Client</td></tr>
<tr><td>Basic description</td><td>Adds an OAuth Client to your SAP Analytics Cloud Embedded Edition</td></tr>
<tr><td>Ideal when</td><td>You'd like to add another OAuth Client and customise the apiRoles that OAuth Client should have. For example, the sample script '721 – Express Setup' will add an OAuth Client, but it will only grant that OAuth Client the apiRole 'SCIM_Public_API'. If you need an OAuth Client with others roles, then you'll need to use this sample script.</td></tr>
<tr><td>Not suitable when</td><td>You'd like to update an existing OAuth Client. Then, you must delete the OAuth client and add another as you cannot update an existing OAuth Client.</td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of OAuth Clients.</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td><code>file_name</code></td><td>string</td><td>The name of the OAuth Client you are going to add. A friendly name is advised.</td></tr>
<tr><td><code>file_redirectUris</code></td><td>string</td><td>String for the URIs. Often left blank.</td></tr>
<tr><td><code>file_apiRoles</code></td><td>array</td><td>List of API Roles the OAuth Client should have. For example ["PROFILE:sap.epm:Story_Listing_Public_API", "PROFILE:sap.epm:SCIM_Public_API", "PROFILE:sap.epm:Modeling_Public_API"]</td></tr>
</table>

The fields used in the data file match the attributes used in the request to the API. For example, 'file_name' matches the attribute 'name'. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of OAuth Clients.

A check is then made comparing the current list of OAuth Clients with the name of the one in the data file. If the one named in the file does not yet exist, then a further request is made to add the OAuth Client with the desired roles as specified in the data file.

The 'Add OAuth Client' PATCH request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'Add OAuth Client' request: 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Adding an OAuth Client can take anywhere from 90 seconds to 5 minutes depending upon the data centre.

The script does NOT update the Postman Environment 'Embedded_OAuthClientName'. This variable is used by other scripts to update other Postman Environment variables for the SCIM API samples to work without the need to manually update them. It means, that you must either update this variable yourself, or use the samples 706, 707 or 708 as appropriate.</td></tr>
</table>

Step-by-step instructions	There are currently no step-by-step instructions are available for this sample script.
Sample Data Files	There are 4 sample data files: 741 sample example 1 - Add 4 OAuth Clients.json 741 sample example 2 - Add SCIM OAuth Client.json 741 sample example 3 - Add Story OAuth Client.json 741 sample example 4 - Add Modeling OAuth Client.json The first example adds 4 OAuth Clients. The first client has the Story Listing role, the second has the SCIM role, the third has Modeling role and the fourth has all three roles. The examples 2, 3 and 4, all add just a single OAuth Client with their respective role for SCIM, Story Listing, and Modeling.

Embedded 742-Fcj-Add Trusted IdP

<table>
<tr><td>Sample</td><td>Embedded 742-Fcj-Add Trusted IdP</td></tr>
<tr><td>Basic description</td><td>Adds a Trusted IdP to your SAP Analytics Cloud Embedded Edition</td></tr>
<tr><td>Ideal when</td><td>You’d like to add trusted IdP to enable server-to-server communication.</td></tr>
<tr><td>Not suitable when</td><td>You’re configuring SAML SSO or updating the authentication method used by SAP Analytics Cloud Embedded Edition. Do not get confused with configuring SAML SSO which is quite different. Please then use sample scripts 732 and 733.</td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of Trusted IdPs.</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td>file_name</td><td>string</td><td>The name of the IdP you are going to add, must be unqiue</td></tr>
<tr><td>file_idpName</td><td>string</td><td>The name of the IdP you are going to add</td></tr>
<tr><td>file_idpCertificate</td><td>string</td><td>The metadata of the IdP you are adding.</td></tr>
</table>

The fields used in the data file match the attributes used in the request to the API. For example, ‘file_name’ matches the attribute ‘name’. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of Trusted IdPs.

A check is then made comparing the current list of IdPs with the name of the one in the data file. If the one named in the file does not yet exist, then a further request is made to add the Trusted IdP as specified in the data file.

The ‘Add Trusted IdP’ PATCH request will return a ‘log id’. The ‘log id’ is then used by another request the return the status of that first ‘Add Trusted IdP’ request: ‘IN PROGRESS, FAILED, SUCCESS’ etc. The log request is repeated, every 30 seconds, until the status is anything other than ‘IN PROGRESS’, when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Adding a Trusted IdP typically takes less than a minute.</td></tr>
<tr><td>Step-by-step instructions</td><td>There are currently no step-by-step instructions are available for this sample script.</td></tr>
<tr><td>Sample Data Files</td><td>There is just 1 sample data file:
742 sample example 1 - Add Trusted IdP.json</td></tr>
</table>

Embedded 743-Fj-Add Live Connection

<table>
<tr><td>Sample</td><td>Embedded 743-Fj-Add Live Connection</td></tr>
<tr><td>Basic description</td><td>Adds a Live Connection to your SAP Analytics Cloud Embedded Edition</td></tr>
<tr><td>Ideal when</td><td>You'd like to add a new live connection</td></tr>
<tr><td>Not suitable when</td><td>You want to update an existing live connection, instead you must delete the connection and create a new one. This could be problematic if you need the same connection ID shared across the landscape when you have a 'dev' and 'prod' setup for life-cycle management. Here the connection ID needs to be respected whilst the contents of the connection may need to point to different data sources. An internal SAP reference FPA34-7489 is used to track this requirement.</td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of Live Connections.
Importantly the 'name' value is used as the basis to determine the 'id' of the connection. It is important that the same 'id' is used across the landscape when using different environments for dev, test and production.</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td>file_liveconnection</td><td>array</td><td>Array of 'name' and 'value' pairs that defines the connection.

For example:

<pre>[{"name": "type","value": "EXTENDEDHANA"},
{"name": "name","value": "MyExtHana"},
{"name": "description", "value": "My Extended Live HANA
connection description"},
{"name": "host", "value": "myhostname"},
{"name": "port","value": "443"},
{"name": "language", "value": "EN"},
{"name": "authenticationMethod", "value": "BASIC"},
{"name": "userName", "value": "myusername"},
{"name": "password", "value": "mypassword"},
{"name": "isSaveCredential", "value": true},
{"name": "contextPath", "value": ""},
{"name": "proxyType", "value": "premise"},
{"name": "proxyHost", "value": "myproxyhost"},
{"name": "proxyPort", "value": "443"},
{"name": "locationId","value": "loc1"}]</pre></td></tr>
</table>

The name value pairs used in the data file match the attributes used in the request to the API. For example, 'type' matches the attribute 'type'. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of Live Connections.

Many checks are made to validate the connection definition before a further request is then made to add the new live connection.

These validation checks include:
• Ensure the connection with the same name doesn't already exist
• The 'type' is either HANA or EXTENDEDHANA
• The connection name length is not more than 10 characters in length
• The 'host' is defined
• The 'authenticationMethod' is either SAML or BASIC</td></tr>
</table>

	• If the 'authenticationMethod' is BASIC, then username and password are specified • If the 'type' is EXTENDEDHANA then 'proxyType' is specified and is either default, internet or premise • If the 'proxyType' is 'premise' then proxyHost and proxyPort are specified In general, always set the type to be 'HANA' and not 'EXTENDEDHANA'. The option for EXTENDEDHANA is needed when connecting to a data source provided as part of an SAP line of business. When adding a connection of type 'HANA' will automatically set the connectionType to Direct. Any parameter that is in the array of will be sent to SAP Analytics Cloud. The list of value pairs is not filtered in any way. It means, if undocumented value pairs are used, they will be sent as part of the connection definition creation. The 'Add Live Connection' PATCH request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'Add Live Connection' request: 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur. Adding a Live Connection typically takes less than 30 seconds.
Step-by-step instructions	There are currently no step-by-step instructions are available for this sample script. However, there is a related blog https://blogs.sap.com/2021/09/02/getting-started-with-sap-analytics-cloud-embedded-edition-btp-service.-part-ii/ When referring to this blog, step 5 is covered by this sample script. It means for step 5 of that blog, its likely to be much easier to use this sample, rather than the one provided in the blog.
Sample Data Files	There are 3 sample data files: 743 sample example 1 - Add HANA Live.json 743 sample example 2 - Add HANA Ext.json 743 sample example 3 - Add Live Connection all params.json

Embedded 751-Fcj-Delete OAuth Client

<table>
<tr><td>Sample</td><td>Embedded 751-Fcj-Delete OAuth Client</td></tr>
<tr><td>Basic description</td><td>Deletes OAuth Clients</td></tr>
<tr><td>Ideal when</td><td>You’d like to delete an existing OAuth client</td></tr>
<tr><td>Not suitable when</td><td></td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of OAuth Clients.</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td><code>file_name</code></td><td>string</td><td>Name of the OAuth client you would like to delete.</td></tr>
</table>

The field used in the data file match the attributes used in the request to the API. For example, ‘file_name’ matches the attribute ‘name’. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of OAuth Clients.

A check is then made by comparing the name of the OAuth Client in the data file matches the name of an OAuth Client registered in SAP Analytics Cloud.

Assuming the OAuth Client does exist, a request is then made to delete it.

The ‘Delete OAuth Client’ PATCH request will return a ‘log id’. The ‘log id’ is then used by another request the return the status of that first ‘Delete OAuth Client’ request: ‘IN PROGRESS, FAILED, SUCCESS’ etc. The log request is repeated, every 30 seconds, until the status is anything other than ‘IN PROGRESS’, when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Deleting an OAuth Client typically takes less than 30 seconds.</td></tr>
<tr><td>Step-by-step instructions</td><td>There are currently no step-by-step instructions are available for this sample script.</td></tr>
<tr><td>Sample Data Files</td><td>There are 4 sample data files that mirror the samples provided in 741:
751 sample example 1 - Delete 4 OAuth Clients.json
751 sample example 2 - Delete SCIM OAuth Client.json
751 sample example 3 - Delete Story OAuth Client.json
751 sample example 4 - Delete Modeling OAuth Client.json</td></tr>
</table>

Embedded 752-Fcj-Delete Trusted IdP

Sample	Embedded 752-Fcj-Delete Trusted IdP
Basic description	Deletes Trusted IdPs
Ideal when	You'd like to delete an existing Trusted IdP
Not suitable when	
Notes	Use the sample 711 to display the current list of Trusted IdPs.
Data file syntax	F: .json Field / Type / Description `file_name` / string / Name of the Trusted IdP you would like to delete. The field used in the data file match the attributes used in the request to the API. For example, 'file_name' matches the attribute 'name'. For more information on these attributes please refer to the official documentation.
Environment	Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform
How the script works	A request is made to read the system configuration which contains the current list of Trusted IdPs. A check is then made by comparing the name of the Trusted IdP in the data file matches the name of a Trusted IdP registered in SAP Analytics Cloud. Assuming the Trusted IdP does exist, a request is then made to delete it. The 'Delete Trusted IdP' PATCH request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'Delete Trusted IdP' request: 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur. Deleting a Trusted IdP typically takes less than 30 seconds.
Step-by-step instructions	There are currently no step-by-step instructions are available for this sample script.
Sample Data Files	There is 1 sample data file that mirrors the sample provided in 742: 752 sample example 1 - Delete Trusted IdP.json

Embedded 753-Fj-Delete Live Connection

<table>
<tr><td>Sample</td><td>Embedded 753-Fj-Delete Live Connection</td></tr>
<tr><td>Basic description</td><td>Deletes Live Connection</td></tr>
<tr><td>Ideal when</td><td>You'd like to delete an existing Live Connection</td></tr>
<tr><td>Not suitable when</td><td></td></tr>
<tr><td>Notes</td><td>Use the sample 711 to display the current list of Live Connections</td></tr>
<tr><td>Data file syntax</td><td>F: .json

<table><tr><th>Field</th><th>Type</th><th>Description</th></tr><tr><td><code>file_name</code></td><td>string</td><td>Name of the Live Connection you would like to delete.</td></tr></table>
The field used in the data file match the attributes used in the request to the API. For example, 'file_name' matches the attribute 'name'. For more information on these attributes please refer to the official documentation.</td></tr>
<tr><td>Environment</td><td>Embedded_uua_url_FQDN, Embedded_uua_clientid, Embedded_uua_clientsecret, Embedded_endpoints_sac_ embedded_edition_service_ config_FQDN, Embedded_tenant_uuid, SACplatform</td></tr>
<tr><td>How the script works</td><td>A request is made to read the system configuration which contains the current list of Live Connections

A check is then made by comparing the name of the Live Connection in the data file matches the name of a Live Connection registered in SAP Analytics Cloud.

Assuming the Live Connection does exist, a request is then made to delete it.

The 'Delete Live Connection' PATCH request will return a 'log id'. The 'log id' is then used by another request the return the status of that first 'Delete Live Connection' request: 'IN PROGRESS, FAILED, SUCCESS' etc. The log request is repeated, every 30 seconds, until the status is anything other than 'IN PROGRESS', when the final status is then shown, and the related Postman Test is updated with the outcome. The Postman console is worth displaying to see the progress of the script and for any clues to any issues that may occur.

Deleting a Live Connection typically takes less than 30 seconds.</td></tr>
<tr><td>Step-by-step instructions</td><td>There are currently no step-by-step instructions are available for this sample script.</td></tr>
<tr><td>Sample Data Files</td><td>There are 3 sample data files that mirror the samples provided in 742:
743 sample example 1 - Delete HANA Live.json
743 sample example 2 - Delete HANA Ext.json
743 sample example 3 - Delete multiple connections.json</td></tr>
</table>